**ARTIFICIAL INTELLIGENT STUDENT INFORMATION NETWORK**

A PROJECT REPORT

*Submitted by*

**Anandha Vignesh .M (111712104007)**

**Ashwin .S (111712104013)**

**Krishna .P (111712104048)**

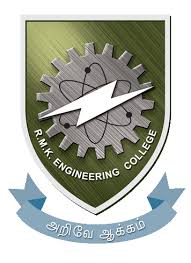
*In partial fulfilment for the award of the degree*

*Of*

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RMK ENGINEERING COLLEGE, KAVARAIPETTAI

ANNA UNIVERSITY :: CHENNAI 600 025

APRIL 2016

**ANNA UNIVERSITY::CHENNAI – 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report “**ARTIFICIALLY INTELLIGENT STUDENT INFORMATION NETWORK**” is the bonafide work of **ANANDHA VIGNESH[111712104007] , ASHWIN[111712104013], KRISHNA[111712104048]** who carried out the project under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **DR. K.L.SHANMUGANATHAN,**  **M.E., M.S., FIE, Ph.D.,**  HEAD OF THE DEPARTMENT  PROFESSOR  Department of Computer Science and  Engineering,  R.M.K Engineering College,  Kavaraipettai - 601 206. | **SIGNATURE**  **DR. K.L.SHANMUGANATHAN,**  **M.E., M.S., FIE, Ph.D.,**  SUPERVISOR  PROFESSOR  Department of Computer Science and  Engineering,  R.M.K Engineering College,  Kavaraipettai - 601 206. |

Submitted to Project Viva Voce Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| **EXTERNAL EXAMINER** | **INTERNAL EXAMINER** |

**ABSTRACT**

The college faculties face a rough time searching for data in the cabinets or computers and as a result they take a long time before they could collect the data, do the analysis, interpret and then present the result. Also a possibility of erroneous presented data is common in manual analysis. The Computers store the data effectively but there arises a need for a mining tool for analysis and interpretation and a HTML-CSS front end to present the result interactively. This report provides the ways to implement an application to act as an artificially intelligent user-friendly agent to mine the available information feedback from students and faculties to present it appropriately. It will also perform the appropriate probability and numerical calculations for setting up question papers, assigning salary and many more. The user interaction is either through touch or voice input and the output is viewing or downloading the related information.

**ACKNOWLEDGEMENT**

Support on demand, encouragement at the needed moment and guidance in the right direction are in dispensable for the success of any project. We have received these in excess from all corners from various people, we are glad to submit our gratitude to them.

We thank **Shri. R.S.Munirathinam, Chairman,** and **Shri. R.M.Kishore, Vice Chairman** of RMK group of Institutions for extending a generous hand in providing the best of resources to the college. Dr**. Elwin Chandra Monnie, the esteemed Head of our Institution** has been a source of motivation to all the staffs and students of our college. We are so much thankful to him. Our sincere thanks to **Dr. Mohammed Junaid, the Vice Principal** for giving the continuous support for the completion of our project.

Our sincere thanks to **Dr. K.L.Shanmuganathan, the Head of the Department and our project supervisor** for his continuous support and motivation throughout our project.

We extend our profound gratitude to **Dr. K.Sethukarasi our Project Coordinator**, who has indeed been a polestar throughout the course of the project. We thank her for giving us full support to complete the project successfully.

Last, but not the least, we take this opportunity to thank all the **staff members** of the Department of Computer Science and Engineering and special mention to **our family, classmates and friends** who offered an unflinching moral support for completion of this project.

|  |  |  |
| --- | --- | --- |
|  |  |  |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | iv |
|  | **LIST OF FIGURES** | viii |
|  | **LIST OF ABBREVIATIONS** | ix |
| 1 | INTRODUCTION | 1 |
| 1.1 | AIM | 1 |
| 1.2 | SYNOPSIS | 1 |
| 2 | SYSTEM STUDY | 2 |
| 2.1 | LITERATURE SURVEY | 2 |
| 2.2 | EXISTING SYSTEM | 3 |
| 2.3 | PROPOSED SYSTEM | 3 |
| 3 | SYSTEM ANALYSIS AND REQUIREMENT SPECIFICATIONS | 4 |
| 3.1 | INTRODUCTION | 4 |
| 3.2 | REQUIREMENT SPECIFICATION | 5 |
| 3.2.1 | HARDWARE REQUIREMENTS | 5 |
| 3.2.2 | SOFTWARE REQUIREMENTS | 5 |
| 3.3 | TECHNOLOGIES USED | 5 |
| 3.3.1 | JAVA | 5 |
| 3.3.2 | DATA MINING | 8 |
| 3.3.3 | APACHE TOMCAT SERVER | 8 |
| 3.3.4 | MYSQL | 9 |
| 4 | SYSTEM DESIGN | 10 |
| 4.1 | SYSTEM ARCHITECTURE | 10 |
| 4.2 | USE CASE DIAGRAM | 11 |
| 4.3 | SEQUENCE DIAGRAM | 12 |
| 4.4 | CLASS DIAGRAM | 13 |
| 4.5 | ACTIVITY DIAGRAM | 14 |
| 4.6 | COLLABORATION DIAGRAM | 15 |
| 4.7 | DATAFLOW DIAGRAM | 15 |
| 5 | SYSTEM DESIGN | 17 |
| 5.1 | MODULES | 18 |
|  | APPENDIX A | 20 |
|  | APPENDIX B | 42 |
|  | REFERENCES | 81 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 3.1 | System Architecture |  |
| 3.2 | Use Case Diagram |  |
| 3.3 | Activity Diagram |  |
| 3.4 | Sequence Diagram |  |
| 3.5 | Collaboration Diagram |  |
| 5.1 | Security authentication |  |

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| 1. | SDLC | Software Development Life Cycle |
| 2. | PSF | Pershing Security Framework |
| 3. | API | Application Program Interface |
| 4. | JDBC | Java Database Connectivity |
| 5. | IDE | Integrated Development Environment |
| 6. | POM | Project Object Model |
| 7. | XML | Extensible Markup Language |
| 8. | JSP | Java server page |
| 9. | EJB | Enterprise JavaBeans |
| 10. | GUI | Graphical User Interface |
|  |  |  |

**CHAPTER 1**

**INTRODUCTION**

The 21st century has brought about many innovations which are mostly in automation to make the lives of people easier and better to live. People look up on technology to artificially sense their needs for reducing their burden and the time taken to complete their jobs. People expect the machine which is an artificial intelligent equipment to sense his / her needs. We would also want to relate relevant day to day information and feedback to get a different perspective to the available information which is useful for interpretation, analysis and evaluation. The artificially intelligent student information network would make all of the above a possibility under a single cloud.

We have already seen how the usual college management works. The attendance, the marks, the syllabus, the time table and all are drawn up or recorded manually. This actually takes some time and is also not perfect. We may not be the perfect judge as to which a tough subject is or who the best teacher to handle crisis is. Further students and faculties would want up to date information in their hands to decide on anything such as the study pattern, the important topics to be covered and all that. So we are bringing out an artificially intelligent application which heavily uses data mining to present the best knowledgeable information about the best teacher for each subject, the rating for the teacher and student likewise, performance of teachers and likewise for students, time table framing, identified tough subjects and question paper framing. The aforementioned information is presented dynamically and not by the administrator but by the Artificial Intelligent agent which utilizes Data Mining.

**1.1 SYSTEM OVERVIEW**

The Artificial Intelligent Agent for the application can be implemented using the Natural Language Processing for voice based input and set of other codes written for arithmetic and probability calculation using the Python Programming Language. In case of a voice input, there are 3 stages which is the syntactic, semantic and pragmatic analysis. This will basically act as a decision support system. For implementing the Data Warehouse for our application, we make use of MySQL and for analysis and interpretation we make use of Python Database API to connect to MySQL database. We utilize the Python Flask API to fetch the results from Python code and publish it in the server at Local host. The new JavaScript Web Speech API makes it easy to add speech recognition to your web pages. This API allows fine control and flexibility over the speech recognition capabilities in Chrome version 25 and later.

**1.2 OBJECTIVE**

Our objective is to store the data easily, analyze and interpret the results effectively and error free without the help of humans and present the result in an interactive and understandable manner requiring lesser or zero key strokes to be performed for giving input.

**CHAPTER 2**

**SYSTEM STUDY**

The system study phase analyses the demerits of the existing system, defines the objectives to be achieved by the new solution and evaluates other solution alternatives.

**2.1 LITERATURE SURVEY**

An extensive study was made on different existing applications which inspired us to come up with this application for the college. Siri in Apple IOS allows you to use your voice to send messages, schedule meetings, place phone calls, and more. But the functionalities of Siri are not specific and in depth . For example, we can’t ask Siri to compute a salary waiver for each faculty or draw up time tables which we regularly do manually in college as a result. Enterprise Resource Tools such as Fedena Pro Support are in their native state and requires human skill to extract vital information from available patterns. The brain child of this innovation has to be the WebFOCUS Business Intelligence platform. It does higher end analytics and presents graphical comparative and vital information. But while this focuses only on analytics and interpretation, our application focuses on the higher level college activities such as periodic calculation of salary of teachers based on college performance and rating and also with the help of this application, we can set up question papers with different range of difficulties. The book on the Natural Language Processing with Python by Steven Bird, Ewan Klein and Edward Loper tells you how Python Programming language is simple and powerful for including excellent functionality for processing linguistic data and presenting it appropriately to the user. We can construct a powerful data analysis tool using MySQL and Mining tool using Python. Further the following papers were a guiding light in our research and development of our project.

***2.1.1 Benefits and introduction to python programming for freshmore students using inexpensive robots***

**Details:** Y. C. Huei, "Benefits and introduction to python programming for freshmore students using inexpensive robots," Teaching, Assessment and Learning (TALE), 2014 International Conference on, Wellington, 2014, pp. 12-17.

**DOI:** 10.1109/TALE.2014.7062611

**Abstract:** Robots were used for teaching at various levels of education. Many research papers reported positive results of using robots to aid students in understanding of programming concepts and writing programs. In this paper, the investigation results had shown that mini-projects using the Finch robot not only enhanced creativity, problem solving skills, and research capability of students but also had helped understood Python programming concepts and writing programs in a collaborative teaching environment at Singapore University of Technology and Design (SUTD). The pedagogical activities of this research were described. Then the experiences and results of findings were shared. The shared knowledge will be beneficial to academics or educators who wish to integrate robots into their programming course, especially in a course of more than 250 students, students with numerous interests, and students with different learning abilities.

