

Sathyabama Institute of Science and Technology (Deemed to be University)

**Submitted in partial fulfillment of the requirements for the award of Bachelor
of Engineering Degree in Computer Science and Engineering**

By

**ALLA ROHITHNAIDU
REG. NO: 38110022**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU,
March 2022.**



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI– 600119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Alla Rohithnaidu (38110022)** who carried out the project entitled “**ONLINE SUBJECTIVE ANSWER CHECKER**”
” under my supervision from December 2021 to March 2022.

Internal Guide

Dr. P. Sardar Maran. ME. Ph. D

Head of the Department

Dr. L. Lakshmanan, M.E., Ph.D.
Dr. S. Vigneshwari, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **ALLA ROHITHNAIDU** hereby declare that the project report entitled **ONLINE SUBJECTIVE ANSWER CHECKER** done by me under the guidance of **Dr. P. SARDAR MARAN .ME. Ph. D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean, School of Computing, Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Dr. P. SARDAR MARAN .ME. Ph.D., for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Examination is a test of a person's knowledge in a particular area which is either subjective or objective or both. Usually, competitive examinations consist of multiple-choice questions or mcqs. Automatic evaluation of the objective exams is beneficial as it saves time, provides efficiency, reduces usage of resources. However, this automated evaluation technique is only for the objective exams and not for the subjective ones. Subjective answer sheet checking is one of the huge administrative tasks for any education institute. In this examination process, candidates need to write answers, an examiner collects those answer sheets and submits them to authority for further checking process. This process involves 3 levels of paper checking: -

- First Level Paper Checker,
- First Level Moderation,
- Second Level Moderation.

So, the amount of pressure education systems and teachers hold is understandable as the number of answer sheets to evaluate is too large. So, there is a necessity for an approach which will reduce the usage of resources by providing an approach which will automatically evaluate the answers given by students and provide results. Such a system is the goal of this paper. We have developed an E-assessment system that checks the answer sheet of the student and provides marks to the same. The system consists of an algorithm that compares the student's answer against three reference answers given by three different faculties and the answer with most close results and with highest precision is taken into consideration and marks are allocated accordingly. Both the answers need not be exactly the same or word to word. This approach can be a quick and easy way for the examiners by reducing their workload. An automatic answer checker application that checks and marks written answers similar to a human being. This software application is built to check subjective answers in an online examination and allocate marks to the user after verifying the answer. The system requires you to store the original answer for the system. This facility is provided to the admin. The admin may insert questions and respective subjective answers in the system. These answers are stored as csv files. When a user takes the test he is provided with questions and area to type his answers. Once the user enters his/her answers the system then compares this answer to original answer written in database and allocates marks accordingly. Both the answers need not be exactly same word to word. The system consists if in build artificial intelligence sensors that verify answers and allocate marks accordingly as good as a human being .

Examinations are of two types-descriptive or objective. Every examination need to be evaluated. Competitive examination are of objective types they are conducted on machine and evaluated on machine also. These systems are very useful in terms of saving resource. It has been observed that there is no provision to extends these systems to subjective questions. Due to some problems these systems cannot be used in board examination where students write subjective answers .AI answer verifier systems will be able to check subjective answer written by student in an examination. It will allocate marks accordingly as good as human being. It will help to reduce the wastage of resources like papers and much more things. It will also overcome human limitations and speed up the overall education system. Our main aim of the system is that it will be helpful in various universities in academics for conducting class test, unit test and final exams.

Question Answering

Question Answering is a specialized form of information retrieval. Given a collection of documents, a Question Answering system attempts to retrieve correct answers to questions posed in natural language. Open-domain question answering requires question answering systems to be able to answer questions about any conceivable topic. Such systems cannot, therefore, rely on hand crafted domain specific knowledge to find and extract the correct answers. Closed-domain question answering deals with questions under a specific domain (for example, medicine or automotive maintenance), and can be seen as an easier task because NLP systems can exploit domain-specific knowledge frequently formalized in ontologies. Alternatively, closed-domain might refer to a situation where only a limited type of questions are accepted, such as questions asking for descriptive rather than procedural information. Our system also comes under closed domain QA where we are supposed for accessing online based subjective examination.

Subjective Examination

Subjective examination has been a major way of evaluating a candidate's knowledge & understanding about on course or subject in traditional education system for centuries (Minsu Jang et al. 2007). Every university has its own examination pattern based on subjective examination. So in this global era of web based education. We need to consider such examination done online (Hanumant R. Gite, C.Namrata Mahender 2012).

Generally the questions may be considered in the following forms.

Define: explain the meaning and (often) provide an appropriate example

Describe / illustrate: present the main points with clear examples that enhance the discussion

Differentiate / distinguish: present the differences between two things

Discuss / explain: present the main points, facts, and details of a topic; give reasons

Enumerate / List / Identify / Outline: write a list of the main points with brief explanations

Interpret: present your analysis of the topic using facts and reasoning

Justify / Prove: present evidence and reasons that support the topic

Summarize: briefly state the main ideas in an organized manner

Trace: state the main points in logical or chronological order

In this paper we have discussed two issues related to examination & a simple Psycho based solution is provided.

Online Subjective Examination

Our system works on an attempt to consider candidates answer by extracting the required intentional part of an answer to a prescribed template or model answer already provided in the Question answering framework. There is always an urge to justify an answer is appropriate or not. That is we have to find the confidence level for a given answer, by comparing it to the model answer. That is every word in an answer does not play an important role while evaluation process. To justify such case we have consider answer in one sentence.

Question Processing Module

The question type, usually based on a taxonomy of possible questions already coded into the system; The expected answer type, through some shallow semantic processing of the question; and The question focus, which represents the main information that is required to answer the user's question. These steps allow the question processing module to finally pass a set of query terms to the Paragraph Indexing module, which uses them to perform the information retrieval.

Answer Processing

The Answer Processing module is responsible for identifying and extracting the emphasized words which are responsible for the response of the answer.

Answer Identification

The use of a part-of-speech tagger (e.g., Python POS tagger) can help to enable recognition of answer candidates within identified model answer. Answer candidates can be ranked based on measures of distance between keywords, numbers of keywords matched and other similar heuristic metrics.

Answer Extraction

Once an answer has been identified, the shallow parsing performed is leveraged to extract only the relevant word or phrase in answer to the question.

Answer Correctness

Confidence in the correctness of an answer can be increased in a number of ways. One way is to use a lexical resource like WordNet (Synonyms) to verify that a candidate response was of the correct answer type.

One line answer

In our system we are paying attention for answer accessing majorly by considering length and paraphrasing. One line answer or Define may have a sentence which may have 10 words or 15 words as per the writing style of the candidates so we cannot fix single line answer with fixed number of words used. So only point to be find single sentence is to find the full stop.

For e.g. one line answer, expressed in different mode or synonym based answer etc. (where s is stands for original and t is for its paraphrase)

Tom purchased a Honda from John.

Tom bought a Honda from John.

It was a Honda that John sold to Tom.

1.1 WHAT IS SOFTWARE?

Software, generally sense, is understood as a group of instructions or programs that instructs to a computer to perform specific tasks. Software could be a general term that's wont to describe computer programs. Scripts, applications, programs and a group of instructions are all different terms wont to describe software.

The theory of software was first proposed by Alan Mathison Turing in 1935 in his essay "Computable numbers with an application to the Entscheidungsproblem." However, the word software was proposed by statistician and mathematician John Tukey in a very 1958 issue of yank Mathematical Monthly within which he discussed the electronic calculators' programs.

Software is typically divided into three categories:

- System software could be a base for application software. System software generally includes operating systems, device drivers, text editors, compilers, disk formatters and utilities helping the pc to regulate more efficiently. it's responsible in providing basic non-specific-task functionalities and management of hardware components. The system software is typically written within the language of C programming.
- Programming software could also be a group of tools to help developers in writing programs. the numerous tools available are linkers, compilers, interpreters debuggers and text editors.
- Application software is typically used to perform certain tasks and also the samples of the applying software includes educational software, database managing systems, office suites, application on gaming. the applying software can either be one program or a gaggle of portable programs.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

For a description of standard objects and modules, see [The Python Standard Library](#). [The Python Language Reference](#) gives a more formal definition of the language. To write extensions in C or C++, read [Extending and Embedding the Python Interpreter](#) and [Python/C API Reference Manual](#). There are also several books covering Python in depth.

This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in [The Python Standard Library](#).

The Python Standard Library

While The Python Language Reference describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components

Dealing with Bugs

Python is a mature programming language which has established a reputation for stability. In order to maintain this reputation, the developers would like to know of any deficiencies you find in Python.

It can be sometimes faster to fix bugs yourself and contribute patches to Python as it streamlines the process and involves less people. Learn how to contribute.

