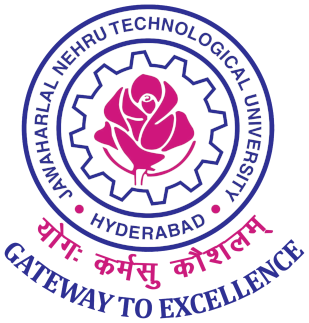
**Jawaharlal Nehru Technological University**

**Department of Computer Science & Engineering**



**CERTIFICATE**

This is to certify that the project entitled **“VIRTUAL CLASSROOM”** is a bonafede work carried out by

**ABHINAV RAJ**

**G S G S SURYA**

**SIMREEN BEGAM**

**LAXMINARAYA**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** from Jawaharlal Nehru Technological University, Hyderabad, under our guidance and supervision.

The results presented in this project have been verified and are found to be satis-factory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide Head of the Department

**Mr. Raghunath Reddy** **Dr. Sheo Kumar**

Professor Professor & H.O.D

Department of CSE, Department of CSE,

**DECLARATION**

This is to certify that the work reported in the present project entitled “**VIRTUAL CLASSROOM**” is record of bonafide work done by us in the Department Of Computer Science and Engineering, CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source.

The results embodied in this project report have not been submitted to any other University or Institute for the reward of any degree to the best of our knowledge and belief.

**ABHINAV RAJ**

**G S G S SURYA**

**SIMREEN BEGAM**

**LAXMINARAYA**

**CONTENTS**

**TOPIC PAGENO**

**ABSTRACT** i

**1. INTRODUCTION** 6

**1.1.** Introduction

**1.2**. Purpose

# 2. LITERATURE SURVEY 7

# 3. SOFTWARE REQUIREMENT ANALYSIS 8

3.1. Introduction

3.2. Analysis Mode

3.3. Existing System

3.4. Proposed System

3.5. Input Design & Output Design

3.6. Main Modules

3.7. Feasibility Study

**4. Hardware & Software Requirements**  12

4.1. Software requirements

4.2. Hardware requirements

**5. SOFTWARE REQUIREMENT SPECIFICATIONS** 13

5.1. Introduction

5.2. Operating Environment

**6. System Design** 14

6.1. Introduction

6.2. E – R Diagrams

6.3. Data Flow Diagrams

6.4. Use Case Diagram

6.5. Activity Diagram

6.6. Sequence Diagram

6.7. Class Diagram

**7. Software Environment** 25

### 7.1. Java Programming Language

7.2. Coding

**8. System Testing** 52

8.1. Introduction

8.2. Unit Testing:

# 8.3. Integration Testing

8.4. Acceptance Testing

**9. Output Screens**  55

**10. Conclusion** 59

**11. Bibliography**  60

**ABSTRACT**

Online reviews have become an important source of information for users before making an informed purchase decision. Early reviews of a product tend to have a high impact on the subsequent product sales. In this paper, we take the initiative to study the behavior characteristics of early reviewers through their posted reviews on two real-world large e-commerce platforms, i.e., Amazon and Yelp(www.yelp.com). In specific, we divide product lifetime into three consecutive stages, namely early, majority and laggards. A user who has posted a review in the early stage is considered as an early reviewer. We quantitatively characterize early reviewers based on their

rating behaviors, the helpfulness scores received from others and the correlation of their reviews with product popularity. We have found that (1) an early reviewer tends to assign a higher average rating score; and (2) an early reviewer tends to post more helpful reviews. Our analysis of product reviews also indicates that early reviewers’ ratings and their received helpfulness scores are likely to influence product popularity. By viewing review posting process as a multiplayer competition game, we propose a novel margin-based embedding model for early reviewer prediction. Extensive experiments on two different e-commerce datasets have shown that our proposed approach outperforms a number of competitive baselines.

**1. INTRODUCTION**

**1.1. INTRODUCTION:**

The emergence of e-commerce websites has enabled users to publish or share purchase experiences by posting product reviews, which usually contain useful opinions, comments and feedback towards a product. As such, a majority of customers will read online reviews before making an informed purchase decision. It has been reported about 71% of global online shoppers read online reviews before purchasing a product. Product reviews, especially

the early reviews (i.e., the reviews posted in the early stage of a product), have a high impact on subsequent product sales . We call the users who posted the early reviews early reviewers. Although early reviewers contribute only a small proportion of reviews, their opinions can determine the success or failure of new products and services. It is important for companies to identify early reviewers since their feedbacks can help companies to adjust marketing strategies and improve product designs, which can eventually lead to the success of their new products. For this reason, early reviewers become the emphasis to monitor and attract at the early promotion stage of a company. The pivotal role of early reviews has attracted

extensive attention from marketing practitioners to induce consumer purchase intentions . For example, Amazon, one of the largest e-commerce companies in the world, has Big Data Management and Analysis Methods, Beijing, China. He is with the School of Engineering and Applied Science, Aston University, United Kingdom is with Department of Computer Science and Operations Research, University of Montreal. advocated the Early Reviewer Program1, which helps to acquire early reviews on products that have few or no reviews.

With this program, Amazon shoppers can learn more about products and make smarter buying decisions. As another related program, Amazon Vine2 invites the most trusted reviewers on Amazon to post opinions about new and prerelease items to help their fellow customers make informed purchase decisions.

**1.2. PURPOSE**

We propose below methodology for solving the problem. Raw data collected would be pre-processed for missing data, anomalies and outliers. Then an algorithm would be trained on this data to create a model. This model would be used for forecasting the final results. ETL stands for Extract, Transform and load. It is a tool which is a combination of three functions. It is used to get data from one database and transform it into a suitable format. Data preprocessing is a data mining technique used to transform sample raw data into an understandable format. Real world collected data may be inconsistent, incomplete or contains an error and hence data preprocessing is required.

# 

The present system is the mannual one. Hence all the information about the Student,courses and faculty details maintained in the file.ForFaculty,they have different-different files for different purpose, Like seprate file for student details, attendence and seprate file for report etc. For Student,they have different-different notebooks for different subjects, sometimes they forget something during lecture. In the present system all work is done on paper. All student who attend physical classroom, write down the notes so time goes in vain due to noting down, drawing figures and in etc. For Faculty, they write whole session attendance in register and at the end of the session the reports are generated. We are not interested in generating report in the middle of the session or as per the requirement because it takes more time in calculation. At the end of session the students who don’t have 75% attendance get a notice so the thing is that we can save our time by using Virtual Classroom System because all the things has been integrated in mean of faculty work, student activity and so on. If we want to do updating in records by pen and paper then it takes a very long boring time but in this system, we can update the record by selecting the relevant information such as faculty information, student information, adding course, exam related information, any news from college or any notices from department.