***2.1.2 Pattern discovery for text mining***

**Details:** V. Aswini and S. K. Lavanya, "Pattern discovery for text mining" Computation of Power, Energy, Information and Communication (ICCPEIC), 2014 International Conference on, Chennai, 2014, pp. 412-416.

**DOI:** 10.1109/ICCPEIC.2014.6915399

**Abstract:** Text mining can be defined as the art of extracting data from large amount of texts. It allows to structure and categorize the text contents which are initially non organized and heterogeneous. Text mining is an important data mining technique which includes the most successful technique to extract the effective patterns. The paper focuses on developing an efficient method for discovering patterns from the document. In text mining field, pattern mining techniques are used to find text patterns, such as frequent item sets, closed frequent item sets, co-occurring terms. This paper presents an innovative and effective pattern discovery technique which includes the process of pattern evolving and pattern deploying, to improve the effectiveness of using and updating discovered patterns for finding relevant and interesting information.

* 1. **EXISTING SYSTEM**

The existing system does hold a database of related college information wherein the user can view the stored information and can do further analysis. Further users with administrative privileges will be responsible to store the data in the database.

***2.2.1 Disadvantages***

1. It mostly presents you non analyzed and non interpreted raw data.

2. Information has to be viewed in the database with no proper basic User Interface.

3. It is strenuous to add the data in the database in the existing system.

4. More keyboard input and multiple screens involved to view the required and necessary output.

* 1. **PROPOSED SYSTEM**

In the proposed system, a python script handles faster insertion, updating and deletion of data for the administrator requiring limited amount of inputs to be entered. Further it is an easy aid for the administrator to add the proper data. Viewing the populated and the interpreted data is much simpler. It just requires two screens: one to handle input through voice and the other to present the analyzed and interpreted output data in an interactive and user friendly manner without the help of the administrator artificially.

***2.3.1 Advantages***

1. **Faster:** Python makes the overall system much faster than the older systems.

2. **Artificial Interpretation:** The application can artificially interpret what the user speaks and saves the time taken in manual selection of commands.

3. **Simple UI:** Lesser number of screens thus making it a simple Interface for the user to understand.

4. **Automated Data mining:** Automatic analysis and pattern extraction done entirely by the python script with the information a voice input away.

**CHAPTER 3**

**SYSTEM ANALYSIS**

Systems are created to solve problems. One can think of the system approach in an organized way of dealing with a problem. The analysis phase is the second phase of SDLC and to decide if the existing project should go ahead and can be implemented with the resources available. This also includes looking at the existing system to see what it is doing for the organization and how well the system is doing its job. The feasibility of the project is also considered and the group has to ask questions such as

• Can this system be created with the resources we have?

• Will this system significantly improve the organization?

• Does the old system necessarily need to be replaced?

**3.1 PROBLEM STATEMENT**

***3.1.1 Problem Definition***

1. Analyzed and interpreted information are more representative.

2. No dedicated tool and UI for detailed college system information storage and data extraction.

3. The systems that are currently in use lack quality and accuracy.

***3.1.2 Problem Identification***

This project is mainly focused on the efficient use of college records to easily establish a relationship between different entities such as Student, teacher, marks secured and performance to determine multiple information such as department pass percentage, tough subject, teacher salary and many more. There are many colleges who do not have dedicated people to simply handle and play with the department data and so our project does away with the need for such a person.

***3.1.3 Problem Analysis***

The most common solution to the problem would be to assign a proper person to work with the college records. It could not be feasible in all the cases. Also, it is very time consuming and inefficient. So the most probable solution would be an online environment to collect the college records, analyze it and present it interactively on its own. Further we utilize the Natural Language Processing algorithm to get the user input, get the required input and do the necessary required actions. The results are presented based upon a closer understanding of voice input and the collected set of related and identified keywords from the list. Then the Flask API effectively renders the output online through the server in an Interactive manner.

**3.2 SCOPE**

***3.2.1 Product Scope***

The product scope is about the product itself. The product scope defines what the product will look like, how will it work, its features, etc. Our system will play a major role in improving the quality of the college system and it could be available for users throughout the college and with the help of professionals, the accuracy of the system could be improved and made available for use easily. The product has a wider range of scope in other domains as well.

***3.2.2 Business Scope***

Business scope contains Textual Scope Description, list of external actors and their roles, description of business activities. The actors of the system are the user who uses the system and the server which hosts the system itself. The users could be the students or teachers or even administrators who needs suggestions regarding their disease. The server on the other hand, accesses all the data and helps the retrieval of the same regarding the users and use those data to fetch the needed college information.

**3.3 PURPOSE**

The purpose of the project is to provide a standalone server-system to handle the entire college information and to make a seriously capable artificial Intelligent System to understand what the user says and provide him with what he needs.

**3.4 RESOURCES REQUIRED**

Resources are required to carry out the project tasks. They can be people, equipment, facilities, funding required for the completion of a project activity. Here the Artificial application is the information provider and the user or the sometimes even the administrator is the knowledge seeker. The development environment required should have a minimum configuration of i3 core processor. The software resources include the operation system, Python interpreter and other editors.

**3.5 FEASIBILITY STUDY**

The feasibility study is to determine whether the solution is achievable, given the organizations, resources and constrains .By performing feasibility study the scope of the system will be defined completely. Most computer systems are developed to satisfy a known user requirement. Once the decision is made a report is forwarded and is known as feasibility report. Feasibility study is an evaluation of system regarding to its workability, impact on the organizations, ability to meet user needs and effective use of resources. The existing system is pasteurized above and based upon the identification of pitfalls in the existing system and relative inconvenience and difficulties, an outline for a new system is conceptualized and framed. During feasibility analysis, the following primary areas were considered very carefully in our project. The feasibility studies are under 3 contexts:

i. Technical Feasibility

ii. Economic Feasibility

iii. Operational Feasibility

***3.5.1 Technical Feasibility***

1. What resources are available for the given developer system?

2. Is the problem worth solving?

In the proposed system technical feasibility centers around the hardware and the software and to what extent it can support the proposed system. The tools are used to develop the application are the best tools available in the technological scenario and hence it requires efficient and versatile programmers and programming skills. Even though the technical requirements are needed for development of our project, any system which has a web browser and internet connectivity could easily use our product. Hence the proposed system is technically feasible.

***3.5.2 Economic Feasibility***

Economic Feasibility is used for evaluating the effectiveness of the system. The procedure is used for determining the cost and the benefits or the savings that are expected from the system and compare with the cost. In our project all the software obtained are open source and so no cost is incurred apart from awarding the developer for his effort in putting all his programming skills to design such a system. So for sure the proposed system is cost effective than the existing system. Hence the proposed system is economically feasible.

***3.5.3 Operational Feasibility***

The main problem in the developing the new system is getting acceptance from the college to provide their details to the system. Another problem is cooperation from the users because many users are reluctant to operate new systems. The product being developed is hosted on a web server so the users are spared from the tedious job of installing software systems. The developed system is usable by range of people who have basic knowledge to surf the internet. Further the admin is given a basic Python UI to add, update or delete record details. Hence the system is operationally feasible.

**3.6 SYSTEM SPECIFICATION**

***3.6.1 Hardware specifications***

Processor : i3 core processor

Web browser : Google Chrome with version 25 and above

Active Internet Connection : Yes

Input device : Standard Keyboard and Mouse.

Output device : Display Screen

**3.6.2 SOFTWARE SPECIFICATIONS:**

Platform : Windows XP / Ubuntu

Front End : Python Flask

Back End : MySQL and NLTK

**3.7 OTHER REQUIREMENTS**

***3.7.1 Functional Requirements***

Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. The plan for implementing functional requirements is detailed in the system design. A function is described as a set of inputs, the behavior, and outputs.

***3.7.2 Non Functional Requirements***

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. The plan for implementing non-functional requirements is detailed in the system architecture.

**CHAPTER 4**

**SYSTEM DESIGN**

System design is the process or art of defining the hardware and software architecture, components , modules, interfaces and data for a computer system to satisfy specified requirements. One could see it as an application of system theories to computing. Some overlap with the discipline of system analysis appears inevitable. Design tools such as Unified Modeling Language (UML), now addresses some of the issues of computer systems design and interfacing.

**4.1 SYSTEM ARCHITECTURE**

sys

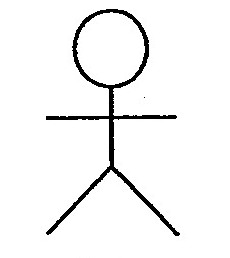
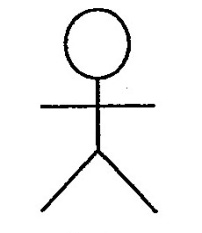
**4.2 USE CASE DIAGRAM**

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

Enter login credentials

Check Details for authenticatiom

Provide access to resources

****

Create and manage sessions

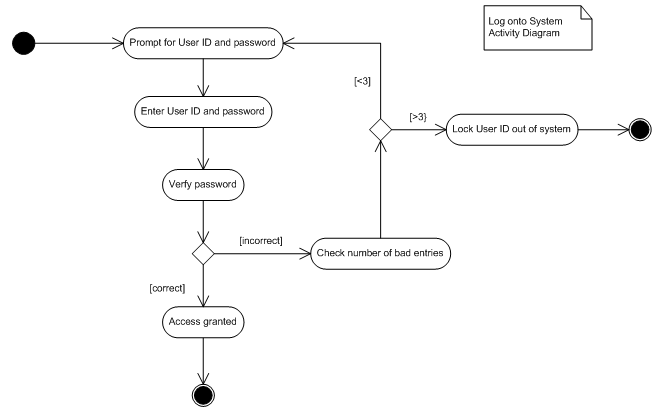
**Administrator End user**

Adding or removing user

Logout

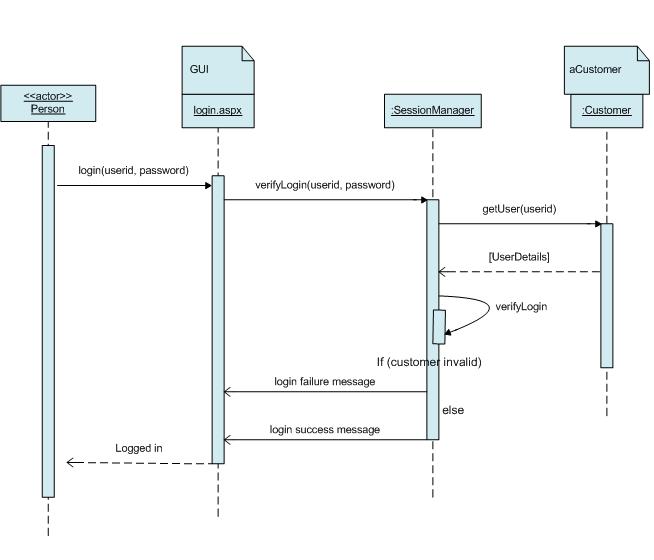
**4.3 ACTIVITY DIAGRAM**

Activity diagram is used to describe the dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity.