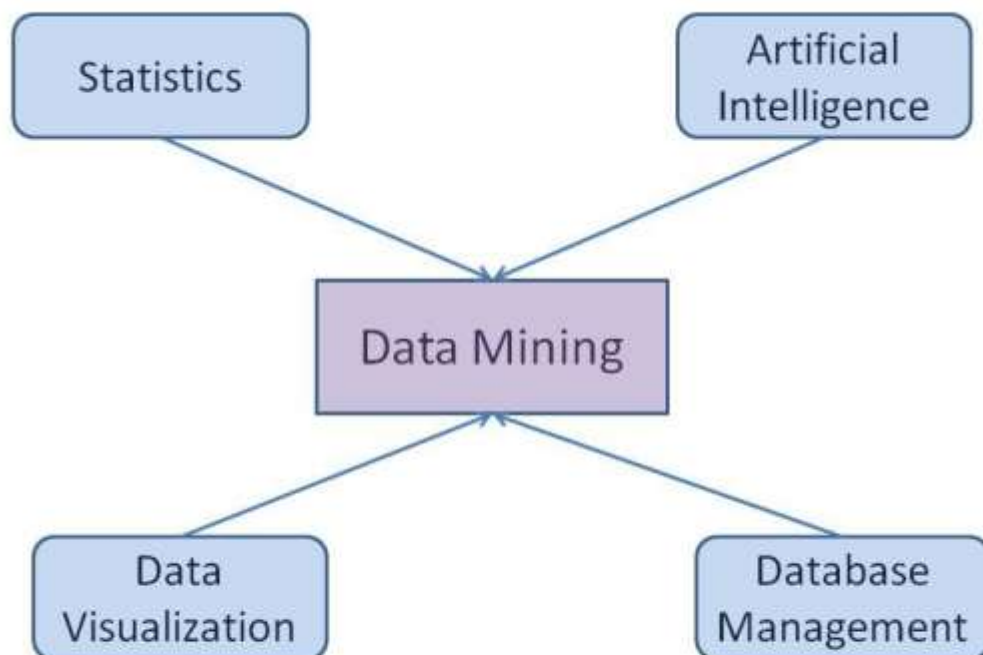
Documentation bugs

If you find a bug in this documentation or would like to propose an improvement, please submit a bug report on the tracker. If you have a suggestion how to fix it, include that as well.

If you're short on time, you can also email documentation bug reports to docs@python.org (behavioral bugs can be sent to python-list@python.org). 'docs@' is a mailing list run by

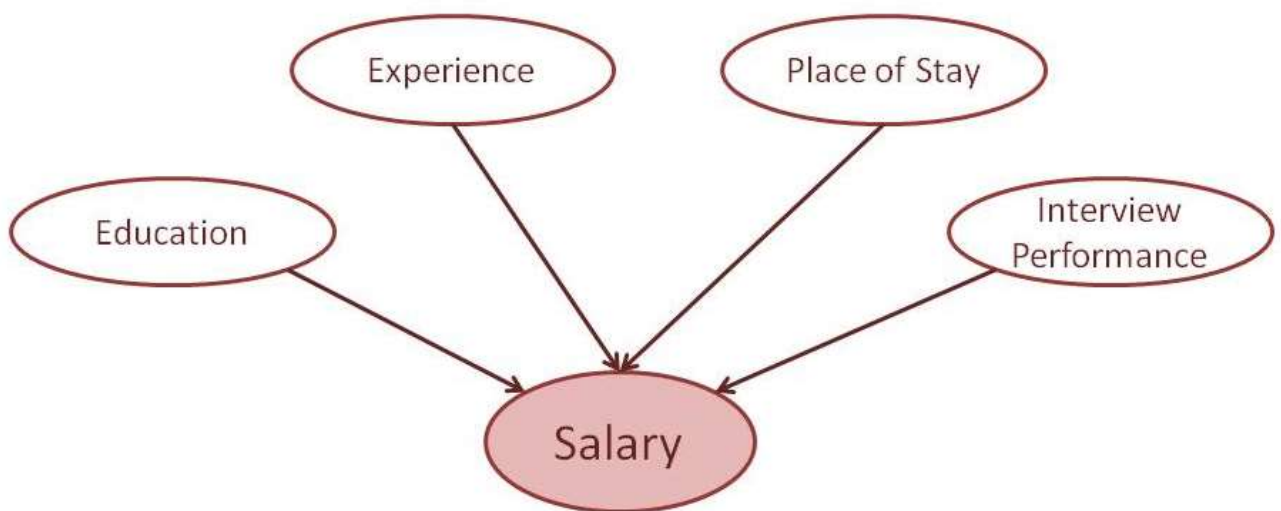
Introduction to Data Mining

Data mining integrates approaches and techniques from various disciplines such as machine learning, statistics, artificial intelligence, neural networks, database management, data warehousing, data visualization, spatial data analysis, probability graph theory etc. In short, data mining is a multi-disciplinary field.



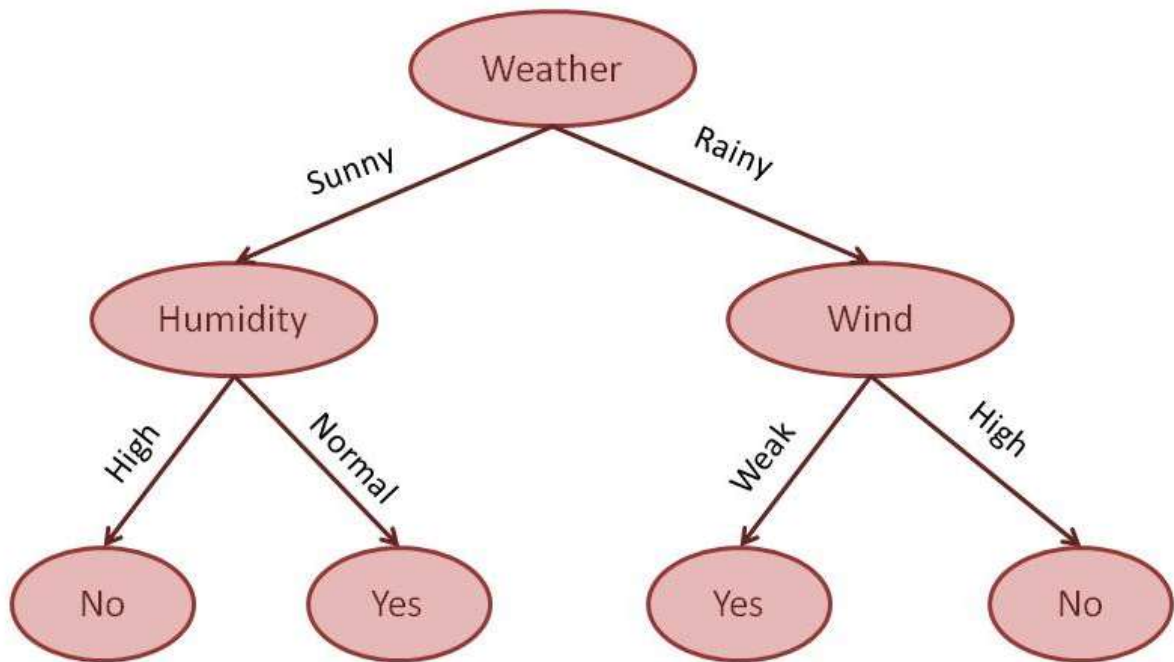
Statistics

Statistics includes a number of methods to analyze numerical data in large quantities. Different statistical tools used in data mining are regression analysis, cluster analysis, correlation analysis and Bayesian network. Statistical models are usually built from a training data set. Correlation analysis identifies the correlation of variables to each other. Bayesian network is a directed graph that represents casual relationship among data found out using the Bayesian probability theorem. Given below is a simple Bayesian network where the nodes represent variables whereas edges represent the relationship between the nodes.



Machine Learning

Machine learning is the collection of methods, principles and algorithms that enables learning and prediction on the basis of past data. Machine learning is used to build new models and to search for a best model matching the test data. Machine learning methods normally use heuristics while searching for the model. Data mining uses a number of machine learning methods including inductive concept learning, conceptual clustering and decision tree induction. A decision tree is a classification tree that decides the class of an object by following the path from the root to a leaf node. Given below is a simple decision tree that is used for weather forecasting.



Database Oriented Techniques

Advancements in database and data warehouse implementation helps data mining in a number of ways. Database oriented techniques are used mainly to develop characteristics of the available data. Iterative database scanning for frequent item sets, attribute focusing, and attribute oriented induction are some of the database oriented techniques widely used in data mining. The iterative database scanning searches for frequent item sets in a database. Attribute oriented induction generalizes low level data into high level concepts using conceptual hierarchies.

Neural Networks

A neural network is a set of connected nodes called neurons. A neuron is a computing device that computes some requirement of its inputs and the inputs can even be the outputs of other neurons. A neural network can be trained to find the relationship between input attributes and output attribute by adjusting the connections and the parameters of the nodes.