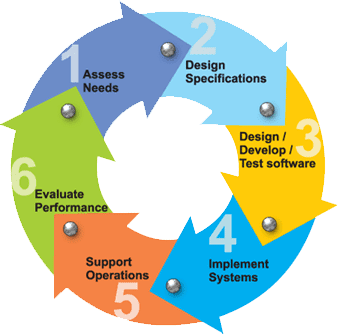
The present system is the mannual one. Hence all the information about the Student,courses and faculty details maintained in the file.ForFaculty,they have different-different files for different purpose, Like seprate file for student details, attendence and seprate file for report etc. For Student,they have different-different notebooks for different subjects, sometimes they forget something during lecture. In the present system all work is done on paper. All student who attend physical classroom, write down the notes so time goes in vain due to noting down, drawing figures and in etc. For Faculty, they write whole session attendance in register and at the end of the session the reports are generated. We are not interested in generating report in the middle of the session or as per the requirement because it takes more time in calculation. At the end of session the students who don’t have 75% attendance get a notice so the thing is that we can save our time by using Virtual Classroom System because all the things has been integrated in mean of faculty work, student activity and so on. If we want to do updating in records by pen and paper then it takes a very long boring time but in this system, we can update the record by selecting the relevant information such as faculty information, student information, adding course, exam related information, any news from college or any notices fro

**3.SYSTEM ANALYSIS**

**3.1 INTRODUCTION**

**Software Development Life Cycle:-**

   There is various software development approaches defined and designed which are used/employed during development process of software, these approaches are also referred as "Software Development Process Models". Each process model follows a particular life cycle in order to ensure success in process of software development.



**Requirements**

Business requirements are gathered in this phase.  This phase is the main focus of the project managers and stake holders.  Meetings with managers, stake holders and users are held in order to determine the requirements.  Who is going to use the system?  How will they use the system?  What data should be input into the system?  What data should be output by the system?  These are general questions that get answered during a requirements gathering phase.  This produces a nice big list of functionality that the system should provide, which describes functions the system should perform, business logic that processes data, what data is stored and used by the system, and how the user interface should work.  The overall result is the system as a whole and how it performs, not how it is actually going to do it.

**Design**

The software system design is produced from the results of the requirements phase.  Architects have the ball in their court during this phase and this is the phase in which their focus lies.  This is where the details on how the system will work is produced.  Architecture, including hardware and software, communication, software design (UML is produced here) are all part of the deliverables of a design phase.

**Implementation**

Code is produced from the deliverables of the design phase during implementation, and this is the longest phase of the software development life cycle.  For a developer, this is the main focus of the life cycle because this is where the code is produced.  Implementation my overlap with both the design and testing phases.  Many tools exists (CASE tools) to actually automate the production of code using information gathered and produced during the design phase.

**Testing**

During testing, the implementation is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase.  Unit tests and system/acceptance tests are done during this phase.  Unit tests act on a specific component of the system, while system tests act on the system as a whole.

So in a nutshell, that is a very basic overview of the general software development life cycle model.  Now let’s delve into some of the traditional and widely used variations.

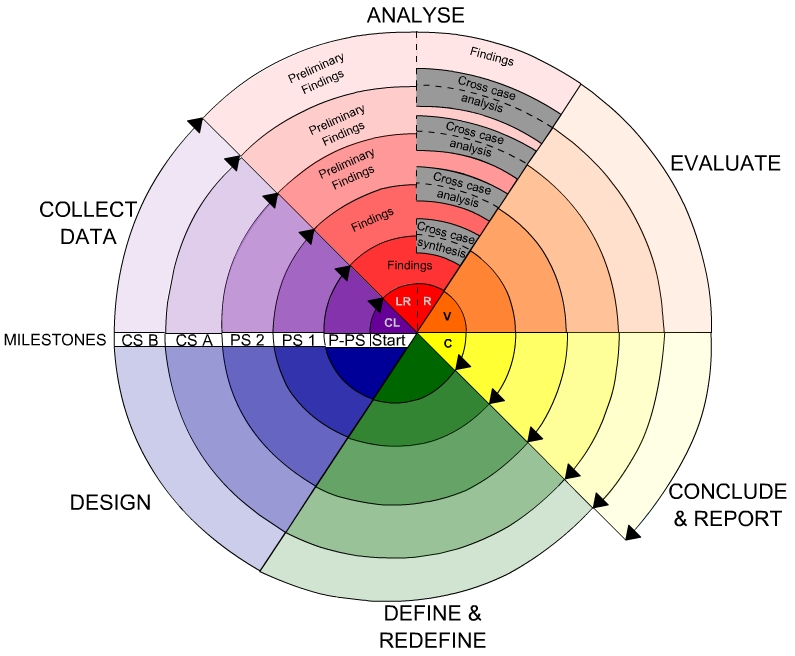
**3.2 STUDY OF THE SYSTEM**

In the flexibility of uses the interface has been developed a graphics concepts in mind, associated through a browser interface.  The GUI’s at the top level has been categorized as follows

1. Administrative User Interface Design
2. The Operational and Generic User Interface Design

The administrative user interface concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection.   The Interface helps the administration with all the transactional states like data insertion, data deletion, and data updating along with executive data search capabilities.

The operational and generic user interface helps the users upon the system in transactions through the existing data and required services.  The operational user interface also helps the ordinary users in managing their own information helps the ordinary users in managing their own information in a customized manner as per the assisted flexibilities.



**3.3. Fundamental Concepts on (Domain):**

What is Cloudera?

Cloudera is revolutionizing enterprise data management by offering the first unified Platform for Big Data: The Enterprise Data Hub. Cloudera offers enterprises one place to store, process, and analyze all their data, empowering them to extend the value of existing investments while enabling fundamental new ways to derive value from their data.

Why do customers choose Cloudera?

Cloudera was the first commercial provider of python-related software and services and has the most customers with enterprise requirements, and the most experience supporting them, in the industry. Cloudera’s combined offering of differentiated software (open and closed source), support, training, professional services, and indemnity brings customers the greatest business value, in the shortest amount of time, at the lowest TCO.

**Data Mining**

There is a huge amount of data available in the Information Industry. This data is of no use until it is converted into useful information. It is necessary to analyze this huge amount of data and extract useful information from it. Extraction of information is not the only process we need to perform; data mining also involves other processes such as Data Cleaning, Data Integration, Data Transformation, Data Mining, Pattern Evaluation and Data Presentation. Once all these processes are over, we would be able to use this information in many applications such as Fraud Detection, Market Analysis, Production Control, Science Exploration, etc.

What is Data Mining?