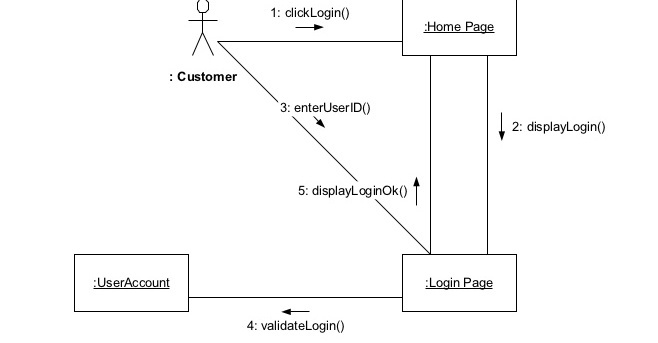
**4.4 SEQUENCE DIAGRAM**

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order.A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.



**4.5 COLLABORATION DIAGRAM**

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles.



**CHAPTER 5**

**TECHNOLOGIES USED**

**5.1 PYTHON**

A useful data type built into Python is the dictionary. Unlike sequences, which are indexed by a range of numbers, dictionaries are indexed by keys, which can be any immutable type; strings and numbers can always be keys. Tuples can be used as keys if they contain only strings, numbers, or tuples; if a tuple contains any mutable object either directly or indirectly, it cannot be used as a key. A pair of braces creates an empty dictionary: {}. Placing a comma-separated list of key: value pairs within the braces adds initial key: value pairs to the dictionary; this is also the way dictionaries are written on output. The main operations on a dictionary are storing a value with some key and extracting the value given the key. It is also possible to delete a key: value pair with DEL. If you store using a key that is already in use, the old value associated with that key is forgotten. It is an error to extract a value using a non-existent key. The keys() method of a dictionary object returns a list of all the keys used in the dictionary, in arbitrary order (if you want it sorted, just apply the sorted() function to it). To check whether a single key is in the dictionary, use the “in” keyword.

Flask library is a light weight web framework in Python. Flask is fun and easy to set up. Flask is considered more Pythonic than Django because Flask web application code is in most cases more explicit. Flask is easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following:

* Importing the API module.
* Acquiring a connection with the database.
* Issuing SQL statements and stored procedures.
* Closing the connection

MySQLdb is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.

The Natural Language toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for the Python programming language. It segments the sentences and tags these according to the part of speech. The program can be designed in the language that is suitable to both the tablet device and the Personal Computers and which properly incorporates the Natural Language Processing Toolkit.

Python3 is installed from Python.org and by using pip installer we install the required wheel files for MySQLdb, flask and nltk.

**5.2 DATA WAREHOUSE**

Enterprise Data warehouse contains all information about subjects. The Data Mart is the departmental subset that focuses on selected subjects. The aforementioned artificial intelligence mechanism forms the Decision Support System (DSS). We perform drill downs and then maintain meta-data for our warehouse. The data about the fee structure, marks and salary forms the operational data. The user guidance support forms the informational data. We make use of the Structured Query Language to add the primitive data and the feedback which is in short the operational data. The informational data is added using Python snippets as mentioned above. We use MySQL Database Management system as our Relational Database Management System for the warehouse.

**5.3 DATA MINING**

Data mining refers to extracting or mining knowledge from large amounts of data. The tasks involved in Data Mining: characterization, discrimination, association, classification, prediction and clustering analysis can be implemented in the Python programming Language. Main objective is the identification of a particular pattern to aid interpretation. This pattern may be either subjective or objective. We include functionalities to perform preprocessing of data to present the final data, the decision tree and also the probability prediction using Bayesian network which is a representation of variables or data and their conditional dependencies. Outliers can be implemented for safety of privacy.

**CHAPTER 6**

**IMPLEMENTATION**

Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in increasing the reliability of the new system. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover and an evaluation of change over methods.

The following working model is explained keeping in mind its implementation using the Python programming language.

**6.1 MODULE DESCRIPTION**

***6.1.1 Database Construction and Deployment (MySQLdb module)***

To work with the data, the priory is to first upload the raw data in the server. What better way than MySQL server to achieve the goal. But sometimes manually entering large number of queries is difficult and time consuming. So for that we provide a module which helps in easy insert, update and delete with the records. This module is a Python code having a set of predefined SQL statements executed based upon the user choices after appropriately establishing a connection. Plus the exceptions are neatly handled in case user enters an inappropriate data.

***6.1.2 Construction of Data Mining module using NLP Algorithm (nltk module)***

In this module we go about in constructing a proper and effective NLP module which does the syntactic, semantic and pragmatic analysis to mine the required data we need. The nltk module in Python provides function to tokenize a whole sentence into a collection of words. And then it aids in Data mining by carrying out a repeated lookup in the predefined lists which acts as the dictionary here. After the required keywords are shortlisted, the appropriate action is executed entirely through the Python code.

***6.1.3 Server Deployment Module (flask module)***

We have obtained the required mined data. But it is important to make the information available to all the users. And since we want all users to have access to it, we deploy it to the server at Local host using Flask library in Python. Python scripts are fast and flask being a light weight framework supports the speed of execution at run time. A flask program creates a module for the application, a set of definitions for the URL mappings of the application and finally a main class which encloses a function to start the application at local host in the server.

**6.2 IMPLEMENTATION DETAILS**

The input is given either in the form of text as an administrator to upload the initial data and voice for other normal users. For the administrator module, the person assigned as admin is required only to input the required data and not the strenuous commands with it as Python provides an easily understandable user friendly interface which can be used by people with basic knowledge in using computers. Once the required data is inserted, deleted or updated, it is ready for mining.

The user or the admin fires a request for mining the desired data. He provides the input as voice recorded or to put in proper words recognized with the help of the Web Speech API in HTML 5. The obtained text is passed as an argument to the URL containing the code for mining the data. This code effectively follows the NLP algorithm. The syntactic analysis is carried out to tokenize the sentences we speak. Then the semantic analysis, groups the necessary tokens to derive the real meaning it intends to put across. Finally the results of voice goes to the Pragmatic Process module which decides what has to be done for the user request. In this case, the predefined lists will check out if such user request can be handled.

Then the control moves to the Data Warehouse containing a large number of databases or Data Marts. The tuples which we require is fetched from appropriate Data Marts. This raw data or tuples has to be preprocessed before analyzing the pattern between different relevant data. In this we perform cleaning, integration, transformation, reduction and finally discretion. The database manipulation and analysis is handled by the MySQL Relational Database Management System.

Now we utilize the database API of Python to connect the MySQL with the Python code. Now we are utilizing Python codes instead of Procedural and trigger statements of MySQL as Python includes easier and more functionalities. The Mining Process takes with the queried data as input and it involves several internal process to obtain the required pattern. Data characterization involves summarization of the general characteristics or features of target data. In Data Discrimination, we perform comparison of the general features of target data with the general features of data from one or a set of contrasting classes. Then we establish a proper association relationship between different relevant datum considered. And then we classify the data based on numerical and probabilistic computations. Finally we cluster the data to include only those which we need to draw up a pattern.

With the help of that pattern (set of function calls)we attempt to trace a decision tree which is a representation of the control flow across different functions to present the needed result.

The result obtained has to rendered in a proper format for which we utilize Bootstrap which is a set of predefined CSS codes for ensuring a neat and responsive design. The output is presented as an AISN response and all the outputs related to the input are presented without any omission.

**6.3 NLP ALGORITHM**

The new algorithm introduced for effective data mining from voice input is the Natural Language Processing Algorithm. There are three major aspects of any natural language understanding theory: The syntax describes the form of the language. It is usually specified by a grammar. The semantics provides the meaning of the utterances or sentences of the language. The pragmatic component explains how the utterances relate to the world. To understand language, an agent should consider more than the sentence; it has to take into account the context of the sentence, the state of the world, the goals of the speaker and the listener, special conventions, and the like.

The efficiency of Natural Language Processing in real world implementation is not exactly 100 percent perfect. But it is just about accurate for the system to understand and generate an algorithm. The easily or mostly solvable problems include Spam Detection, tagging, named entity recognition. Further problems include Sentiment analysis, Co-reference resolution problem, and Word sense disambiguation problem, Parsing, Machine Translation and Information Translation. These problems are solvable if proper research is done and corresponding implementation is included. Some of the problems which are almost impossible to solve are summarization of input and implementation of a dialog system that prompts a related query to the input if the input is ambiguous. Our system simply reacts by saying something like “I do not understand you. Come again”.

**CHAPTER 7**

**RESULTS AND DISCUSSION**

Suppose a department Head wants to set up the time-table for the new semester, here is how the process takes place in the application. The input would be "Please schedule the time-table for CSE". Due to the lack of information at the semantic phase, all available framed timetables will be displayed. From the tokenization, the key words identified were "time-table", "CSE". Now referring to the dictionary, the appropriate case was located by the agent and the input was understandable. Now it fetches the mark table and the passstatus table from the Data Warehouse. After prior preprocessing, we extract the tuples needed for that particular branch and year from the first table and then we extract the previous year results of Computer Science and Engineering alone from the second table. In the mining process we obtain a pattern relating the difficulty of the subjects to the results and subsequently the allocation of number of classes. If some subjects have almost the same level of difficulty, then probabilistic calculations are made which decides which subject should be given the most number of classes after the comparison with other subjects. The data virtual tree is pruned to contain only the subject information and the related pattern. Finally a subject can be allocated at different times based on the conditional probability. Here is a decision tree snippet to illustrate class allocation.