Data Visualization

The information extracted from large volumes of data should be presented well to the end user and data visualization techniques make this possible. Data is transformed into different visual objects such as dots, lines, shapes etc and displayed in a two or three dimensional space. Data visualization is an effective way to identify trends, patterns, correlations and outliers from large amounts of data.

Summary

Data mining combines different techniques from various disciplines such as machine learning, statistics, database management, data visualization etc. These methods can be combined to deal with complex problems or to get alternative solutions. Normally data mining system employs one or more techniques to handle different kinds of data, different data mining tasks, different application areas and different data requirements.

Patterns in Data Mining

1. Association

The items or objects in relational databases, transactional databases or any other information repositories are considered, while finding associations or correlations.

2. Classification

- The goal of classification is to construct a model with the help of historical data that can accurately predict the value.
- It maps the data into the predefined groups or classes and searches for the new patterns.

For example:

To predict weather on a particular day will be categorized into - sunny, rainy, or cloudy.

3. Regression

- Regression creates predictive models. Regression analysis is used to make predictions based on existing data by applying formulas.
- Regression is very useful for finding (or predicting) the information on the basis of previously known information.

4. Cluster analysis

- It is a process of portioning a set of data into a set of meaningful subclass, called as cluster.
- It is used to place the data elements into the related groups without advanced knowledge of the group definitions.

5. Forecasting

Forecasting is concerned with the discovery of knowledge or information patterns in data that can lead to reasonable predictions about the future.

Technologies used in data mining

Several techniques used in the development of data mining methods. Some of them are mentioned below:

1. Statistics:

- It uses the mathematical analysis to express representations, model and summarize empirical data or real world observations.
- Statistical analysis involves the collection of methods, applicable to large amount of data to conclude and report the trend.

2. Machine learning

- Arthur Samuel defined machine learning as a field of study that gives computers the ability to learn without being programmed.
- When the new data is entered in the computer, algorithms help the data to grow or change due to machine learning.
- In machine learning, an algorithm is constructed to predict the data from the available database (Predictive analysis).
- It is related to computational statistics.

The four types of machine learning are:

1. Supervised learning

- It is based on the classification.
- It is also called as inductive learning. In this method, the desired outputs are included in the training dataset.

2. Unsupervised learning

Unsupervised learning is based on clustering. Clusters are formed on the basis of similarity measures and desired outputs are not included in the training dataset.

3. Semi-supervised learning

Semi-supervised learning includes some desired outputs to the training dataset to generate the appropriate functions. This method generally avoids the large number of labeled examples (i.e. desired outputs) .

4. Active learning

- Active learning is a powerful approach in analyzing the data efficiently.
- The algorithm is designed in such a way that, the desired output should be decided by the algorithm itself (the user plays important role in this type).

3. Information retrieval

Information deals with uncertain representations of the semantics of objects (text, images).
For example: Finding relevant information from a large document.

1.2 WHAT IS SOFTWARE DEVELOPMENT LIFE CYCLE(SDLC)

The software development life cycle can be a framework that defines the tasks performed at each step within the software development process. SDLC could be a structure followed by a development team within the software organization. It consists of an in depth plan describing the way to develop, maintain and replace specific software. The life cycle defines a technique for improving the standard of software and therefore the overall development process. The software development life cycle is additionally called the software development process. SDLC consists of following activities:

- 1.Planning: the foremost important parts of software development, requirement gathering or requirement analysis are usually done by the foremost skilled and experienced software engineers within the organization. After the necessities are gathered from the client, a scope document is made during which the scope of the project is decided and documented.
- 2.Implementation: The software engineers start writing the code in keeping with the client's requirements.
- 3.Testing: this is often the method of finding defects or bugs within the created software.

4.Documentation: Every step within the project is documented for future reference and for the development of the software within the development process. the planning documentation may include writing the appliance programming interface (API).

5.Deployment and maintenance: The software is deployed after it's been approved for release.

6.Maintaining: Software maintenance is completed for future reference. Software improvement and new requirements (change requests) can take longer than the time needed to form the initial development of the software.

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

SDLC (Spiral Model):

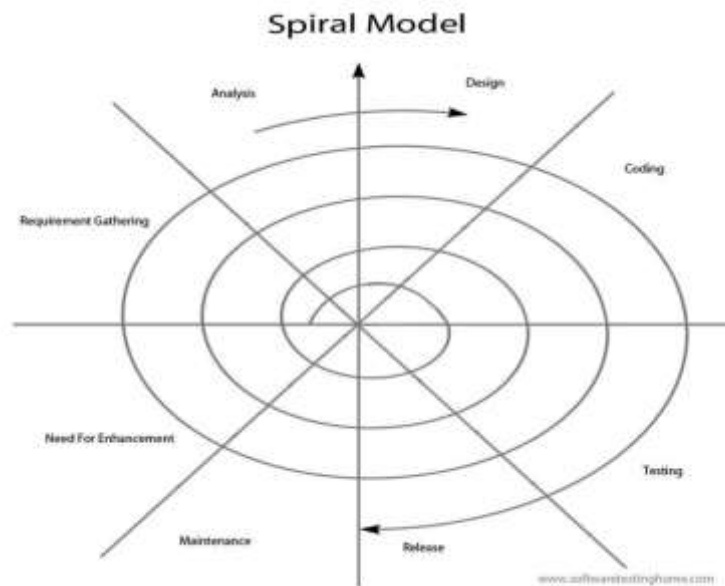


Fig 1: Spiral Model

Stages of SDLC:

- Requirement Gathering and Analysis
- Designing
- Coding
- Testing
- Deployment

Requirements Definition Stage and Analysis:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.

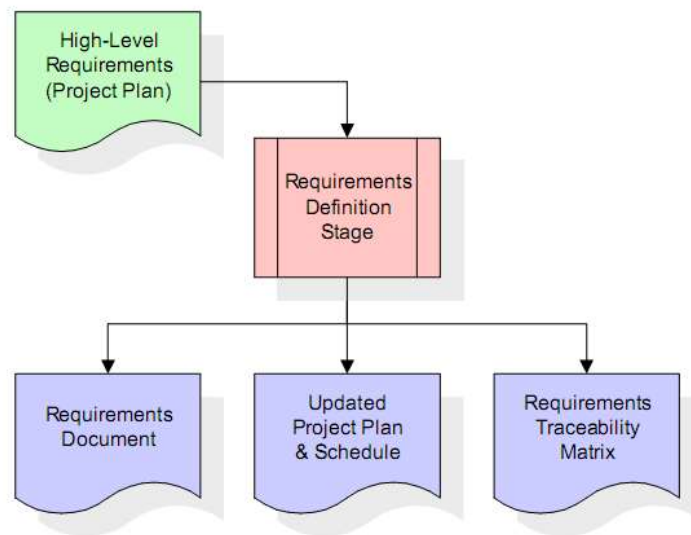


Fig 2: Requirement Stage

These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). the requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document. The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term *requirements traceability*. The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

Design Stage:

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

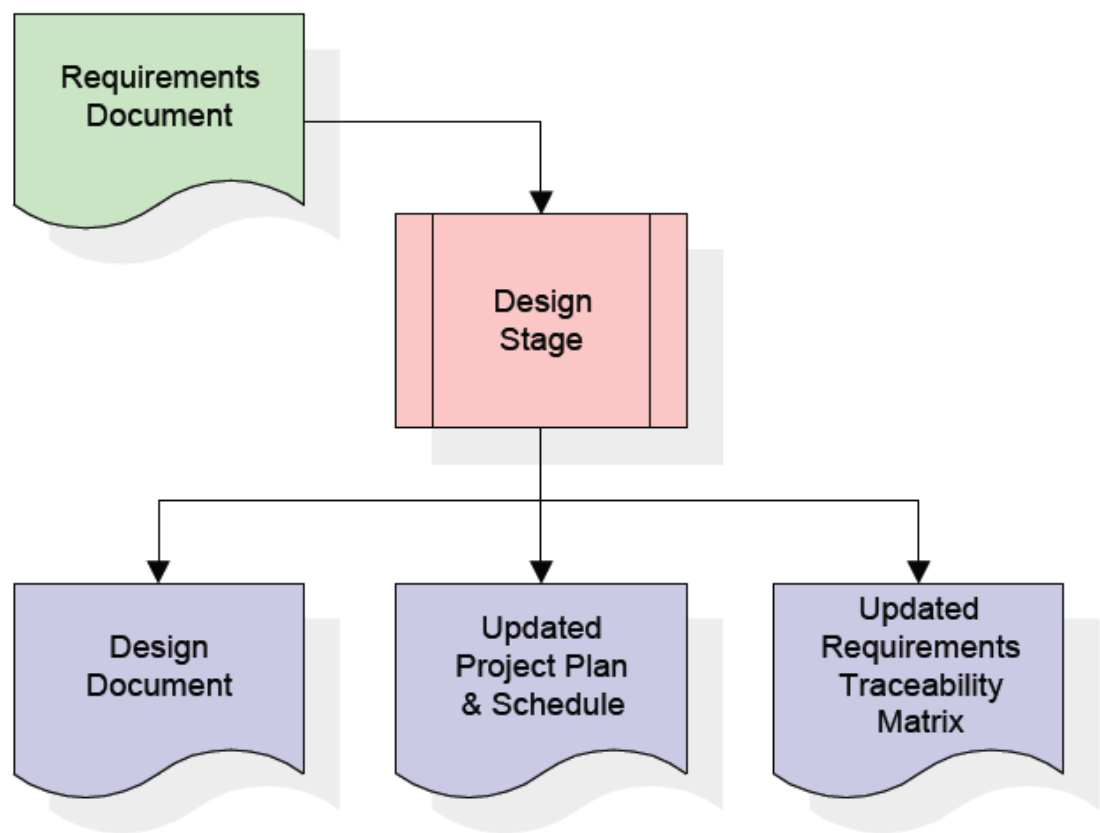


Fig 3: Design Stage

When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

Development Stage:

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be

produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

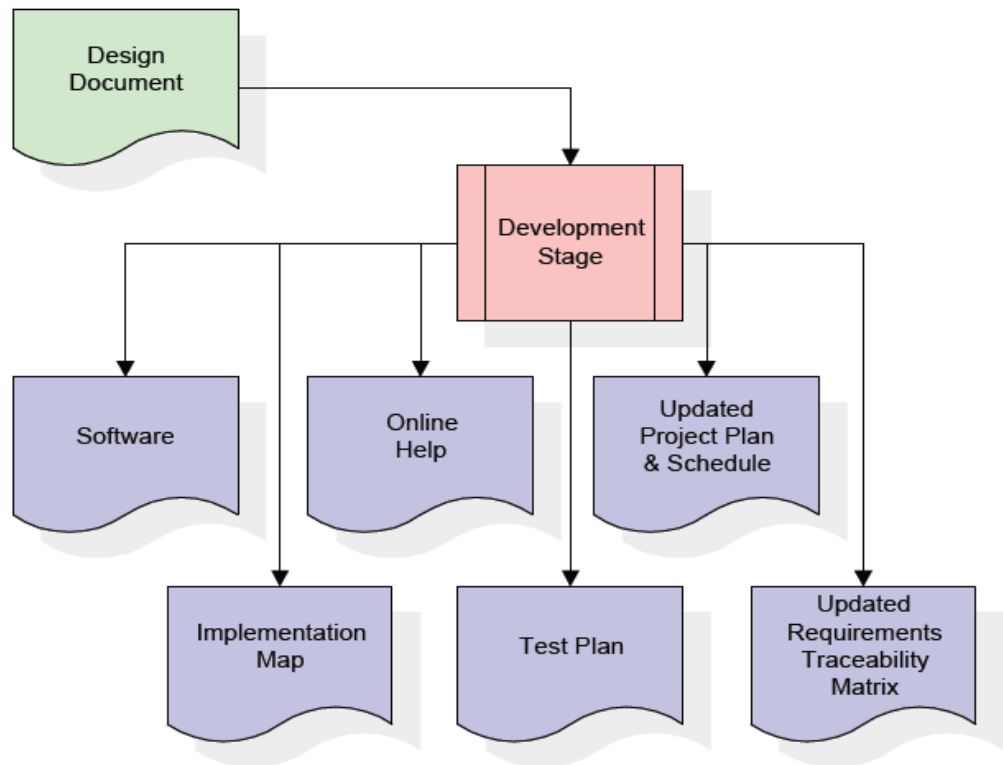


Fig 4: Development Stage

The RTM will be updated to show that each developed artefact is linked to a specific design element, and that each developed artefact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

Integration & Test Stage:

During the integration and test stage, the software artefacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability.

During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

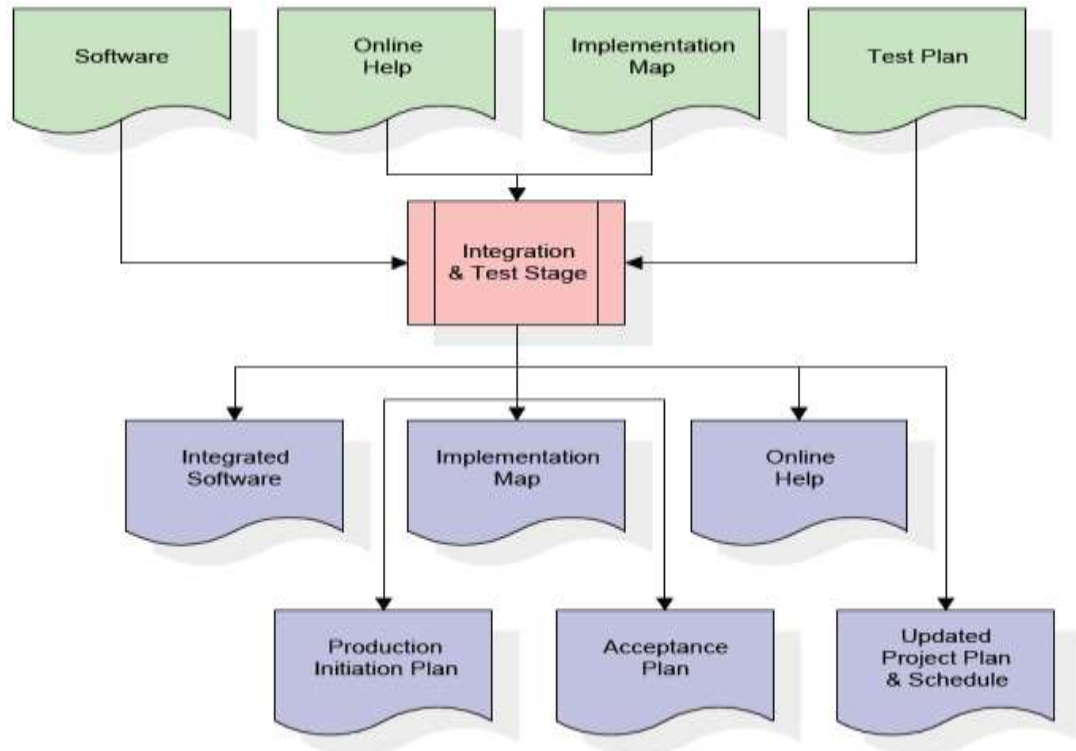


Fig 5: Integration and Test stage

The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

Installation & Acceptance Stage

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.

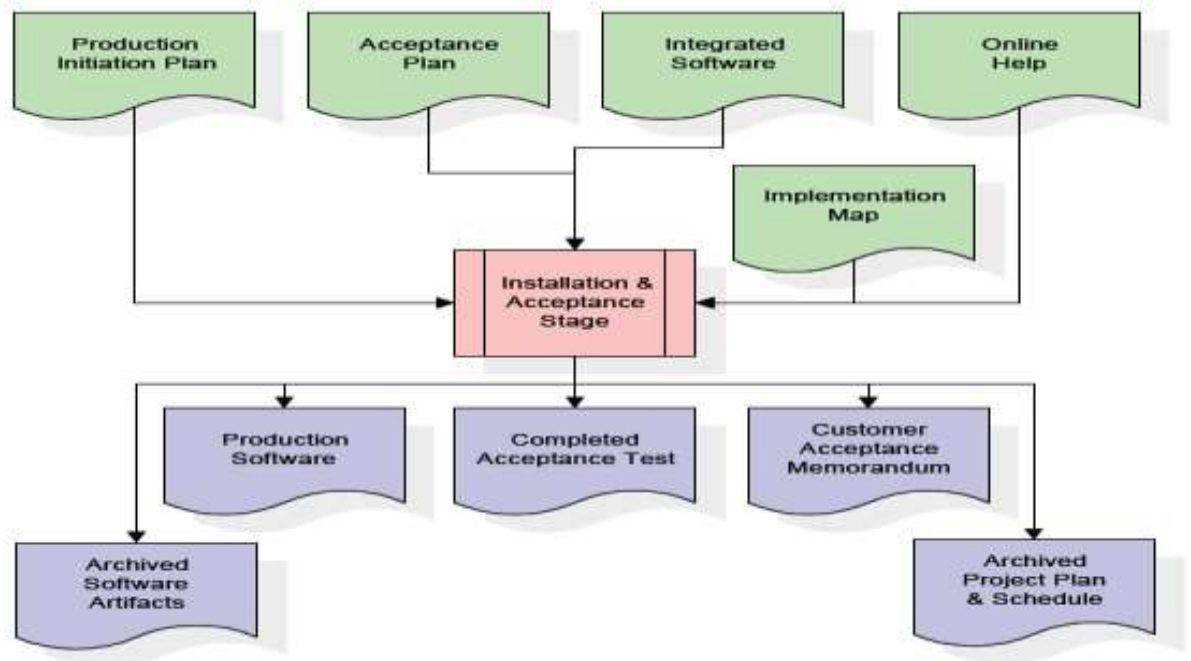


Fig 6: Installation and Acceptance Stage

The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labour data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

2.4 SYSTEM ARCHITECTURE

Architecture Flow:

Below architecture diagram represents mainly flow of request from the users to database through servers. In this scenario overall system is designed in three tiers separately using three layers called presentation layer, business layer, data link layer. This project was developed using 3-tier architecture.