Data Mining is defined as extracting information from huge sets of data. In other words, we can say that data mining is the procedure of mining knowledge from data. The information or knowledge extracted so can be used for any of the following applications:

 Market Analysis

 Fraud Detection

 Customer Retention

 Production Control

 Science Exploration

Data Mining Applications

Data mining is highly useful in the following domains:  Market Analysis and Management

  Corporate Analysis & Risk Management  Fraud Detection

**DATA MINING ON WHAT KIND OF DATA?**

**DATA WAREHOUSES**

A data warehouse is a repository of information collected from multiple sources, stored under a unified schema, and that usually resides at a single site Data warehouses are constructed via a process of data cleaning, data integration, data transformation, data loading, and periodic data refreshing

**OBJECT-RELATIONAL DATABASES**

Based on an object-relational data model Extends the relational model by providing a rich data type for handling complex objects and object orientation Objects that share a common set of properties can be grouped into an object class. Each object is an instance of its class. Object classes can be organized into class/subclass hierarchies

**ADVANCED DATA AND INFORMATION SYSTEMS**

With the progress of database technology, various kinds of advanced data and information systems have emerged and are undergoing development to address the requirements of new applications handling spatial/temporal data (such as maps) engineering design data (such as the design of buildings, system components, or integrated circuits) hypertext and multimedia data (including text, image, video, and audio data) time-related data (such as historical records or stock exchange data) stream data (such as video surveillance and sensor data, where data flow in and out like streams) the World Wide Web (a huge, widely distributed information repository made available by the Internet)

**THE WORLD WIDE WEB:**

The World Wide Web and its associated distributed information services, such as Yahoo! and Google provide rich, worldwide, on-line information services, where data objects are linked together to facilitate interactive access Capturing user access patterns in such distributed information environments is called Web usage mining (or Weblog mining)

Database or data warehouse server responsible for fetching the relevant data, based on the user’s data mining request can be decouples/loose coupled/tightly coupled with the database layer

Knowledge base the domain knowledge that is used to guide the search or evaluate the interestingness of resulting patterns interestingness constraints or thresholds, metadata, concept hierarchies, etc.

Data mining engine this is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association and correlation analysis, classification, prediction, cluster analysis, outlier analysis, and evolution analysis query languages (DMQL) based on mining primitives to access the data

Pattern evaluation module interacts with the data mining modules so as to focus the search toward interesting patterns may use interestingness thresholds to filter out discovered patterns may be integrated with the mining module.

**3.4.  SYSTEM ANALYSIS:**

The **Systems Development Life Cycle (SDLC)**, or *Software Development Life Cycle* in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems.

In software engineering the SDLC concept underpins many kinds of [software development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies). These methodologies form the framework for planning and controlling the creation of an information system the [software development process](http://en.wikipedia.org/wiki/Software_development_process).

**SOFTWARE MODEL OR ARCHITECTURE ANALYSIS:**

                        Structured project management techniques (such as an SDLC) enhance management’s control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. The ―one size fits all‖ approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to address the above mentioned defects by using a new hypothetical model for SDLC described elsewhere. The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

**3.4.Functional requirements:**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization,.
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.
* Understanding user’s preferences, expertise level and his business requirements through a friendly questionnaire.
* Input data can be in four different forms - Relational DB, text files, .xls and xml files. For testing and demo you can choose data from any domain. User-B can provide business data as input.

**3.5.EXISTING SYSTEM:**

**K-Mean Clustering:**

Clustering is unsupervised Learning Algorithm and the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

**DISADVANTAGES:**

1. Choosing k manually

Use the “Loss vs. Clusters” plot to find the optimal (k)

1. Being dependent on initial values.

For a low k, you can mitigate this dependence by running k-means several times with different initial values and picking the best result. As k increases, you need advanced versions of k-means to pick better values of the initial centroids (called k-means seeding).

1. Clustering data of varying sizes and density.

k-means has trouble clustering data where clusters are of varying sizes and density.

1. Clustering outliers.

Centroids can be dragged by outliers, or outliers might get their own cluster instead of being ignored. Consider removing or clipping outliers before clustering.

1. Scaling with number of dimensions.

As the number of dimensions increases, a distance-based similarity measure converges to a constant value between any given examples. Reduce dimensionality either by using [PCA](https://wikipedia.org/wiki/Principal_component_analysis) on the feature data.

**3.6.PROPOSED SYSTEM:**

**Recommender System:**

1. Content-based filtering technique
2. Collaborative filtering  technique

c. Model-based filtering technique

d. Memory-based filtering technique

1.User-based  technique

2.Item-based technique

e. Hybrid filtering technique

**Content-based filtering:**

Content-based technique is a domain-dependent algorithm and it emphasizes more on the analysis of the attributes of items in order to generate predictions. When documents such as web pages, publications and news are to be recommended,

Content-based filtering technique is the most successful. In content-based filtering technique, recommendation is made based on the user profiles using features extracted from the content of the items the user has evaluated in the past.

**Collaborative filtering**

Collaborative filtering is a domain-independent prediction technique for content that cannot easily and adequately be described by metadata such as movies and music. Collaborative filtering technique works by building a database (user-item matrix) of preferences for items by users. It then matches users with relevant interest and preferences by calculating similarities between their profiles to make recommendations.

**SUPPORT VECTOR MACHINES**

“Support Vector Machine” (SVM) is a supervised [machine learning algorithm](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2?utm_source=blog&utm_medium=understandingsupportvectormachinearticle) which can be used for both classification or regression challenges. However,  it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well

**Advantages:**

1. SVM works relatively well when there is clear margin of separation between classes.
2. SVM is more effective in high dimensional spaces.
3. SVM is effective in cases where number of dimensions is greater than the number of samples.
4. SVM is relatively memory efficient

**4.HARDWARE & SOFTWARE REQUIREMENTS:**

Hardware Requirements:

Processor : Pentium-III (or) Higher

Ram : 64MB (or) Higher

Cache : 512MB

Hard disk : 10GB

Software Requirements:

Technology : Java 2 Standard Edition, JDBC

WebServer : Tomcat 6.0

Client Side Technologies : HTML, CSS, JavaScript

Server Side Technologies : Servlets, JSP

Data Base Server : MySQL

Operating System : Microsoft Windows, Linux or Mac.

**5.SOFTWARE REQUIREMENT SPECIFICATIONS**

## 5.1. FUNCTIONAL REQUIREMENTS

**OUTPUT DESIGN:**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**OUTPUT DEFINITION**

# The outputs should be defined in terms of the following points:

* + - Type of the output
    - Content of the output
    - Format of the output
    - Location of the output
    - Frequency of the output
    - Volume of the output
    - Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**INPUT DESIGN**

Input design is a part of overall system design.  The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT STAGES:**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**INPUT TYPES:**

It is necessary to determine the various types of inputs.  Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**INPUT MEDIA:**

At this stage choice has to be made about the input media.  To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive.  As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**ERROR AVOIDANCE**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system.  This can be achieved only by means of careful control each time the data is handled.

**ERROR DETECTION**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**DATA VALIDATION**

Procedures are designed to detect errors in data at a lower level of detail.  Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors.  The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct.  Validations have been included where necessary.