Assigning difficulty,

IF (result > 80) THEN difficulty = "easy";

ELSE IF (60 < result <=80) THEN difficulty = "medium";

ELSE difficulty= "hard";

Same level of Difficulty issue,

IF difficulty= "hard" and probability="yes" THEN

num-class=5;

ELSE IF difficulty= "hard" and probability="no"

THEN num-class=4;

ELSE IF difficulty= "medium" and probability="yes"

THEN num-class=4;

ELSE IF difficulty= "medium" and probability="no"

THEN num-class=3;

ELSE num-class=3;

Another instance would be the salary increment calculator of the faculties based on their average and pass rate. Consider the marks updated periodically, then finally she will have a particular rating at the end of the semester evaluated based on the pass percentage and average. The respective fields are fetched from the Warehouse and the average of the above 2 is calculated during the classification phase of the mining process. Then the following Decision Tree Snippet shows what to do:

average calculation,

current\_average= ( average\_in\_subject + pass\_percentage )/2 ;

salary increment calculation,

increment=( current\_average-previous\_average );

IF increment>0 THEN salary+=increment\*100;

ELSE salary+= 0;

We can easily set a easy or a moderate or a tough question paper by utilizing the Bayesian Network mechanism in the Prediction phase. It is based on Conditional probability relating the probability of maximum or minimum occurrence of question with the probability of occurrence of other similar questions of the same discipline.

**CHAPTER 8**

**CONCLUSION AND FUTURE WORK**

**8.1 CONCLUSION**

This application is generic and can be extended to different areas such as food chains, tourism, defense, agriculture and the like. Future likely addition to the above application would be the sensors. For instance to automatically sense and record the attendance saving further time. This would lead to the creation of a powerful Internet of Things tool which is the need of the hour at the moment.

**8. 2 FUTURE WORK**

Some of the listed plans for future are as below:

1. Implementing a machine learning module again using Python utilizing PyBrain library.

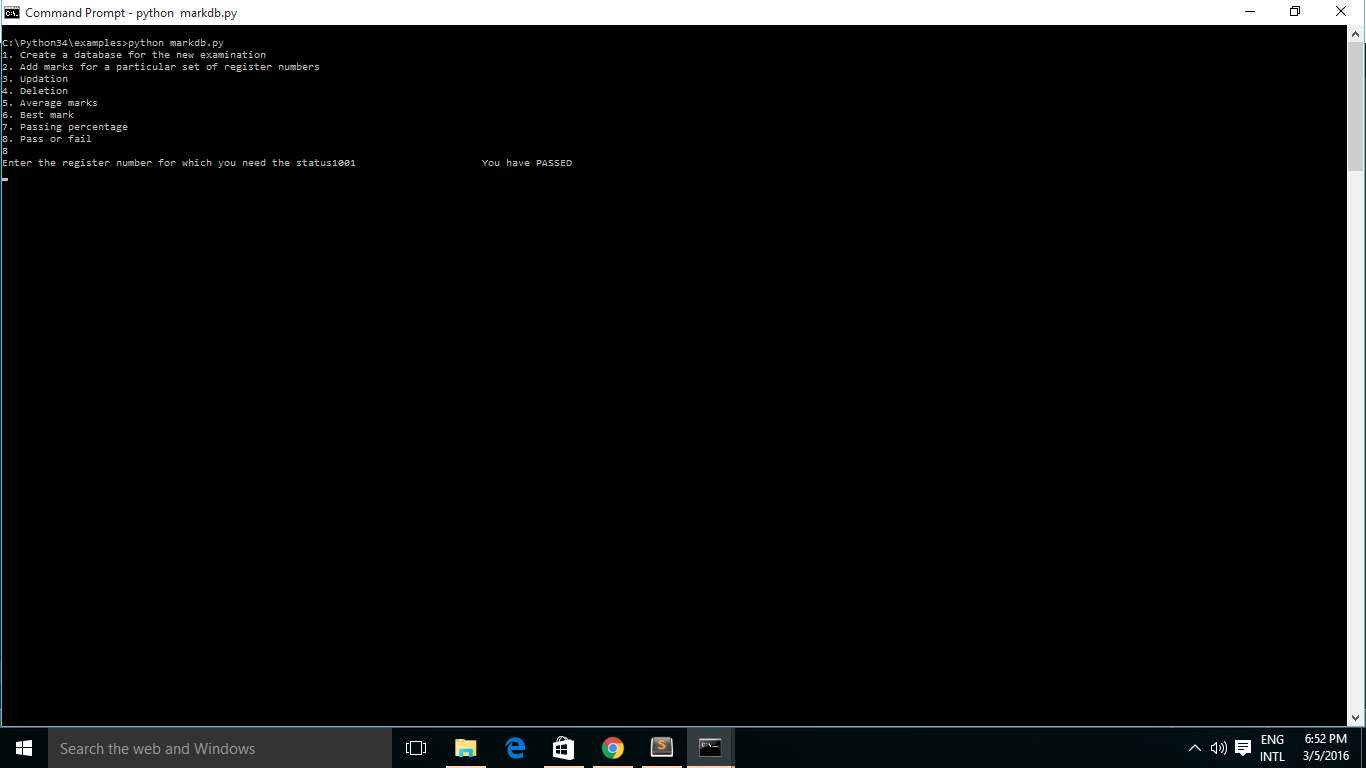
2. Providing a hybrid application portable across multiple mobile devices using Apache Cordova.