3-Tier Architecture:

The three-tier software architecture (three layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging.

The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability,

reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems..

Advantages of Three-Tier:

- Separates functionality from presentation.
- Clear separation - better understanding.
- Changes limited to well define components.

CHAPTER-2

LITERATURE REVIEW

[1] Sheeba Praveen, “An Approach to Evaluate Subjective Questions for Online Examination System”, International Journal of Innovative Research in Computer and Communication Engineering. Vol. 2, Issue 11, November 2014. This system solves the problem of deducing knowledge represented by partially or grammatically incorrect sentences, and will translate the meaning conveyed by the student in different forms and sentences, propose a normalized plan of action for grading the answers, ways to interpret the mathematical formulas and expressions however the system is limited to non-mathematical subjects only.

[2] Amarjeet Kaur, M Sasikumar, Shikha Nema, Sanjay Pawar(2013), “Algorithm for Automatic Evaluation of Single Sentence Descriptive Answer”, International Journal of Inventive Engineering and Sciences (IJIES). Single sentence descriptive answers which are grammatically correct and have no spelling errors is considered as the text. Their approach is to represent standard answers in the form of graphs and then comparing it by applying similarity measures for the allocation of marks.

[3] Aditi Tulaskar¹, Aishwarya Thengal², Kamlesh Koyande³, “Subjective Answer Evaluation System”, Department of Information Technology Vidyalankar Institute of Technology, Mumbai, India. The proposed system will allot the marks according to the percentage of accuracy present in the answer. This is a software system in which students will be authenticated by using student login. After the authentication, students will be provided with the questions. The proposed system is designed to evaluate answers for five students providing five different answers. The standard answer is stored in the database with the keywords, meaning and the description of that answer. Then each answer is evaluated by matching the keywords as well as its synonyms with the standard answer. It will also check the grammar and spellings of the words. After the evaluation, the answer is graded depending on the correctness of it.

[4] Asmita Dhokrat, Gite Hanumant R., C. Namrata Mahender (2017), "Automated Answering for Subjective Examination", (IJCA). The proposed system has two login facilities. The user login is the login allocated for the students. As soon as you click the student login button you will be asked to enter login id and password. The system will check for the id and automatically display students' name, email id and phone number for verification. The user login writes answers with respect to the question uploaded. The system will show marks scored as soon as you enter the next button. The admin login will let the teachers login. In admin login each user will have his own password and id through which they can login in to the system. The admin can add/subtract questions, check for students' marks and so on. Just like the teachers can do manually.

[5] Merien Mathew, Ankit Chavan, Siddharth Baikar, "ONLINE SUBJECTIVE ANSWER CHECKER", International Journal of Scientific & Engineering Research, 2017. The original answer is required to be stored in the system. This is done by the admin. The admin may insert any number of questions and respective subjective answers. The answers are stored as notepad files. When a user takes the test he/she is provided with questions and area to type the answer. Once the user enters his/her answers, these answers are then compared by the system with the original answers written in the database and marks are allocated accordingly. Both the answers need not be exactly the same or word to word. The system consists of built AI sensors that verify answers and allocate marks accordingly as good as a human being

1. FEASIBILITY STUDY

Feasibility study is conducted once the problem is clearly understood. The feasibility study which is a high level capsule version of the entire system analysis and design process. The objective is to determine whether the proposed system is feasible or not and it helps us to the minimum expense of how to solve the problem and to determine, if the Problem is worth solving. The following are the three important tests that have been carried out for feasibility study.

This study tells about how this package is useful to the users and its advantages and disadvantages, and also it tells whether this package is cost effective are not. There are three types of feasibility study, they are

- Economic Feasibility.
- Technical Feasibility.
- Operational Feasibility.

3.1 TECHNICAL FEASIBILITY

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis. Understand the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies. Is the required technology available with the organization?

3.2 OPERATIONAL FEASIBILITY

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the

system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

- ☐ Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.
- ☐ Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.
- ☐ Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project
- ☐ Have the user been involved in the planning and development of the project?
- ☐ Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

3.3 ECONOMIC FEASIBILITY

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

System Design

UML Diagrams

UML (Unified Modeling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. It was initially started to capture the behavior of complex software and non-software system and now it has become an OMG standard. This tutorial gives a complete understanding on UML.

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously making efforts to create a truly industry standard.

- UML stands for Unified Modeling Language.
- UML is different from the other common programming languages such as C++, Java, COBOL, etc.
- UML is a pictorial language used to make software blueprints.
- UML can be described as a general-purpose visual modeling language to visualize, specify, construct, and document software system.
- Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object- oriented analysis and design. After some standardization, UML has become an OMG standard.

Components of the UML

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete.

UML includes the following nine diagrams, the details of which are described in the subsequent chapters.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

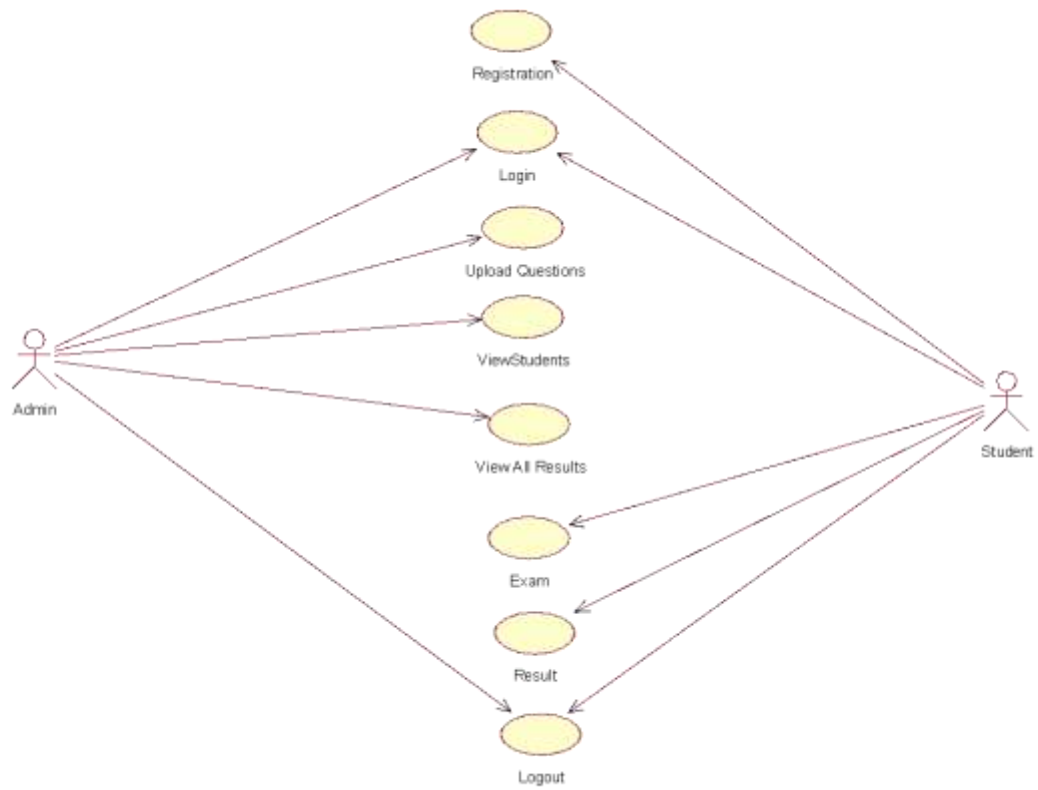
The following are the main components of uml: -