**6. SELECTED SOFTWARE**

**6.1. INTRODUCTION**

**IMPLIMENTATION ON (PYTHON)**

What Is A Script?

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

**Scripts are reusable**

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

**Scripts are editable**

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

**You will need a text editor**

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

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

**Difference between a script and  a program**

Script:

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

Program:

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

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

**5.2Python**

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

Python concepts

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

• Open source general-purpose language.

 • Object Oriented, Procedural, Functional

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

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

• Great interactive environment

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

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

**History of Python**

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

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

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

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

**Python Features**

Python's features include −

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

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

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

**Dynamic vs Static**

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

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

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

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

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

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

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

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

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

**Variables**

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

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

**Standard Data Types**

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

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

Python Numbers

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

## Python Strings

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

## Python Lists

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

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

## Python Tuples

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

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

## Python Dictionary

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

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

**Different modes in python**

Python has two basic modes: normal and interactive.

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

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

# 20 Python libraries

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

**2.** Scrapy. If you are involved in webscraping then this is a must have library for you. After using this library you won’t use any other.

**3.** wxPython. A gui toolkit for python. I have primarily used it in place of tkinter. You will really love it.

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

**5.** SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

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

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

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

**9.** SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

**10.** matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

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

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

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

**14.** pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.

**15.** Scapy. A packet sniffer and analyzer for python made in python.

**16.** pywin32. A python library which provides some useful methods and classes for interacting with windows.

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

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

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

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

**Numpy**

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

**• Offers Matlab**-ish capabilities within Python

 • Fast array operations

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

**matplotlib**

• High quality plotting library.

**Python modules**

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

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

**Testing code**

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

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

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

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

he entire process is known as the development cycle.

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

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

## Functions in Python

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

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

## To carry out that specific task, the function might or might not need multiple inputs. When the task is carred out, the function can or can not return one or more values.There are three types of functions in python:

## help() ,min() ,print().

## Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e. the naming of people in firstname and familiy name (surname).

 An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames.   
Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

 This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

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

Some namespaces in Python:

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

**Garbage Collection**

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

**Python-Data Base Communication**

Connector/Python provides a connect() call used to establish connections to the MySQL server. The following sections describe the permitted arguments for connect() and describe how to use option files that supply additional arguments.

A database is an organized collection of data. The data are typically organized to model aspects of reality in a way that supports processes requiring this information.

The term "database" can both refer to the data themselves or to the database management system. The Database management system is a software application for the interaction between users database itself.

Databases are popular for many applications, especially for use with web applications or customer-oriented programs

Users don't have to be human users. They can be other programs and applications as well. We will learn how Python or better a Python program can interact as a user of an SQLdatabase.   
  
This is an introduction into using SQLite and MySQL from Python.

The Python standard for database interfaces is the Python DB-API, which is used by Python's database interfaces.

The DB-API has been defined as a common interface, which can be used to access relational databases.

 In other words, the code in Python for communicating with a database should be the same, regardless of the database and the database module used. Even though we use lots of SQL examples, this is not an introduction into SQL but a tutorial on the Python interface.

SQLite is a simple relational database system, which saves its data in regular data files or even in the internal memory of the computer, i.e. the RAM.

It was developped for embedded applications, like Mozilla-Firefox (Bookmarks), Symbian OS or Android.

 SQLITE is "quite" fast, even though it uses a simple file. It can be used for large databases as well.

If you want to use SQLite, you have to import the module sqlite3. To use a database, you have to create first a Connection object.

The connection object will represent the database. The argument of connection - in the following example "companys.db" - functions both as the name of the file, where the data will be stored, and as the name of the database. If a file with this name exists, it will be opened.

 It has to be a SQLite database file of course! In the following example, we will open a database called company.

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library.

For notes detailing the changes in each release of Connector/Python, see MySQL Connector/Python Release Notes.

MySQL Connector/Python includes support for:

* Almost all features provided by MySQL Server up to and including MySQL Server version 5.7.
* Converting parameter values back and forth between Python and MySQL data types, for example Python datetime and  MySQL DATETIME. You can turn automatic conversion on for convenience, or off for optimal performance.
* All MySQL extensions to standard SQL syntax.
* Protocol compression, which enables compressing the data stream between the client and server.
* Connections using TCP/IP sockets and on Unix using Unix sockets.
* Secure TCP/IP connections using SSL.
* Self-contained driver. Connector/Python does not require the MySQL client library or any Python modules outside the standard library

**5.3jupyter notebook:**

* The **Jupyter** Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning

**7.SYSTEM DESIGN**

**7.1. INTRODUCTION**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. The purpose of the design phase is to plan a solution of the problem specified by the requirement document.  This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs.  The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document.  This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

            System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results.  At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During, Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase, the details of the data of a module is usually specified in a high-level  design description language, which is independent of the target language in which the software will eventually be implemented.

  In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules.  In other works, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue.

  Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.

During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.

Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

**Data Flow Diagrams:**

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed.  The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts.  The Basic Notation used to create a DFD’s are as follows:

**1. Dataflow:** Data move in a specific direction from an origin to a    destination.

**2.  Process:** People, procedures, or devices that use or produce (Transform) Data.  The physical component is not identified.

**3. Source:** External sources or destination of data, which may be People, programs, organizations or other entities.

**4. Data Store:** Here data are stored or referenced by a process in the System.

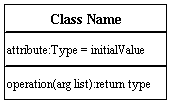
**What is a UML Class Diagram?**

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

**Basic Class Diagram Symbols and Notations**

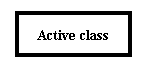
Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and write operations into the third.



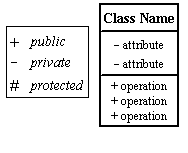
**Active Class**

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.



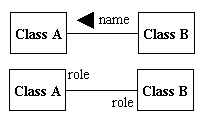
**Visibility**

Use visibility markers to signify who can access the information contained within a class. 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. [.](http://www.smartdraw.com/resources/tutorials/Text-and-Tables)



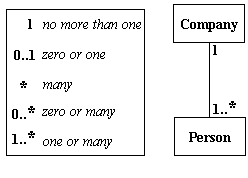
**Associations**

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.  
***Note:*** It's uncommon to name both the association and the class roles.



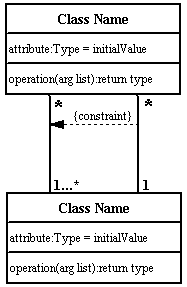
**Multiplicity (Cardinality)**

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.



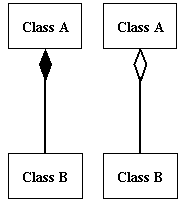
**Constraint**

Place constraints inside curly braces {}.

http://wc1.smartdraw.com/resources/tutorials/images/uml_constraint.gif*Simple Constraint* 

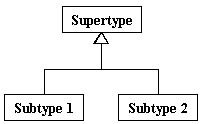
**Composition and Aggregation**

Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustrate **composition** with a filled diamond. Use a hollow diamond to represent a simple **aggregation** relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate



**Generalization**

Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.