3. Making insertion, deletion and updation in admin side completely artificial Intelligent.

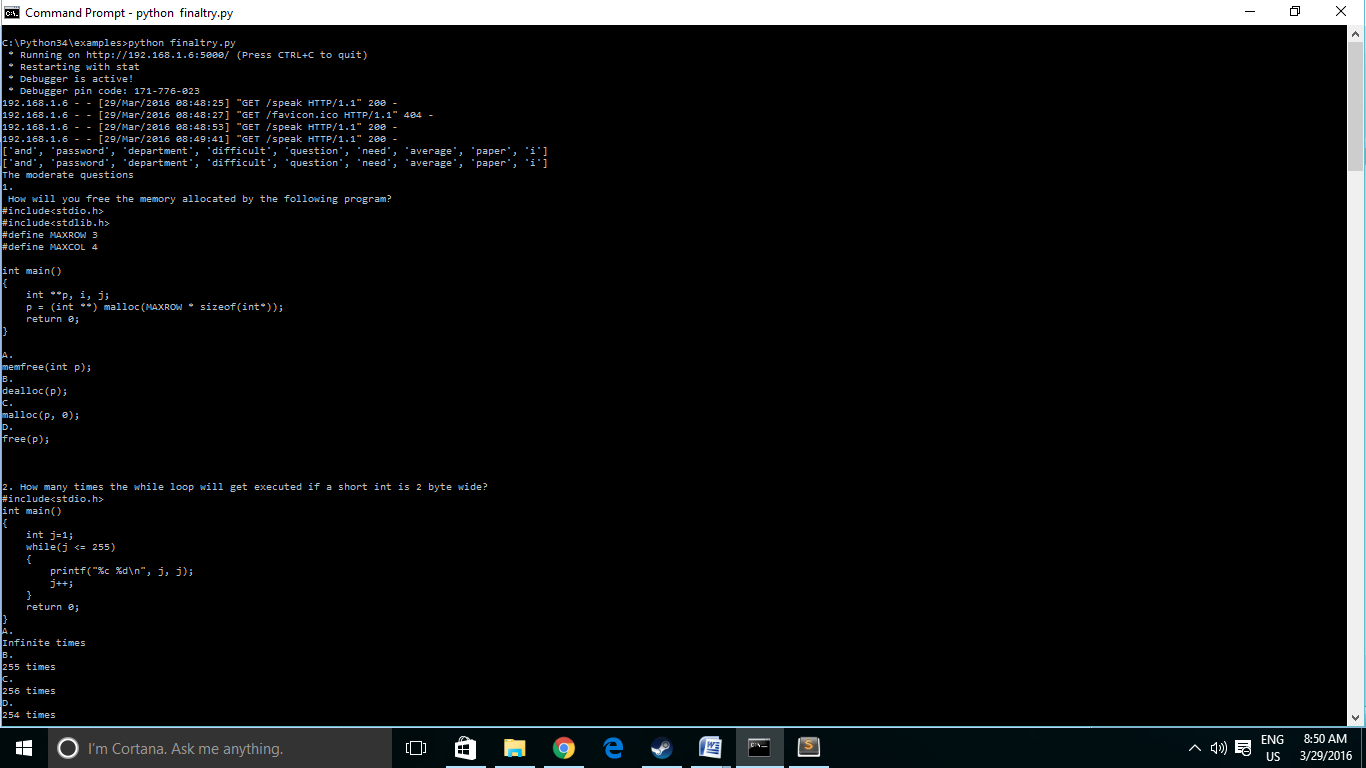
**APPENDIX 1**

**SCREENSHOTS**

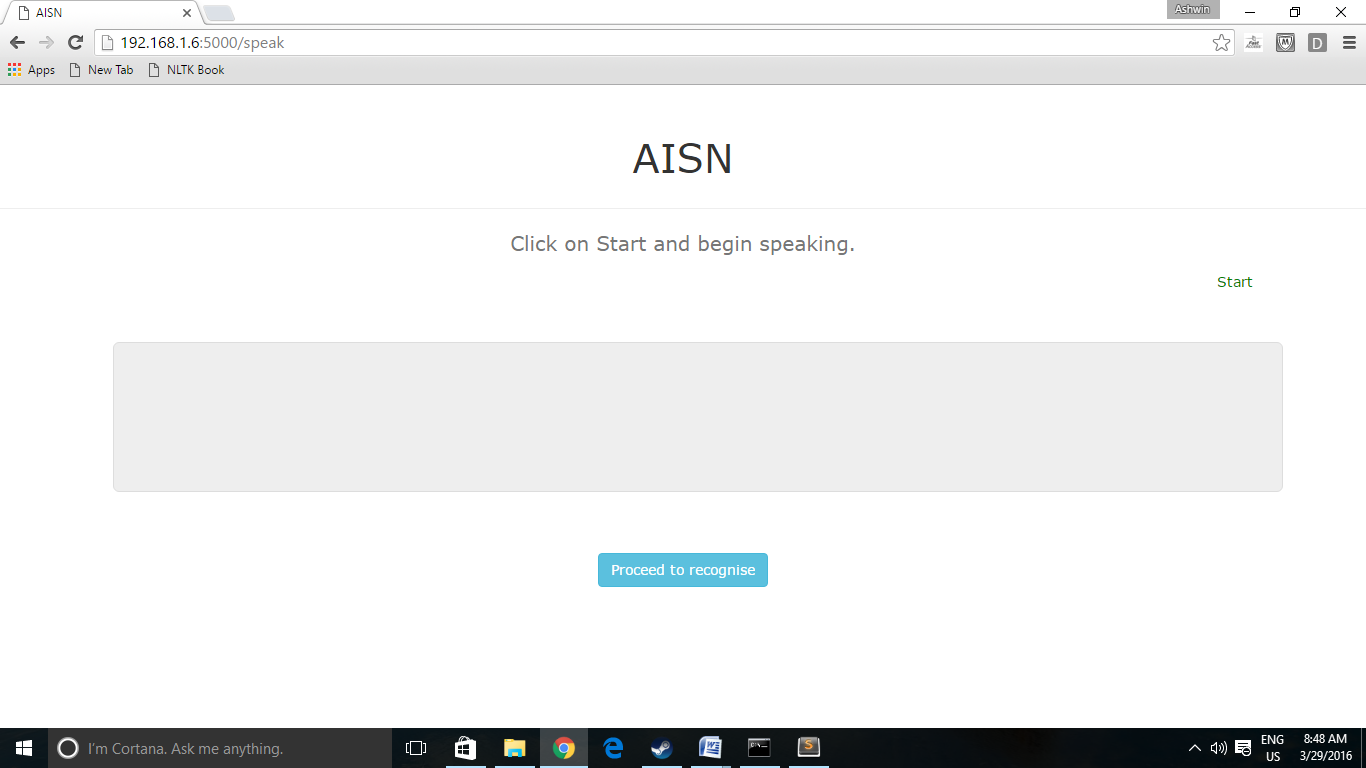
**PYTHON INTERPRETED OUTPUT ACCESSIBLE BY ADMINISTRATOR**



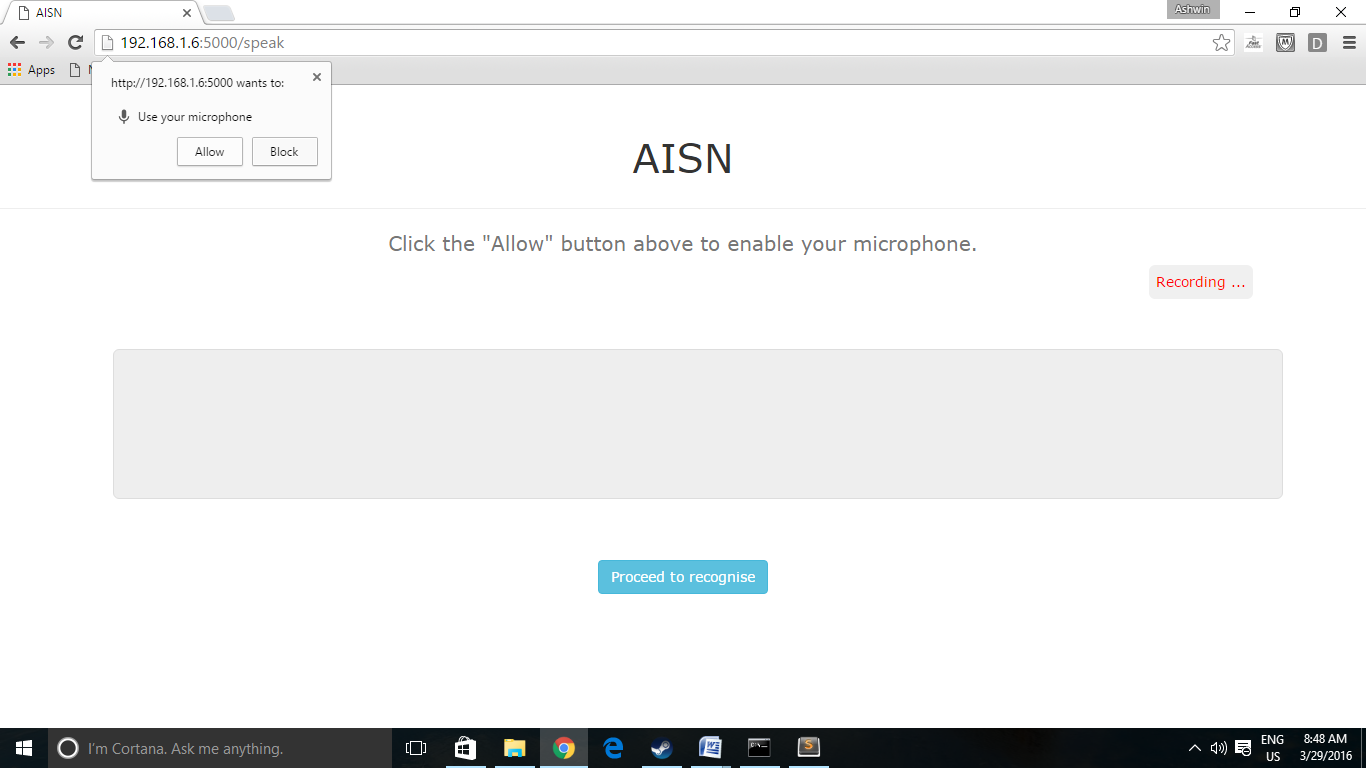
**STARTING UP SERVER AT LOCAL HOST USING FLASH AT THE CONSOLE**

****

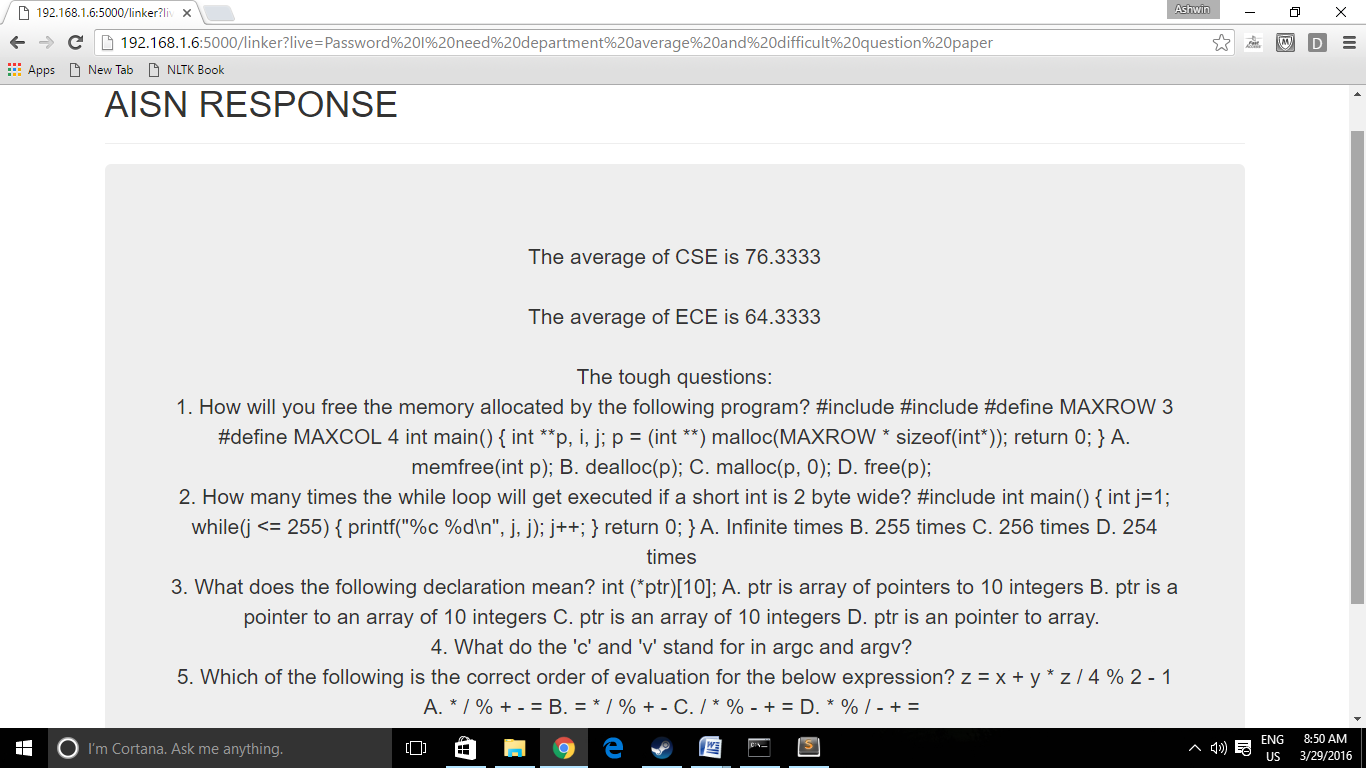
**AISN DASHBOARD**



**GRANTING PERMISSION FOR MICROPHONE OUTPUT**

****

**DATA MINED OUTPUT PRESENTED IN THE DASHBOARD USING NLP ALGORITHM**

****

**APPENDIX 2**

**SAMPLE CODE**

**FLASK CODE TO MINE THE DATA USING NLP ALGORITHM**

1. **from** flask **import** Flask, render\_template, json, request
2. **from** nltk **import** wordpunct\_tokenize
3. **from** flask.ext.mysqldb **import** MySQL
4. **import** random
5. **import** MySQLdb
6. **import** re
7. app = Flask(\_\_name\_\_)
8. mysql = MySQL()
10. # MySQL configurations
11. app.config['MYSQL\_DATABASE\_USER'] = 'root'
12. app.config['MYSQL\_DATABASE\_PASSWORD'] = 'root'
13. app.config['MYSQL\_DATABASE\_DB'] = 'ashwin'
14. app.config['MYSQL\_DATABASE\_HOST'] = 'localhost'
15. mysql.init\_app(app)
17. stringer = ""
18. db = MySQLdb.connect("localhost", "root", "root", "ashwin")
19. cursor = db.cursor()
20. ultiflag = 0
21. flag = 0
22. @app.route("/")
23. **def** main():
24. says = ["Hello !!!", "Hi, How are you?","Hey, How are you doing?","Hi, How have you been?"]
25. result = random.choice(says)
26. **return** "<h1><a href='/home'>"+result+"</a></h1>"
27. @app.route("/home")
28. **def** home():
29. **return** render\_template('mainer.html')
30. @app.route('/final',methods=['POST','GET'])
31. **def** message():
32. **if** request.method == 'POST':
33. app.logger.debug(" entered message function"+ request.form['query'])
34. q = request.form['query']
35. replierlist = wordpunct\_tokenize(q)
36. **for** reply **in** replierlist:
37. **if** reply **in** ["hi","hello","good morning","good afternoon","good evening","good"]:
38. **return** "Well that's good let's get rolling"+"<br>"+"<a href='/speak'> Display </a>"
39. **else**:
40. **return** "<br>hmm anyways let's start"+"<br>"+"<a href='/speak'> Display </a>"
41. @app.route('/test')
42. **def** test():
43. cursor.execute("select \* from student")
44. data = cursor.fetchall()
45. strr = ""
46. **for** row **in** data:
47. strr += row[0]+" "+str(row[1])+""" """
48. **return** strr
49. @app.route('/speak')
50. **def** speaker():
51. **return** render\_template('webspeechdemo.html')
52. @app.route('/linker')
53. **def** linker():
54. live = request.args.get('live')
55. live = live.lower()
56. replierlist = list(set(wordpunct\_tokenize(live)))
57. **print**(replierlist)
58. ultiflag = 0
59. stringer = ""
60. **for** reply **in** replierlist:
61. **if** reply **in** ["hi","hello","good morning","good afternoon","good evening","good"]:
62. ultiflag = 1
63. stringer += "<br>Well that's good let's get rolling<br>"
64. **if** reply **in** ["password"]:
65. **print**(replierlist)
66. ultiflag = 1
67. **for** reply **in** replierlist:
68. namelister = []
69. reglister = []
70. sublister = []
71. deptlister = []
72. teachlister = []
73. cursor.execute("select \* from student")
74. result = cursor.fetchall()
75. **for** row **in** result:
76. namelister.append(row[0])
77. reglister.append(str(row[1]))
78. cursor.execute("select distinct Subject from mark")
79. result = cursor.fetchall()
80. **for** row **in** result:
81. sublister.append(row[0].lower())
82. cursor.execute("select distinct Dept from mark")
83. result = cursor.fetchall()
84. **for** row **in** result:
85. deptlister.append(row[0].lower())
86. cursor.execute("select distinct Empno from mark")
87. result = cursor.fetchall()
88. **for** row **in** result:
89. teachlister.append(str(row[0]))
90. **if** reply **in** ["student","students"]:
91. **for** reply **in** replierlist:
92. **if** reply **in** ["details","detail","information"]:
93. subflag = 0
94. **for** reply **in** replierlist:
95. **if**  reply **in** namelister:
96. subflag = 1
97. cursor.execute("select Name,Regno,Subject,Dept,Marks from student a, mark b where a.Register\_number = b.Regno and a.Name = '%s'" %(reply))
98. result = cursor.fetchall()
99. stringer += "<div class=""table-responsive""><table class=""table table-hover""><thead><tr><th>Name</th><th>Register Number</th><th>Subject</th><th>Department</th><th>Marks</th></tr></thead><tbody>"
100. **for** row **in** result:
101. stringer += "<tr><td>"+row[0].title()+"</td><td>"+str(row[1])+"</td><td>"+row[2]+"</td><td>"+row[3]+"</td><td>"+str(row[4])+"</td></tr>"
102. stringer += "</tbody></table></div><hr>"
103. **if** reply **in** reglister:
104. subflag = 1
105. cursor.execute("select \* from mark where Regno = %d"%(int(reply)))
106. result = cursor.fetchall()
107. stringer += "<div class=""table-responsive""><table class=""table table-hover""><thead><tr><th>Register Number</th><th>Subject</th><th>Department</th><th>Marks</th></tr></thead><tbody>"
108. **for** row **in** result:
109. stringer += "<tr><td>"+str(row[0])+"</td><td>"+row[2]+"</td><td>"+row[3]+"</td><td>"+str(row[4])+"</td></tr>"
110. stringer += "</tbody></table></div><hr>"
111. **if** subflag == 0:
112. cursor.execute("select \* from mark")
113. result = cursor.fetchall()
114. stringer += "<br>The recently published results<br>"
115. stringer += "<div class=""table-responsive""><table class=""table table-hover""><thead><tr><th>Register Number</th><th>Subject</th><th>Department</th><th>Marks</th></tr></thead><tbody>"
116. **for** row **in** result:
117. stringer += "<tr><td>"+str(row[0])+"</td><td>"+row[2]+"</td><td>"+row[3]+"</td><td>"+str(row[4])+"</td></tr>"
118. stringer += "</tbody></table></div><hr>"
119. subflag = 0
120. **if** reply **in** ["average","averages"]:
121. sql = "select Regno, AVG(Marks) from mark group by Regno"
122. cursor.execute(sql)
123. result = cursor.fetchall()
124. **for** row **in** result:
125. stringer += "<br>"+ str(row[0])+ " has overall average of "+ str(row[1])+"<br>"
126. **if** reply **in** ["teacher","teachers","professor"]:
127. **for** reply **in** replierlist:
128. **if** reply **in** ["percentage","percent"]:
129. subfla = 0
130. **for** reply **in** replierlist:
131. **if** reply **in** teachlister:
132. subflag = 1
133. cursor.execute("drop table if exists pass4")
134. cursor.execute("drop table if exists total4")
135. sql = "CREATE table pass4 (pass INT) SELECT COUNT(\*) as pass FROM mark WHERE Marks > 49 and Empno=%d" % (int(reply))
136. cursor.execute(sql)
137. sql = "CREATE table total4 (total INT) SELECT COUNT(\*) as total FROM mark WHERE Empno=%d" %(int(reply))
138. cursor.execute(sql)
139. sql = """SELECT (pass4.pass \* 100)/ (total4.total) FROM pass4, total4"""
140. cursor.execute(sql)
141. result= cursor.fetchall()
142. **for** row **in** result:
143. stringer += "<br>The passing percentage produced by "+reply+" is "+str(row[0])+"<br>"
144. **if** subfla == 0:
145. cursor.execute("drop table if exists pass4")