1. Use-case Diagram
2. Class Diagram
3. Sequence Diagram
4. Activity Diagram
5. Collaboration Diagram

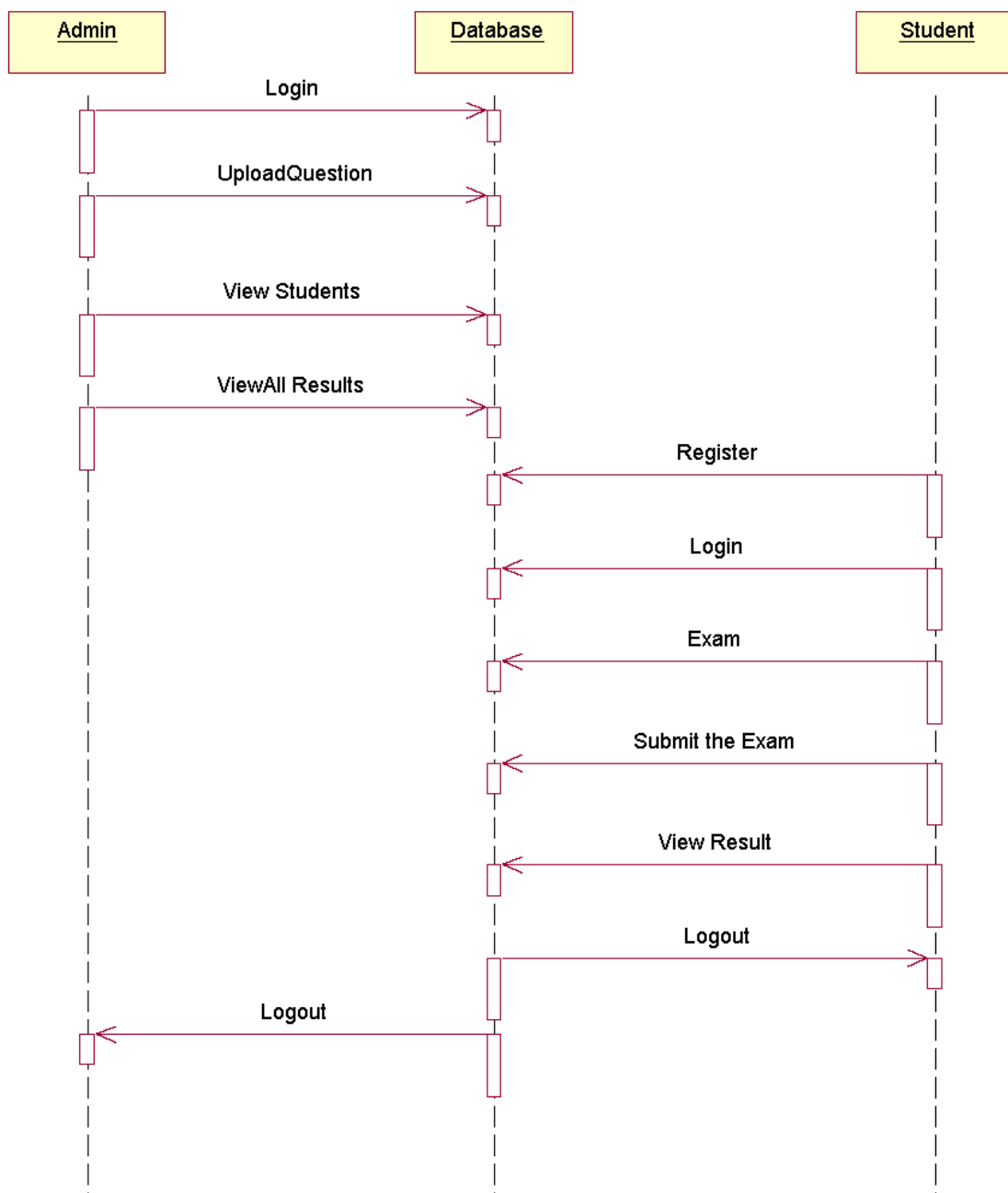
4.4 UML DIAGRAMS

4.4.1 USE CASE DIAGRAM

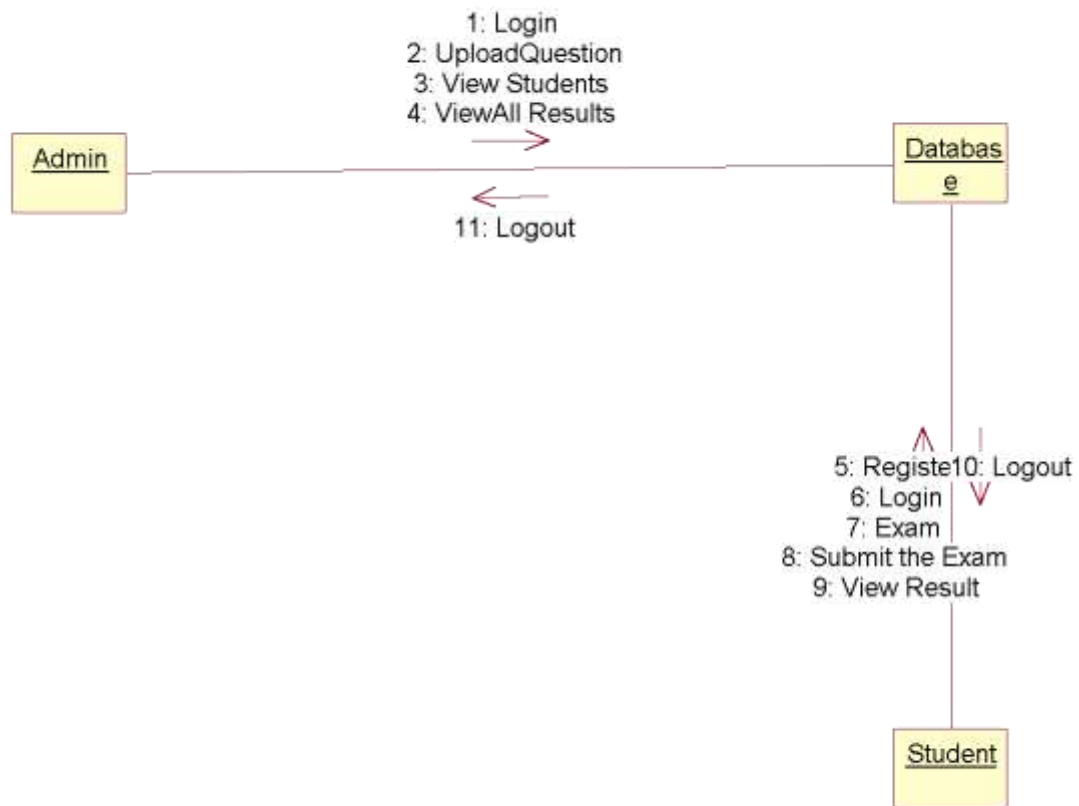
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies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



4.4.2 STATE DIAGRAM



Sequence



Collaboration diagram

1. TESTING

SYSTEM TESTING

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

TYPES OF TESTS

Unit testing

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

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is

more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

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

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

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

System Test

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

White Box Testing

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

Black Box Testing

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

6.1 Unit Testing:

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

Test strategy and approach

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

Test objectives

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

Features to be tested

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

6.2 Integration Testing

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

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

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

6.3 Acceptance Testing

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

CHAPTER-6

IMPLEMENTATION& RESULTS

7.2 CODING & TESTING

```
import pathlib
```

```
from flask import Flask
```

```
from flask import Flask, flash, redirect, render_template, request, session, abort
```

```

import os

import pandas

#from werkzeug import secure_filename

import mysql.connector

import sys

import glob

import math

import numpy as np

from gtts import gTTS

import speech_recognition as sr

import nltk

#nltk.download('punkt')

#nltk.download('all')


noise_test=["is","a","this","an","the","are","?", "for","of","who","in","many","the
y",

"by","who","what","why","when","where","does","how","there","which","any",

"it","have","if","on","or","st","nd","/", "and","has","can","you","was",]


def remove_noise(input_text):

    words=input_text.split()

```

```

noise_free_words=[word for word in words if word not in noise_test]

noise_free_text=" ".join(noise_free_words)

return noise_free_text


rdata=[]

subject=""

app = Flask(__name__)

db= mysql.connector.connect(user='root', database='online')


@app.route('/')

def home():

    return render_template('index.html')

@app.route('/aboutus')

def aboutus():

    return render_template('aboutus.html')


@app.route('/adminlogin')

def adminlogin():

    return render_template('adminlogin.html')


@app.route('/adminhome')

def adminhome():

    return render_template('adminhome.html')

```

```
@app.route('/userhome')
```

```
def userhome():
```

```
    return render_template('userhome.html')
```

```
@app.route('/QueryM')
```

```
def QueryM():
```

```
    return render_template('QueryM.html',msg="")
```

```
@app.route('/userlogin')
```

```
def userlogin():
```

```
    return render_template('userlogin.html')
```

```
@app.route('/userreg')
```

```
def userreg():
```

```
    return render_template('userreg.html')
```

```
@app.route('/Query1')
```

```
def Query1():
```

```
    return render_template('query.html')
```

```
@app.route('/Query')
```

```
def Query():  
    return render_template('query1.html')  
  
@app.route('/upload')  
def Upload():  
    return render_template('upload.html')  
  
@app.route('/adminlogin', methods=['POST'])  
def do_adminlogin():  
    flag=False  
    username=request.form['username']  
    password=request.form['password']  
    if username=='admin' and password=='admin':  
        session['logged_in'] = True  
        flag=True  
    else:  
        flag=False  
    if flag:  
        return render_template('adminhome.html')  
    else:  
        return render_template('adminlogin.html')
```

```

@app.route('/userlogin', methods=['POST'])

def do_userlogin():

    flag=False

    cursor = db.cursor()

    username=request.form['username']

    password=request.form['password']

    sql = "SELECT * FROM users WHERE userid= '%s' and password = '%s' " %
(username,password)

    print("Sql is ",sql)

    cursor.execute(sql)

    rows_count = cursor.fetchall()

    print("XXX ",type(rows_count))

    if len(rows_count) > 0:

        session['logged_in'] = True

        session['uid'] = username

        flag=True

    else:

        flag=False

    if flag:

        return render_template('userhome.html')

    else:

        return render_template('userlogin.html')