In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the super class.

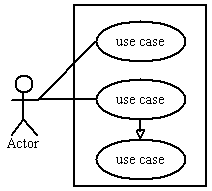
**What is a UML Use Case Diagram?**

Use case diagrams model the functionality of a system using actors and use cases. Use cases are services or functions provided by the system to its users.

**Basic Use Case Diagram Symbols and Notations**

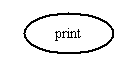
**System**

Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



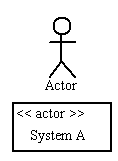
**Use Case**

Draw use cases using ovals. Label with ovals with verbs that represent the system's functions.



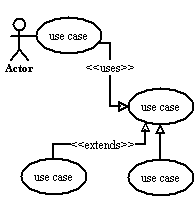
**Actors**

Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



**Relationships**

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.



**Sequence Diagram**

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

**Basic Sequence Diagram Symbols and Notations**

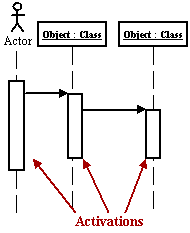
**Class roles**

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Class roles

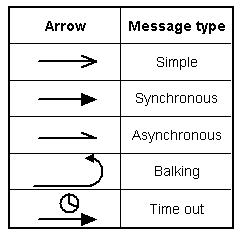
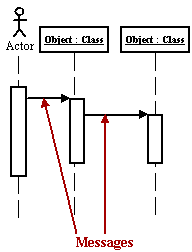
**Activation**

Activation boxes represent the time an object needs to complete a task.



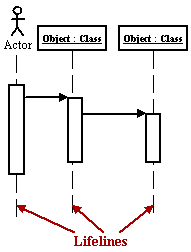
**Messages**

Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

  
*Various message types for Sequence and Collaboration diagrams*

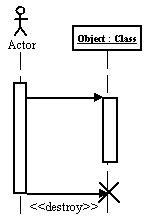
**Lifelines**

Lifelines are vertical dashed lines that indicate the object's presence over time.



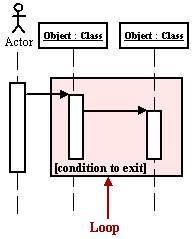
**Destroying Objects**

Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X.



**Loops**

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].



**Collaboration Diagram**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

**Basic Collaboration Diagram Symbols and Notations**

**Class roles**

Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.

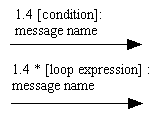
Class roles

**Association roles**

Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.  
Association roles

**Messages**

Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution. Sequence numbering can become nested using the Dewey decimal system. For example, nested messages under the first message are labeled 1.1, 1.2, 1.3, and so on. The a condition for a message is usually placed in square brackets immediately following the sequence number. Use a \* after the sequence number to indicate a loop.  
[Learn how to add arrows to your lines.](http://www.smartdraw.com/resources/tutorials/Lines)



**Activity Diagram**

An activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation. Because an activity diagram is a special kind of state chart diagram, it uses some of the same modeling conventions.

**Basic Activity Diagram Symbols and Notations**

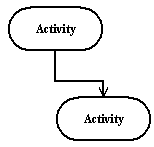
**Action states**

Action states represent the non interruptible actions of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.



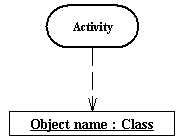
**Action Flow**

Action flow arrows illustrate the relationships among action states.



**Object Flow**

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



**Initial State**

A filled circle followed by an arrow represents the initial action state.

Initial State

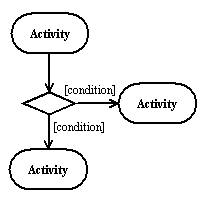
**Final State**

An arrow pointing to a filled circle nested inside another circle represents the final action state.

Final State

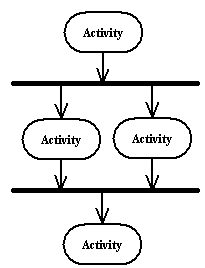
**Branching**

A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."



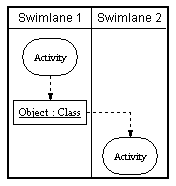
**Synchronization**

A synchronization bar helps illustrate parallel transitions. Synchronization is also called forking and joining.



**Swimlanes**

Swimlanes group related activities into one column.

**State chart Diagram**

A state chart diagram shows the behavior of classes in response to external stimuli. This diagram models the dynamic flow of control from state to state within a system.

**Basic State chart Diagram Symbols and Notations**

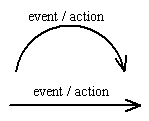
**States**

States represent situations during the life of an object. You can easily illustrate a state in Smart Draw by using a rectangle with rounded corners.



**Transition**

A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.



**Initial State**

A filled circle followed by an arrow represents the object's initial state.

Initial State

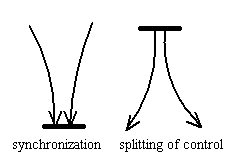
**Final State**

An arrow pointing to a filled circle nested inside another circle represents the object's final state.

Final State

**Synchronization and Splitting of Control**

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.



**STATE CHART DIAGRAM:**

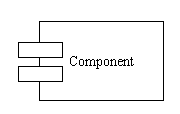
**What is a UML Component Diagram?**

A component diagram describes the organization of the physical components in a system.

**Basic Component Diagram Symbols and Notations**

**Component**

A component is a physical building block of the system. It is represented as a rectangle with tabs.  
[Learn how to resize grouped objects like components.](http://www.smartdraw.com/resources/tutorials/Objects)



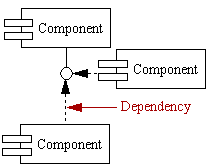
**Interface**

An interface describes a group of operations used or created by components.

Interface

**Dependencies**

Draw dependencies among components using dashed arrows.  
[Learn about line styles in SmartDraw.](http://www.smartdraw.com/resources/tutorials/Lines)



**COMPONENT DIAGRAM:**

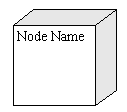
**What is a UML Deployment Diagram?**

Deployment diagrams depict the physical resources in a system including nodes, components, and connections.

**Basic Deployment Diagram Symbols and Notations**

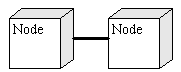
**Component**

A node is a physical resource that executes code components.  
[Learn how to resize grouped objects like nodes.](http://www.smartdraw.com/resources/tutorials/Objects)



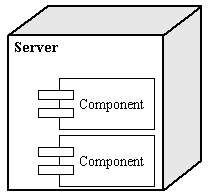
**Association**

Association refers to a physical connection between nodes, such as Ethernet.  
[Learn how to connect two nodes.](http://www.smartdraw.com/resources/tutorials/Lines)



**Components and Nodes**

Place components inside the node that deploys them.