146. cursor.execute("drop table if exists total4")
147. sql = "CREATE table pass4 (pass INT, Emp INT) SELECT COUNT(\*) as pass, Empno as Emp FROM mark WHERE Marks > 49 GROUP BY Empno"
148. cursor.execute(sql)
149. sql = "CREATE table total4 (total INT, Emp INT) SELECT COUNT(\*) as total, Empno as Emp FROM mark GROUP BY Empno"
150. cursor.execute(sql)
151. sql = """SELECT distinct (pass4.pass \* 100)/ (total4.total), Empno FROM pass4, total4,mark where pass4.Emp = total4.Emp and total4.Emp = mark.Empno"""
152. cursor.execute(sql)
153. result= cursor.fetchall()
154. **for** row **in** result:
155. stringer += "<br>The passing percentage produced by "+str(row[1])+ " is"+str(row[0])+"<br>"
156. subfla = 0
157. **if** reply **in** ["salary", "takeaway","amount","fee"]:
158. subflag = 0
159. **for** reply **in** replierlist:
160. **if** reply **in** teachlister:
161. subflag = 1
162. sql = "select Name, Salary from teacher where Empid=%d" % (int(reply))
163. cursor.execute(sql)
164. result = cursor.fetchall()
165. **for** row **in** result:
166. stringer += "<br> The salary of "+row[0].title()+" is "+str(row[1])+"<br>"
167. **if** subflag == 0:
168. sql = "select Name, Salary from teacher"
169. cursor.execute(sql)
170. result = cursor.fetchall()
171. **for** row **in** result:
172. stringer += "<br> The salary of "+row[0].title()+" is "+str(row[1])+"<br>"
173. subflag = 0
174. **if** reply **in** ["departments", "department"]:
175. flag = 1
176. **for** reply **in** replierlist:
177. **if** reply **in** ["average","averages"]:
178. subflag = 0
179. **for** reply **in** replierlist:
180. **if** reply **in** deptlister:
181. reply = reply.upper()
182. subflag = 1
183. sql = "select Dept, AVG(Marks) from mark group by Dept having Dept = '%s'" % (reply)
184. cursor.execute(sql)
185. result = cursor.fetchall()
186. **for** row **in** result:
187. stringer += "<br>The average of "+row[0]+" is "+str(row[1])+"<br>"
188. **if** subflag == 0:
189. sql = "select Dept, AVG(Marks) from mark group by Dept"
190. cursor.execute(sql)
191. result = cursor.fetchall()
192. **for** row **in** result:
193. stringer += "<br>The average of "+row[0]+" is "+str(row[1])+"<br>"
194. subflag = 0
195. **if** reply **in** ["topper","first"]:
196. subflag = 0
197. **for** reply **in** replierlist:
198. **if** reply **in** deptlister:
199. subflag = 1
200. reply = reply.upper()
201. sql = """drop view if exists totals"""
202. cursor.execute(sql)
203. sql = "CREATE VIEW totals AS SELECT Name AS totname, Register\_number AS totreg, Dept AS totdept,SUM(Marks) AS total FROM student A, mark B where A.Register\_number= B.Regno GROUP BY A.Register\_number HAVING B.Dept = '%s'" % (reply)
204. cursor.execute(sql)
205. sql = "SELECT totname, totreg, total, totdept FROM totals where total = (SELECT MAX(total) from totals group by totdept having totdept= '%s')" % (reply)
206. cursor.execute(sql)
207. result = cursor.fetchall()
208. **for** row **in** result:
209. stringer += "<br>The topper of "+row[3]+" is "+row[0]+"<br>"
210. **if** subflag == 0:
211. sql = """drop view if exists totals"""
212. cursor.execute(sql)
213. sql = "CREATE VIEW totals AS SELECT Name AS totname, Register\_number AS totreg, Dept AS totdept,SUM(Marks) AS total FROM student A, mark B where A.Register\_number= B.Regno GROUP BY A.Register\_number"
214. cursor.execute(sql)
215. **for** dp **in** deptlister:
216. dp = dp.upper()
217. sql = "SELECT totname, totreg, total, totdept FROM totals where total = (SELECT MAX(total) from totals group by totdept having totdept= '%s')" % (dp)
218. cursor.execute(sql)
219. result = cursor.fetchall()
220. **for** row **in** result:
221. stringer += "<br>The topper of "+row[3]+" is "+row[0]+"<br>"
222. subflag = 0
223. **if** reply **in** ["percentage","percent"]:
224. subflag = 0
225. **for** reply **in** replierlist:
226. **print**(deptlister)
227. **if** reply **in** deptlister:
228. subflag = 1
229. reply = reply.upper()
230. **print**(reply)
231. cursor.execute("delete from passstatus")
232. cursor.execute("INSERT into passstatus(Regisno,Dpt) SELECT DISTINCT Regno as Regisno,Dept as Dpt from mark")
233. cursor.execute("drop table if exists pass1")
234. cursor.execute("drop table if exists total1")
235. cursor.execute("UPDATE passstatus INNER JOIN mark ON (Regisno = Regno) set Status = 'PASS' where mark.Marks > 49")
236. cursor.execute("UPDATE passstatus INNER JOIN mark ON (Regisno = Regno)set Status = 'FAIL' where mark.Marks < 50")
237. cursor.execute("CREATE table pass1 (pass INT,dept VARCHAR(50)) SELECT COUNT(\*) as pass, Dpt as dept FROM passstatus WHERE Status = 'PASS' and Dpt = '%s'" %(reply))
238. cursor.execute("CREATE table total1 (pass INT,dept VARCHAR(50)) SELECT COUNT(\*) as pass, Dpt as dept FROM passstatus WHERE Dpt = '%s'" %(reply))
239. cursor.execute("""SELECT (pass1.pass \* 100)/ (total1.pass), pass1.dept FROM pass1, total1 where pass1.dept = total1.dept""")
240. result = cursor.fetchall()
241. **for** row **in** result:
242. stringer += "<br>The department "+ row[1]+" has a pass percentage of "+ str(row[0])+"<br>"
243. **if** subflag == 0:
244. cursor.execute("delete from passstatus")
245. cursor.execute("INSERT into passstatus(Regisno,Dpt) SELECT DISTINCT Regno as Regisno,Dept as Dpt from mark")
246. cursor.execute("drop table if exists pass1")
247. cursor.execute("drop table if exists total1")
248. cursor.execute("UPDATE passstatus INNER JOIN mark ON (Regisno = Regno) set Status = 'PASS' where mark.Marks > 49")
249. cursor.execute("UPDATE passstatus INNER JOIN mark ON (Regisno = Regno)set Status = 'FAIL' where mark.Marks < 50")
250. cursor.execute("CREATE table pass1 (pass INT,dept VARCHAR(50)) SELECT COUNT(\*) as pass, Dpt as dept FROM passstatus WHERE Status = 'PASS' GROUP BY Dpt")
251. cursor.execute("CREATE table total1 (pass INT,dept VARCHAR(50)) SELECT COUNT(\*) as pass, Dpt as dept FROM passstatus GROUP BY dept")
252. cursor.execute("""SELECT (pass1.pass \* 100)/ (total1.pass), pass1.dept FROM pass1, total1 where pass1.dept = total1.dept""")
253. result = cursor.fetchall()
254. **for** row **in** result:
255. stringer += "<br>The department "+ row[1]+" has a pass percentage of "+ str(row[0])+"<br>"
256. subflag = 0
257. **if** reply **in** ["subject","subjects"]:
258. **for** reply **in** replierlist:
259. **if** reply **in** ["average","averages"]:
260. sql = "select Subject, AVG(Marks) from mark group by Subject"
261. cursor.execute(sql)
262. result = cursor.fetchall()
263. **for** row **in** result:
264. stringer += "<br>The average recorded for "+row[0]+" is "+str(row[1])+"<br>"
265. **if** reply **in** ["topper","first","best"]:
266. subflag = 0
267. **for** reply **in** replierlist:
268. **if** reply **in** sublister:
269. subflag = 1
270. reply = reply.upper()
271. sql = "SELECT Name, Register\_number, Marks from student A, mark B where A.Register\_number = B.Regno and B.Subject = '%s' and B.Marks = (select MAX(Marks) from mark group by Subject having Subject= '%s')" % (reply,reply)