```

```
@app.route('/UserResult')
```

```
def do_UserResult():
```

```
    flag=False
```

```
    cursor = db.cursor()
```

```
    username=session['uid']
```

```
    sql = "SELECT * FROM result where username='"+username+"'"
```

```
    print("Sql is ",sql)
```

```
    cursor.execute(sql)
```

```
    rows_count = cursor.fetchall()
```

```
    print(rows_count)
```

```
    return render_template('UserResult.html',data=rows_count)
```

```
@app.route('/Result')
```

```
def do_AllUserResult():
```

```
    flag=False
```

```
    cursor = db.cursor()
```

```
    sql = "SELECT * FROM result"
```

```
    print("Sql is ",sql)
```

```
    cursor.execute(sql)
```

```
    rows_count = cursor.fetchall()
```

```
    print(rows_count)
```

```
return render_template('AllUserResult.html',data=rows_count)
```

```
@app.route('/UserList')
```

```
def do_UserList():
```

```
    flag=False
```

```
    cursor = db.cursor()
```

```
    sql = "SELECT * FROM users"
```

```
    print("Sql is ",sql)
```

```
    cursor.execute(sql)
```

```
    rows_count = cursor.fetchall()
```

```
    print(rows_count)
```

```
    return render_template('userlist.html',data=rows_count)
```

```
@app.route('/registration', methods=['POST'])
```

```
def do_registration():
```

```
    flag=False
```

```
    name=request.form['name']
```

```
    email=request.form['email']
```

```
    phone=request.form['phone']
```

```
    address=request.form['address']
```

```

userid=request.form['userid']

password=request.form['password']

cursor1 = db.cursor()

s='insert      into      users(name,email,phonenumber,address,userid,password)
values("%s","%s","%s","%s","%s","%s")' %
(name,email,phone,address,userid,password)

print("Sql is ",s)

cursor1.execute(s)

db.commit()

return render_template('userlogin.html')

```

```
@app.route('/QueryDBM', methods=['POST'])
```

```
def QueryDBM():
```

```
    global rdata
```

```
    global subject
```

```
    s=set()
```

```
    file = pathlib.Path("welcome.mp3")
```

```
    if file.exists ():
```

```
        os.remove("welcome.mp3")
```

```
    ques=[]
```

```
    ans=[]
```

```

s3=""

r = sr.Recognizer()

with sr.Microphone() as source:

    print("Speak:")

    audio = r.listen(source)


try:

    s3=r.recognize_google(audio)

except sr.UnknownValueError:

    print("Could not understand audio")

except sr.RequestError as e:

    print("Could not request results; {0}".format(e))


print ("Ur Spech is ",s3)


subject=remove_noise(s3).lower()

f=True

if subject=='artificial' or subject=='artificialintelligence' or subject=='artificial
intelligence' or subject=='ai' or subject=='AI':

    subject='ai'

elif subject=='cyber' or subject=='cyber security' or subject=='cyber':

    subject='cybersecurity'

elif subject=='embeded' or subject=='embeded system' or subject=='embededsystem':

```

```

    subject='embededsystem'

elif subject=='big data' or subject=='bigdata' or subject=='big':

    subject='bigdata'

elif subject=='pem':

    subject='pem'

else:

    f=False

if f:

    cursor = db.cursor()

    sql="select * from "+subject

    print("Sql is ",sql)

    rows_count=cursor.execute(sql)

    data=cursor.fetchall()

    #print("Data is ",data)

    #print(data[3][1])

    import random

    while True:

        x=random.randint(1,len(data))

        s.add(x)

        if len(s)==10:

            break

```

```

    rdata=list(s)

    print("RRR New data ",rdata)

    return render_template('Query1.html',ques=data,ss=rdata,sub=subject)

else:

    return render_template('QueryM.html',msg="Please Give the Voice")


@app.route('/QueryDB', methods=['POST'])

def QueryDB():

    global rdata

    global subject

    s=set()

    subject=request.form['sub']


    cursor = db.cursor()

    sql="select * from "+subject

    print("Sql is ",sql)

    rows_count=cursor.execute(sql)

    data=cursor.fetchall()

    #print("Data is ",data)

    #print(data[3][1])

    import random

```

```
while True:
```

```
    x=random.randint(1,len(data))
```

```
    s.add(x)
```

```
    if len(s)==10:
```

```
        break
```

```
rdata=list(s)
```

```
print("RRR data ",rdata)
```

```
return render_template('Query1.html',ques=data,ss=rdata,sub=subject)
```

```
@app.route('/QueryDB1', methods=['POST'])
```

```
def QueryDB1():
```

```
    global rdata
```

```
    ctr=0
```

```
    ques=[]
```

```
    ans=[]
```

```
    username=session['uid']
```

```
    print("Rdata is ",rdata)
```

```
    s1=request.form['query1']
```

```
    s2=request.form['query2']
```

```
    s3=request.form['query3']
```

```
    s4=request.form['query4']
```

```
s5=request.form['query5']

s6=request.form['query6']

s7=request.form['query7']

s8=request.form['query8']

s9=request.form['query9']

s10=request.form['query10']

sub=request.form['sub']

if s1=='':

    s1="x"

if s2=='':

    s2="x"

if s3=='':

    s3="x"

if s4=='':

    s4="x"

if s5=='':

    s5="x"

if s6=='':

    s6="x"

if s7=='':

    s7="x"

if s8=='':

    s8="x"
```

```

if s9=='':

    s9="x"

if s10=='':

    s10="x"

ans.append(s1)

ans.append(s2)

ans.append(s3)

ans.append(s4)

ans.append(s5)

ans.append(s6)

ans.append(s7)

ans.append(s8)

ans.append(s9)

ans.append(s10)

print(ans)

print("R data is : ",rdata)

ind=0

while ind<len(rdata):

    #s3=s3.replace('?',')

    #print ("Ur Spech is ",s3)

    X=remove_noise(ans[ind])

    cursor = db.cursor()

    sql="select * from "+subject+" where id= '"+str(rdata[ind])+'"

```

```
print("sql is ",sql)

rows_count=cursor.execute(sql)

data=cursor.fetchall()


Y=remove_noise(data[0][2])

print("X value is "+X)

print("Y value is "+Y)

# Program to measure the similarity between

# two sentences using cosine similarity.

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize


# X = input("Enter first string: ").lower()

# Y = input("Enter second string: ").lower()


#Y ="Lights out horror movie"

#X ="love horror movies"


#Y ="I love horror movies"

# tokenization
```

```

X_list = word_tokenize(X)

Y_list = word_tokenize(Y)


# sw contains the list of stopwords

sw = stopwords.words('english')

l1=[];l2=[]


# remove stop words from the string

X_set = {w for w in X_list if not w in sw}

Y_set = {w for w in Y_list if not w in sw}


# form a set containing keywords of both strings

rvector = X_set.union(Y_set)

for w in rvector:

    if w in X_set: l1.append(1) # create a vector

    else: l1.append(0)

    if w in Y_set: l2.append(1)

    else: l2.append(0)

c = 0


# cosine formula

for i in range(len(rvector)):

    c+= l1[i]*l2[i]

```

```

        cosine = c / float((sum(l1)*sum(l2))**.5)

        print("similarity: ", cosine)

        if cosine>.50:

            ctr=ctr+1

            ind=ind+1

            print("Passs ",ind+1,' ',cosine)

    cursor1 = db.cursor()

    eresult="Fail"

    if ctr>=4:

        eresult="Pass"

    s='insert      into      result(username,subject,eresult,passinnumberofsubjects)
values("%s","%s","%s","%s")' % (username,sub,eresult,str(ctr))

    print("Sql is ",s)

    cursor1.execute(s)

    db.commit()

    return render_template('Result.html',res=eresult)

@app.route('/suggestion')

def suggestionDB():

    cursor = db.cursor()

    rows_count=cursor.execute("select * from suggestion")

    data=cursor.fetchall()

    return render_template('adminhome2.html',ddata=data)

```

```

@app.route('/uploadDB', methods=['POST'])

def do_uploadDB():

    name = request.form['fname']

    f = request.files['files']

    df = pandas.read_csv(f,encoding='cp1252')

    #print(df)

    cursor1 = db.cursor()

    #cursor1.execute("delete from quest")

    #db.commit()


    for index, row in df.iterrows():

        cursor = db.cursor()

        cursor.execute('insert into '+name+'(question,answer) values("%s", "%s")' % \

            (row['Questions'],row['Answers']))

        db.commit()

    return render_template('adminhome.html',msg="File UploadedSuccessfully")


@app.route("/logout")