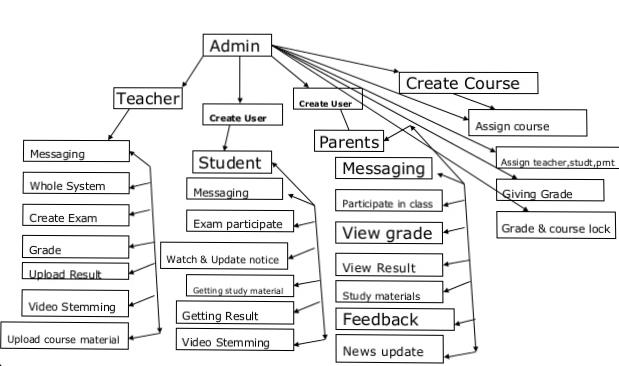


**7.2. Data Flow Diagram / Flow Diagram**

**Data Flow Diagram / Use Case Diagram / Flow Diagram**

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

**Data Flow Diagram:**

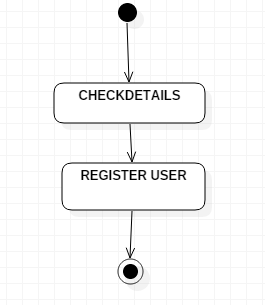


6.5. **ACTIVITY DIAGRAM**

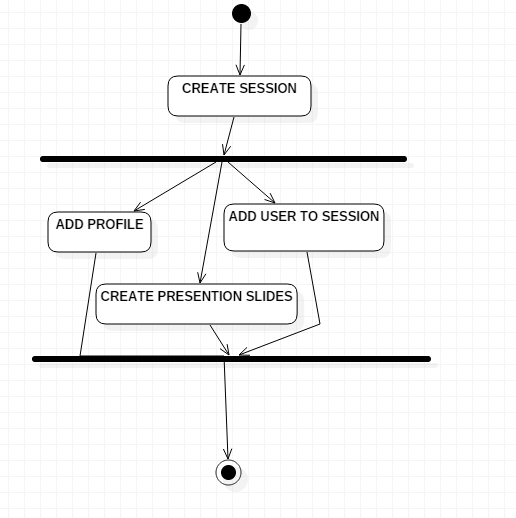
An activity diagram is characterized by states that denote various operations. Transition from one state to the other is triggered by completion of the operation. The purpose of an activity is symbolized by round box, comprising the name of the operation. An operation symbol indicates the execution of that operation. This activity diagram depicts the internal state of an object.

**Fig.** Activity Diagram

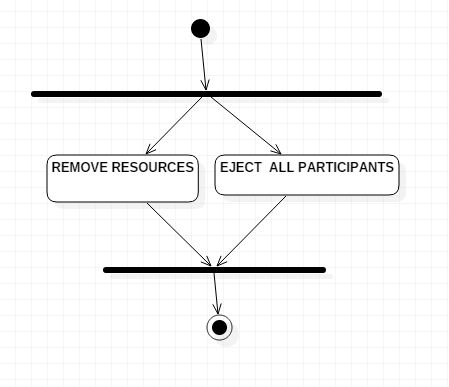
**FOR REGISTER:**

****

**FORCREATE PRESENTATION:**

****

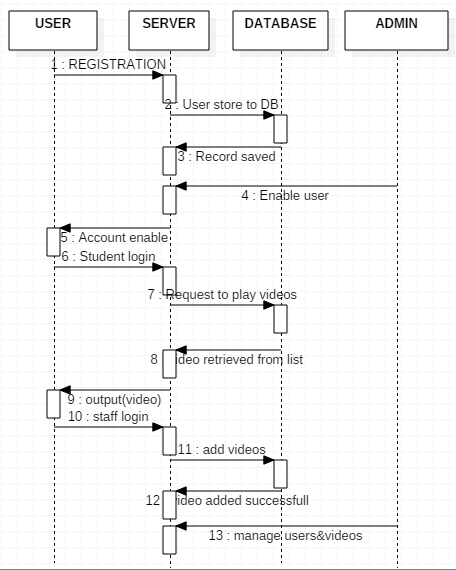
**FOR END PRESENTATION:**

****

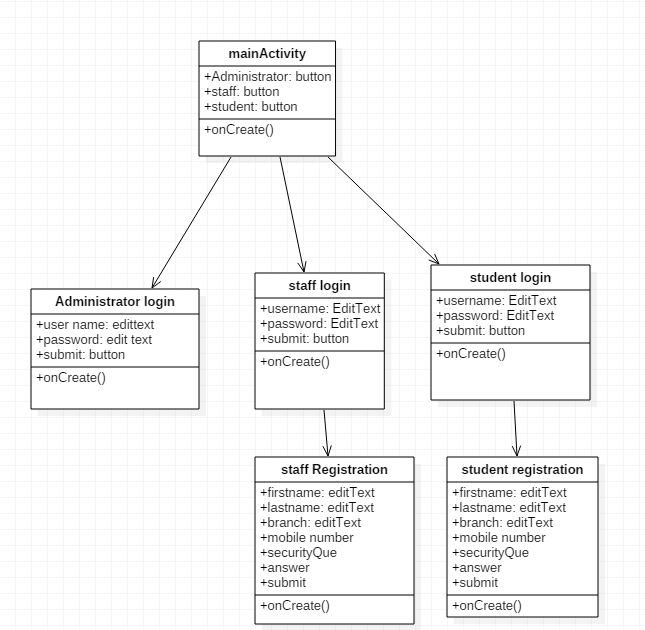
**UML SEQUENCE DIAGRAM**

The sequence diagrams are an easy and intuitive way of describing thesystem’s behavior, which focuses on the interaction between the system and the environment. This notational diagram shows the interaction arranged in a time sequence. The sequence diagram has two dimensions: the vertical dimension represents the time, the horizontal dimension represents different objects. The vertical line also called the object’s *lifeline* represents the object’s existence during the interaction.