272. cursor.execute(sql)
273. result = cursor.fetchall()
274. **for** row **in** result:
275. stringer += "<br>The "+reply+" topper is "+row[0]+" with "+str(row[2])+"<br>"
276. **if** subflag == 0:
277. s = ""
278. **for** sb **in** sublister:
279. sb = sb.upper()
280. sql = "SELECT Name, Register\_number, Subject, Marks from student A, mark B where A.Register\_number = B.Regno and B.Subject = '%s' and B.Marks = (select MAX(Marks) from mark group by Subject having Subject ='%s')" %(sb,sb)
281. cursor.execute(sql)
282. result = cursor.fetchall()
283. **for** row **in** result:
284. s += "<br>The "+row[2]+" topper is "+row[0]+" with "+str(row[3])+"<br>"
285. stringer += s
286. subflag = 0
287. **if** reply **in** ["percent","percentage"]:
288. subflag = 0
289. **for** reply **in** replierlist:
290. **if** reply **in** sublister:
291. subflag = 1
292. reply = reply.upper()
293. cursor.execute("drop table if exists pass")
294. cursor.execute("drop table if exists total")
295. sql = "CREATE table pass (pass INT, sub VARCHAR(50)) SELECT COUNT(\*) as pass, Subject as sub FROM mark WHERE Marks > 49 and Subject='%s'" %(reply)
296. cursor.execute(sql)
297. sql = "CREATE table total (total INT, sub VARCHAR(50)) SELECT COUNT(\*) as total, Subject as sub FROM mark WHERE Subject='%s'" %(reply)
298. cursor.execute(sql)
299. sql = """SELECT (pass.pass \* 100)/ (total.total), pass.sub FROM pass, total where pass.sub = total.sub"""
300. cursor.execute(sql)
301. result= cursor.fetchall()
302. **for** row **in** result:
303. stringer += "<br> The pass percentage of "+row[1]+" is "+str(row[0])+"<br>"
304. **if** subflag == 0:
305. cursor.execute("drop table if exists pass")
306. cursor.execute("drop table if exists total")
307. sql = "CREATE table pass (pass INT, sub VARCHAR(50)) SELECT COUNT(\*) as pass, Subject as sub FROM mark WHERE Marks > 49 GROUP BY Subject"
308. cursor.execute(sql)
309. sql = "CREATE table total (total INT, sub VARCHAR(50)) SELECT COUNT(\*) as total, Subject as sub FROM mark GROUP BY Subject"
310. cursor.execute(sql)
311. sql = """SELECT (pass.pass \* 100)/ (total.total), pass.sub FROM pass, total where pass.sub = total.sub"""
312. cursor.execute(sql)
313. result= cursor.fetchall()
314. **for** row **in** result:
315. stringer += "<br> The pass percentage of "+row[1]+" is "+str(row[0])+"<br>"
316. subflag = 0
317. **if** reply **in** ["teacher","teachers","staff"]:
318. **for** reply **in** replierlist:
319. **if** reply **in** ["rating","scorecard","points","ratings"]:
320. utliflag = 1
321. strr = ""
322. cursor.execute("drop table if exists pass5")
323. cursor.execute("drop table if exists total5")
324. cursor.execute("drop table if exists passer5")
325. cursor.execute("drop table if exists averager5")
326. cursor.execute("drop table if exists subratingtable5")
327. cursor.execute("create table averager5 (Emp VARCHAR(20), Avg INT) select Empno as Emp, AVG(Marks) as Avg from mark group by Empno")
328. sql = "CREATE table pass5 (pass INT, Emp VARCHAR(50)) SELECT COUNT(\*) as pass, Empno as Emp FROM mark WHERE Marks > 49 group by Empno"
329. cursor.execute(sql)
330. cursor.execute("CREATE table total5 (total INT, Emp VARCHAR(50)) SELECT Empno as Emp, COUNT(\*) as total FROM mark group by Empno")
331. cursor.execute("CREATE table passer5(Emp VARCHAR(50), Passavg INT) SELECT pass5.Emp as Emp, (pass5.pass \* 100)/ (total5.total) as Passavg FROM pass5, total5 where pass5.Emp = total5.Emp")
332. cursor.execute("create table subratingtable5 (Emp VARCHAR(20), Point FLOAT) select passer5.Emp as Emp, ((averager5.Avg + passer5.Passavg)/20) as Point from passer5,averager5 where passer5.Emp = averager5.Emp")
333. cursor.execute("UPDATE teacher INNER JOIN subratingtable5 ON (teacher.Empid = subratingtable5.Emp) set teacher.Point = subratingtable5.Point")
334. cursor.execute("UPDATE teacher set Diff= teacher.Point/2")
335. cursor.execute("select a.Name, b.Emp, b.Point from teacher a, subratingtable5 b where a.Empid = b.Emp order by b.Point desc")
336. result = cursor.fetchall()
337. i = 0
338. **for** row **in** result:
339. i+=1
340. **print**("Position %d: %s ( AISNdb Rating: %d )" % (i, row[0], row[2]))
341. strr += "<br>"+"Position "+str(i)+": "+row[0]+" ( AISNdb Rating: "+str(row[2])+" )"
342. stringer += strr
343. **if** reply **in** ["student","students"]:
344. **for** reply **in** replierlist:
345. **if** reply **in** ["pass","passed","passes"]:
346. ultiflag = 1
347. stro = ""
348. cursor.execute("delete from passstatus")
349. cursor.execute("INSERT into passstatus(Regisno,Dpt) SELECT DISTINCT Regno as Regisno,Dept as Dpt from mark")
350. sql = "UPDATE passstatus INNER JOIN mark ON (Regisno = Regno) set Status = 'PASS' where mark.Marks > 49"
351. cursor.execute(sql)
352. sql = "UPDATE passstatus INNER JOIN mark ON (Regisno = Regno)set Status = 'FAIL' where mark.Marks < 50"
353. cursor.execute(sql)
354. flag = 0
355. **for** reply **in** replierlist:
356. namelister = []
357. reglister = []
358. cursor.execute("select Name from student")
359. result = cursor.fetchall()
360. **for** row **in** result:
361. namelister.append(row[0])
362. cursor.execute("select Register\_number from student")
363. result = cursor.fetchall()
364. **for** row **in** result:
365. reglister.append(str(row[0]))
366. **if** reply **in** namelister:
367. flag = 1
368. cursor.execute("select Name, Status from student a, passstatus b where a.Register\_number=b.Regisno and a.Name='%s'" %(reply))
369. result = cursor.fetchall()
370. **for** rober **in** result:
371. stro += "<br>"+(rober[0]).title()+ " has "+rober[1]+"ED<br>"
372. **if** reply **in** reglister:
373. flag = 1
374. cursor.execute("select Regisno, Status from passstatus where Regisno = %d" %(int(reply)))
375. result = cursor.fetchall()
376. **for** rober **in** result:
377. stro += "<br>"+str(rober[0])+" has "+rober[1]+"ED<br>"
378. **if** flag == 0:
379. sql = "SELECT \* FROM passstatus"
380. cursor.execute(sql)
381. result = cursor.fetchall()
382. **for** row **in** result:
383. stro += "<br>"+str(row[0])+" has "+row[2]+"ED<br>"
384. flag = 0
385. stringer += stro
386. **if** reply **in** ["rating","ranking","rank","ranks"]:
387. ultiflag = 1
388. saro = ""
389. cursor.execute("drop table if exists averagerq")
390. cursor.execute("drop table if exists passq")
391. cursor.execute("drop table if exists totalq")
392. cursor.execute("drop table if exists passerq")
393. cursor.execute("drop table if exists subratingtableq")
394. sql = "create table averagerq (Nam VARCHAR(20), Reg INT, Dpt VARCHAR(20), Avg INT) select Name as Nam, Regno as Reg, Dept as Dpt, AVG(Marks) as Avg from student a, mark b where a.Register\_number = b.Regno group by b.Regno"
395. cursor.execute(sql)
396. sql = "CREATE table passq (Reg INT,pass INT) SELECT Regno as Reg,COUNT(\*) as pass FROM mark WHERE Marks > 49 group by Regno"
397. cursor.execute(sql)
398. sql = "CREATE table totalq (Reg INT,total INT) SELECT Regno as Reg,COUNT(\*) as total FROM mark group by Regno"