```

```
def logout():
```

```
    session['logged_in'] = False
```

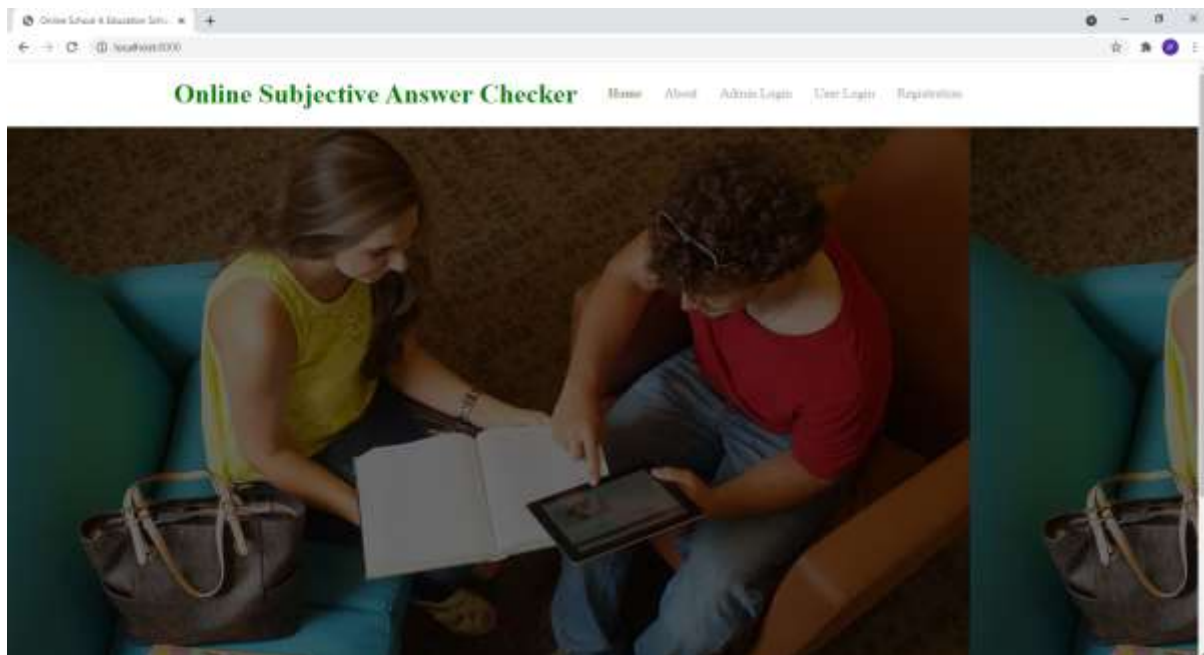
```
    return home()
```

```
if __name__ == "__main__":
```

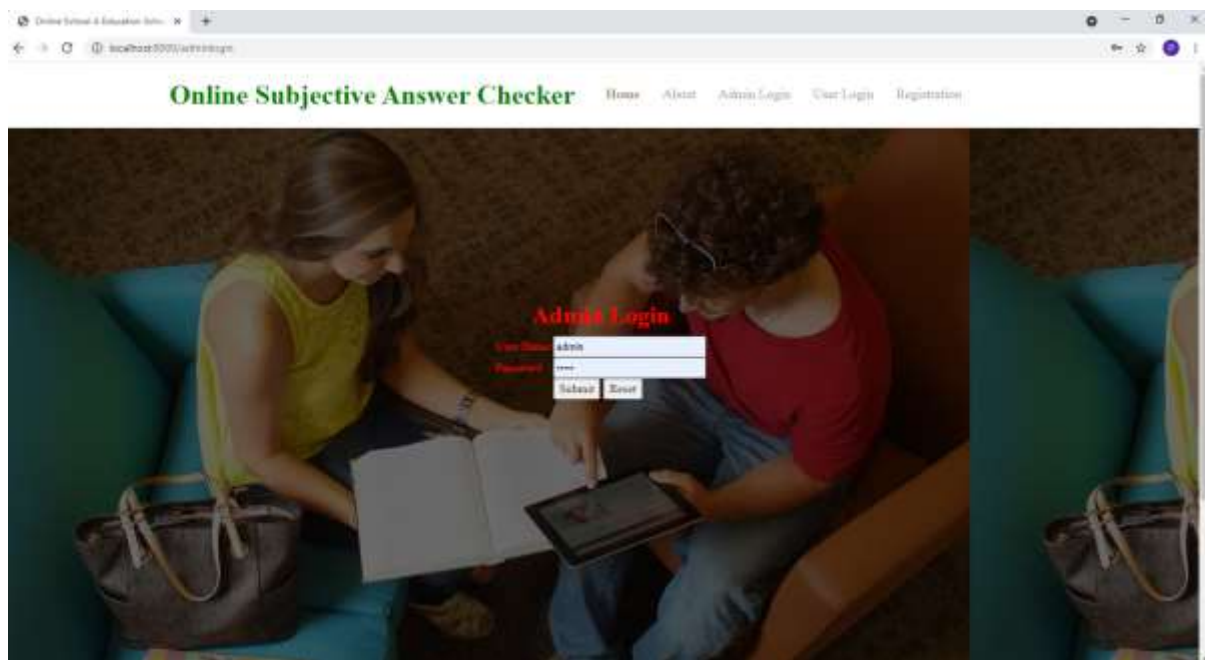
```
    app.secret_key = os.urandom(12)
```

```
    app.run(debug=True,host='0.0.0.0', port=8000)
```

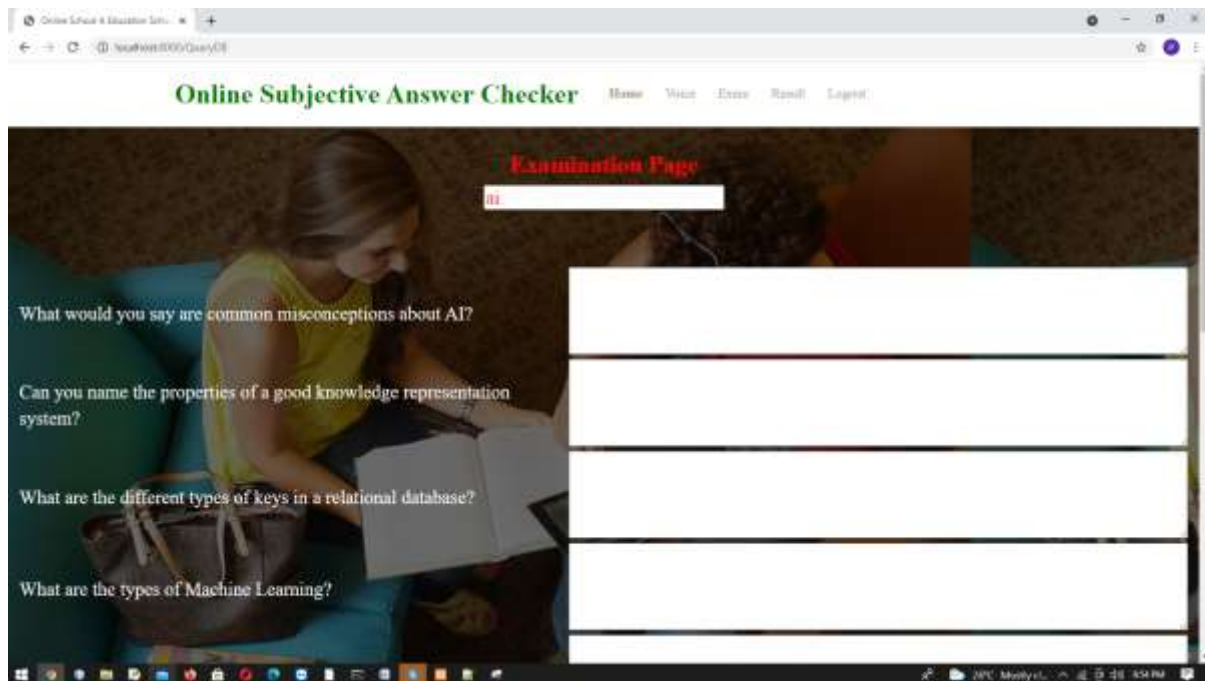
screen shots



Home page







CONCLUSION

Third party tool can be used for the grammar checking and spell checking module for better accuracy. With proper grammar checker the algorithm can also support compound and complex answers. 2. The spell checker module provides limited relaxation to the students. The available relaxation rules should vary according to different questions. There is provision to insert words in the files. New words can be added for new question-answer sets accordingly. The files can be upgraded so that the design can support a wide variety of answers.

REFERENCES

[1]<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=315646&queryText%3DA%3Aanswer+Checker>