**Fig.6.6.**Sequence Diagram



**6.7. CLASS DIAGRAM:**



6.8.**USE CASE DIAGRAM:**

CloudServer



database



validation



showres



invalid



showfake



pass



wrtefile



dbconnection()



validation()



showfiles()



download()



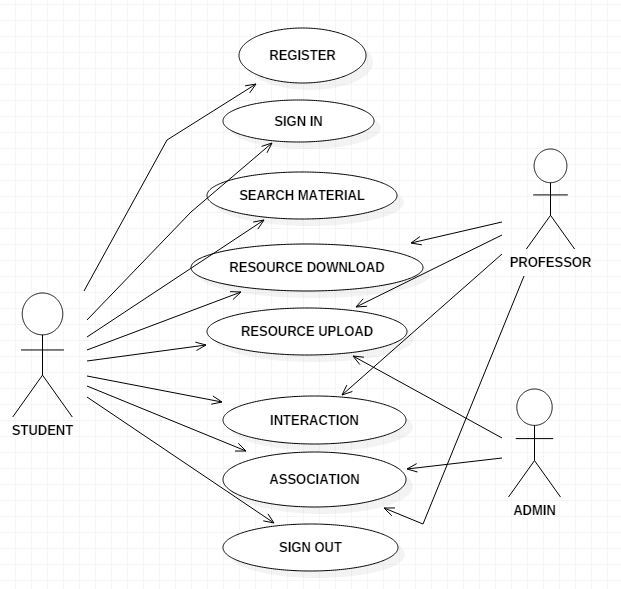
invalid()



showfake()



wrtefile()

****

**SOFTWARE ENVIRONMENT:**

**Java code:**

importjava.io.File;

importjava.io.FileInputStream;

importjava.io.IOException;

importjava.io.PrintWriter;

importjava.sql.Connection;

importjava.sql.DriverManager;

importjava.sql.PreparedStatement;

importjava.util.Hashtable;

importjavax.servlet.ServletException;

importjavax.servlet.http.HttpServlet;

importjavax.servlet.http.HttpServletRequest;

importjavax.servlet.http.HttpServletResponse;

importjavazoom.upload.MultipartFormDataRequest;

importjavazoom.upload.UploadBean;

importorg.apache.poi.hssf.usermodel.HSSFRow;

importorg.apache.poi.hssf.usermodel.HSSFSheet;

importorg.apache.poi.hssf.usermodel.HSSFWorkbook;

importorg.apache.poi.poifs.filesystem.POIFSFileSystem;

public class MarksUpload extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throwsServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

try {

MultipartFormDataRequestmrequest=null;

String filetoUpload="";

String filetype="";

String pname="";

String mfd="";

String brand="";

String discount="";

String cid="";

String mrp="";

UploadBeanupBean=new UploadBean();

if (MultipartFormDataRequest.isMultipartFormData(request)){

mrequest = new MultipartFormDataRequest(request);

Hashtable files = mrequest.getFiles();

filetype=mrequest.getParameter("imagefile");

filetoUpload=mrequest.getParameter("fname");

File ftu=new File(filetoUpload);

String fileName=ftu.getName();

System.out.println(fileName+"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

String appPath = getServletConfig().getServletContext().getRealPath("");

// constructs path of the directory to save uploaded file

String savePath = appPath + File.separator ;

String finalpath = savePath+ "sheets\\"+fileName;

upBean.setFolderstore(finalpath);

// System.out.println("\*\*\*\*\*\*\*\*\*\*9999999999999999"+finalpath);

upBean.store(mrequest,"imagefile");

//System.out.println("777777777777success");

String zzz = finalpath+"\\"+fileName;

System.out.println(zzz);

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/virtualclass","root","");

con.setAutoCommit(false);

PreparedStatementpstm = null ;

FileInputStream input = new FileInputStream(zzz);

POIFSFileSystemfs = new POIFSFileSystem( input );

HSSFWorkbookwb = new HSSFWorkbook(fs);

HSSFSheet sheet = wb.getSheetAt(0);

HSSFRow row;

for(int i1=1; i1<=sheet.getLastRowNum(); i1++){

row = sheet.getRow(i1);

String htno = row.getCell(0).getStringCellValue();

// System.out.println(htno);

String branch = row.getCell(1).getStringCellValue();

//System.out.println(branch);

String year = row.getCell(2).getStringCellValue();

//System.out.println(year);

String subname = row.getCell(3).getStringCellValue();

//System.out.println(subname);

int internal = (int) row.getCell(4).getNumericCellValue();

//System.out.println(internal);

int external = (int) row.getCell(5).getNumericCellValue();

//System.out.println(external);

double total = (int) row.getCell(6).getNumericCellValue();

// System.out.println(total);

String acyear = row.getCell(7).getStringCellValue();

//System.out.println(acyear);

pstm = con.prepareStatement("INSERT INTO marks VALUES(?,?,?,?,?,?,?,?)");

pstm.setString(1, htno);

pstm.setString(2, branch);

pstm.setString(3, year);

pstm.setString(4, subname);

pstm.setInt(5, internal);

pstm.setInt(6, external);

pstm.setDouble(7, total);

pstm.setString(8, acyear);

pstm.executeUpdate();

}

//System.out.println("################");

response.sendRedirect("adminhome.jsp");

}

} catch(Exception e){

e.printStackTrace();

out.close();

}

}

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throwsServletException, IOException {

processRequest(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throwsServletException, IOException {

processRequest(request, response);

}

}

7.2. JS Code:

Login Page code:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@include file="DBConnection.jsp"%>

<!DOCTYPEhtml PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<metahttp-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<%

String utype = request.getParameter("utype");

String uname = request.getParameter("uname");

String password = request.getParameter("password");

if (utype.equals("Student"))

{

ps=con.prepareStatement("select \* from signup where uname=? and password=? andutype=?");

ps.setString(1,uname);

ps.setString(2,password);

ps.setString(3,utype);

rs=ps.executeQuery();

while(rs.next())

{

String uname1 = rs.getString("uname");

String password1 = rs.getString("password");

String utype1 = rs.getString("utype");

if(uname.equals(uname1) &&password.equals(password1) &&utype.equals(utype1))

{

session.setAttribute("user",uname);

Cookie c=new Cookie("loginname",uname);

response.addCookie(c);

PreparedStatement pst1=con.prepareStatement("insert into cookies values (?)");

pst1.setString(1,uname);

pst1.executeUpdate();

response.sendRedirect("studenthome.jsp");

}

else {

response.sendRedirect("login.html");

}

}

}

else if (utype.equals("Staff"))

{

ps=con.prepareStatement("select \* from signup where uname=? and password=? andutype=?");

ps.setString(1, uname);

ps.setString(2, password);

ps.setString(3,utype);

rs = ps.executeQuery();

while(rs.next())

{

String uname1 = rs.getString("uname");

String password1 = rs.getString("password");

String utype1 = rs.getString("utype");

if(uname.equals(uname1) &&password.equals(password1) &&utype.equals(utype1))

{

session.setAttribute("user",uname);

Cookie c=new Cookie("loginname",uname);

response.addCookie(c);

PreparedStatement pst1=con.prepareStatement("insert into cookies values (?)");

pst1.setString(1,uname);

pst1.executeUpdate();

response.sendRedirect("staffhome.jsp");

}

else {

response.sendRedirect("login.html");

}

}

}

else if(utype.equalsIgnoreCase("Admin"))

{

if(uname.equalsIgnoreCase("admin") &&password.equalsIgnoreCase("admin"))

{

session.setAttribute("user",uname);

response.sendRedirect("adminhome.jsp");

}

}

else

{

System.out.println("Invalid User");

}

%>

</body>

</html>

**Registration page :**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@include file="DBConnection.jsp"%>

<!DOCTYPEhtml PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<metahttp-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<%

String utype=request.getParameter("utype");

String login=request.getParameter("uname");

String password=request.getParameter("password");

String education=request.getParameter("education");

String email=request.getParameter("email");

String mobile=request.getParameter("mobile");

String gender=request.getParameter("gender");

String dd=request.getParameter("dob");

String address=request.getParameter("addr");

ps=con.prepareStatement("insert into signup values(?,?,?,?,?,?,?,?,?)");

ps.setString(1, login);

ps.setString(2, password);

ps.setString(3, education);

ps.setString(4, email);

ps.setString(5, mobile);

ps.setString(6, gender);

ps.setString(7, dd);

ps.setString(8, address);

ps.setString(9, utype);

int i = ps.executeUpdate();

if(i>0)

{

response.sendRedirect("login.html");

}

else

{

response.sendRedirect("register.html");

}

%>

</body>

</html>

**Staff home page:**

<!DOCTYPEhtml>

<html lang="en">

<head>

<title>Virtual Class</title>

<metacharset="utf-8">

<link rel="stylesheet" href="css/reset.css" type="text/css" media="all">

<link rel="stylesheet" href="css/layout.css" type="text/css" media="all">

<link rel="stylesheet" href="css/style.css" type="text/css" media="all">

<script type="text/javascript" src="js/jquery-1.5.2.js" ></script>

<script type="text/javascript" src="js/cufon-yui.js"></script>

<script type="text/javascript" src="js/cufon-replace.js"></script>

<script type="text/javascript" src="js/Molengo\_400.font.js"></script>

<script type="text/javascript" src="js/Expletus\_Sans\_400.font.js"></script>

<!--[if lt IE 9]>

<script type="text/javascript" src="js/html5.js"></script>

<style type="text/css">.bg, .box2{behavior:url("js/PIE.htc");}</style>

<![endif]-->

</head>

<body id="page1">

<div class="body1">

<div class="main">

<!-- header -->

<header>

<div class="wrapper">

<nav>

<ul id="menu">

<li><a href="staffhome.jsp">Home</a></li>

<li><a href="Vcrclass\_staff.jsp">Class Room</a></li>

<!--<li><a href="ChatRoom.jsp">Chat Room</a></li> -->

<!--<li><a href="exampaper.jsp">Examination</a></li> -->

<li><a href="staff\_viewstaff.jsp">View Staff Details</a></li>

<li><a href="viewqueries.jsp">View Doubts</a></li>

<li><a href="staff\_viewnews.jsp">News</a></li>

<li><a href="staff\_about.jsp">About Us</a></li>

<li><a href="staff\_contact.jsp">Contact Us</a></li>

<li class="end"><a href="logout.jsp">Sign Out</a></li>

</ul>

</nav>

<ul id="icon">

<li><a href="#"><imgsrc="images/icon1.jpg" alt=""></a></li>

<li><a href="#"><imgsrc="images/icon2.jpg" alt=""></a></li>

<li><a href="#"><imgsrc="images/icon3.jpg" alt=""></a></li>

</ul>

</div>

<h3>Staff Home Page</h3>

<br/>

<%

String name = (String)session.getAttribute("user");

session.setAttribute("user",name);

%>

Welcome to <%=name %>

</header>

<!-- / header -->

</div>

</div>

<div class="body2">

<div class="main">

<!-- content -->

<section id="content">

<div class="wrapper">

<div class="pad1 pad\_top1">

<article class="cols marg\_right1">

<figure><a href="#"><imgsrc="images/page1\_img1.jpg" alt=""></a></figure>

<span class="font1">Our Mission Statement</span></article>

<article class="cols marg\_right1">

<figure><a href="#"><imgsrc="images/page1\_img2.jpg" alt=""></a></figure>

<span class="font1">Performance Report</span></article>

<article class="cols">

<figure><a href="#"><imgsrc="images/page1\_img3.jpg" alt=""></a></figure>

<span class="font1">Prospective Parents</span></article>

</div>

</div>

</section>

<!-- content -->

</div>

</div>

<script type="text/javascript">Cufon.now();</script>

</body>

</html>

**8. SYSTEM TESTING**

8.1. INTRODUCTION

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

8.2. Unit Testing:

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

**Test objectives**

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

**Features to be tested**

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

.

8.3.Integration testing:

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

**Functional test**

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

Functional testing is centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

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

**System Test**

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

**White Box Testing**

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

**Black Box Testing**

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

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

8.4. Acceptance Testing

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

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

**9.Output Screens:**





















**10. CONCLUSION:**

The virtual classroom transcends the boundaries of location, time and space providing a flexible learning environment for all. Schools, universities and corporate organizations benefit form Virtual classrooms because it provides an excellent way for experts to teach a geographically dispersed group of students without hassle.

**11. BIBLIOGRAPHY**

Advanced Java Programming - Dietel and Dietel

Mastering JAVA 2 - John Zukowski

Java Server Programming - Apress

Software Engineering - Roger S Pressman

Análysis&Design of InformationSystems – Senn

**REFERENCES :**

[1] J. McAuley and A. Yang, “Addressing complex and subjective product-related queries with customer reviews,” in WWW, 2016, pp. 625–635.

[2] N. V. Nielsen, “E-commerce: Evolution or revolution in the fastmoving

consumer goods world,” nngroup. com, 2014.

[3] W. D. J. Salganik M J, Dodds P S, “Experimental study of inequality

and unpredictability in an artificial cultural market,” in

ASONAM, 2016, pp. 529–532.

[4] R. Peres, E. Muller, and V. Mahajan, “Innovation diffusion and

new product growth models: A critical review and research directions,”

International Journal of Research in Marketing, vol. 27, no. 2,

pp. 91 – 106, 2010.

[5] L. A. Fourt and J. W. Woodlock, “Early prediction of market

success for new grocery products.” Journal of Marketing, vol. 25,

no. 2, pp. 31 – 38, 1960.

[6] B. W. O, “Reference group influence on product and brand purchase

decisions,” Journal of Consumer Research, vol. 9, pp. 183–194,

1982.

[7] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, “Imagebased

recommendations on styles and substitutes,” in SIGIR, 2015,

pp. 43–52.

[8] E. M.Rogers, Diffusion of Innovations. New York: The Rise of High-

Technology Culture, 1983.

[9] K. Sarkar and H. Sundaram, “How do we find early adopters

who will guide a resource constrained network towards a desired

distribution of behaviors?” in CoRR, 2013, p. 1303.

[10] D. Imamori and K. Tajima, “Predicting popularity of twitter accounts

through the discovery of link-propagating early adopters,”

in CoRR, 2015, p. 1512.