399. cursor.execute(sql)
400. sql = "CREATE table passerq(Re VARCHAR(50), Passavg INT) SELECT passq.Reg as Re, (passq.pass \* 100)/ (totalq.total) as Passavg FROM passq, totalq where passq.Reg = totalq.Reg"
401. cursor.execute(sql)
402. cursor.execute("create table subratingtableq (Reg VARCHAR(20), Point FLOAT) select passerq.Re as Reg, ((averagerq.Avg + passerq.Passavg)/20) as Point from passerq,averagerq where passerq.Re = averagerq.Reg")
403. namelister = []
404. reglister = []
405. cursor.execute("select Name from student")
406. result = cursor.fetchall()
407. **for** row **in** result:
408. namelister.append(row[0])
409. cursor.execute("select Register\_number from student")
410. result = cursor.fetchall()
411. **for** row **in** result:
412. reglister.append(str(row[0]))
413. flag = 0
414. **for** reply **in** replierlist:
415. **if** reply **in** namelister:
416. flag = 1
417. cursor.execute("select Nam, Point from averagerq, subratingtableq where averagerq.Reg=subratingtableq.Reg and averagerq.Nam = '%s'"% (reply))
418. result = cursor.fetchall()
419. **for** row **in** result:
420. saro += "<br>"+row[0].title() + " has an AISNdb rating of "+ str(row[1])+"<br>"
421. **if** reply **in** reglister:
422. flag = 1
423. cursor.execute("select \* from subratingtableq where Reg=%d"%(int(reply)))
424. result = cursor.fetchall()
425. **for** row **in** result:
426. saro += str(row[0]) + " has an AISNdb rating of "+ str(row[1])+"<br>"
427. **if** flag == 0:
428. cursor.execute("select Nam, Point from averagerq, subratingtableq where averagerq.Reg=subratingtableq.Reg")
429. result= cursor.fetchall()
430. **for** row **in** result:
431. saro += row[0]+ " has an AISNdb rating of " +str(row[1])+"<br>"
432. flag = 0
433. stringer += saro
434. **if** reply **in** ["questions","question paper","question"]:
435. flag = 0
436. **for** reply **in** replierlist:
437. **if** reply **in** ["easy","easier","easiest","simple","low"]:
438. flag = 1
439. ultiflag = 1
440. with open('question\_paper.txt', 'r+') as content\_file:
441. content = content\_file.read()
442. strunger = re.split("\d+[)] ",content)
443. **del** strunger[len(strunger) - 1]
444. **del** strunger[0]
445. substringer = strunger[0:10]
446. **print**("The easy questions")
447. random.shuffle(substringer)
448. q1,q2,q3,q4,q5 = substringer[:5]
449. **print**("1. "+q1)
450. **print**("2. "+q2)
451. **print**("3. "+q3)
452. **print**("4. "+q4)
453. **print**("5. "+q5)
454. stringer += "<br>The easy questions:</br>"+"1. "+q1+"<br>2. "+q2+"<br>3. "+q3+"<br>4. "+q4+"<br>5. "+q5+"<br>"
455. **if** reply **in** ["medium","moderate","ok","middle"]:
456. flag = 1
457. ultiflag = 1
458. with open('question\_paper.txt', 'r+') as content\_file:
459. content = content\_file.read()
460. strunger = re.split("\d+[)] ",content)
461. **del** strunger[len(strunger) - 1]
462. **del** strunger[0]
463. substringer = strunger[6:16]
464. **print**("The moderate questions")
465. random.shuffle(substringer)
466. q1,q2,q3,q4,q5 = substringer[:5]
467. **print**("1. "+q1)
468. **print**("2. "+q2)
469. **print**("3. "+q3)
470. **print**("4. "+q4)
471. **print**("5. "+q5)
472. stringer += "<br>The moderate questions:</br>"+"1. "+q1+"<br>2. "+q2+"<br>3. "+q3+"<br>4. "+q4+"<br>5. "+q5+"<br>"
473. **if** reply **in** ["hard","difficult","high","tough"]:
474. flag = 1
475. ultiflag = 1
476. with open('question\_paper.txt', 'r+') as content\_file:
477. content = content\_file.read()
478. strunger = re.split("\d+[)] ",content)
479. **del** strunger[len(strunger) - 1]
480. **del** strunger[0]
481. substringer = strunger[10:20]
482. **print**("The moderate questions")
483. random.shuffle(substringer)
484. q1,q2,q3,q4,q5 = substringer[:5]
485. **print**("1. "+q1)
486. **print**("2. "+q2)
487. **print**("3. "+q3)
488. **print**("4. "+q4)
489. **print**("5. "+q5)
490. stringer += "<br>The tough questions:</br>"+"1. "+q1+"<br>2. "+q2+"<br>3. "+q3+"<br>4. "+q4+"<br>5. "+q5+"<br>"
491. **if** flag == 0:
492. ultiflag =1
493. stringer += "<br> May I know the difficulty level thalaiva :P"
494. flag = 0
495. **if** reply **in** ["time table","schedule","plan","timetable"]:
496. flag = 0
497. deptlister = []
498. cursor.execute("select distinct Dept from mark")
499. result = cursor.fetchall()
500. **for** row **in** result:
501. deptlister.append(row[0].lower())
502. **for** reply **in** replierlist:
503. ding = ""
504. **if** reply **in** deptlister:
505. ding = ""
506. flag = 1
507. ultiflag = 1
508. sublister = []
509. pointlister = []
510. dailylist = []
511. reply = reply.upper()
512. cursor.execute("drop table if exists averager")
513. cursor.execute("drop table if exists pass3")
514. cursor.execute("drop table if exists total3")
515. cursor.execute("drop table if exists passer")
516. cursor.execute("drop table if exists subratingtable")
517. cursor.execute("create table averager (Sub VARCHAR(20), Avg INT) select Subject as Sub, AVG(Marks) as Avg from mark group by Subject")
518. cursor.execute("CREATE table pass3 (pass INT, Sub VARCHAR(50)) SELECT Subject as Sub, COUNT(\*) as pass FROM mark WHERE Marks > 49 group by Subject")
519. cursor.execute("CREATE table total3 (total INT, Sub VARCHAR(50)) SELECT Subject as Sub, COUNT(\*) as total FROM mark group by Subject")
520. cursor.execute("CREATE table passer(Sub VARCHAR(50), Passavg INT) SELECT pass3.Sub as Sub, (pass3.pass \* 100)/ (total3.total) as Passavg FROM pass3, total3 where pass3.Sub = total3.sub")
521. cursor.execute("create table subratingtable (Sub VARCHAR(20), Point FLOAT) select passer.Sub as Sub, ((averager.Avg + passer.Passavg)/20) as Point from passer,averager where passer.Sub = averager.Sub")
522. sql = "select distinct Sub,Point from subratingtable a,mark b where a.Sub = b.Subject and b.Dept ='%s'" % (reply)
523. cursor.execute(sql)
524. result = cursor.fetchall()
525. stringer += "<div class=""table-responsive""><table class=""table table-hover""><thead><tr><th>DAYS/PERIODS</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr></thead><tbody>"
526. **for** x **in** result:
527. sublister.append(x[0])
528. **for** y **in** result:
529. pointlister.append(y[1])
530. periods = 8
531. **for** i **in** range(0, len(pointlister)):
532. **if** pointlister[i]> 8:
533. count = periods \* 0.2
534. **while**(count>0):
535. dailylist.append(sublister[i])
536. count -= 1
537. **elif** pointlister[i]>= 6 **and** pointlister[i]<=8:
538. count = periods \* 0.3
539. **while**(count>0):
540. dailylist.append(sublister[i])
541. count -= 1
542. **else**:
543. count = periods \* 0.5
544. **while**(count>0):
545. dailylist.append(sublister[i])
546. count -= 1
547. days = 6
548. aliasdays = days
549. **while**(days>0):
550. random.shuffle(dailylist)
551. ding+="<tr>"
552. ding += "<td>"+str(aliasdays - days + 1)+"</td>"
553. **for** x **in** range(0,periods):
554. ding += "<td>"+str(dailylist[x])+"</td>"
555. days -= 1
556. ding += "</tr>"
557. ding += "</tbody></table></div><hr>"
558. stringer += ding
559. **if** flag == 0:
560. ultiflag = 1
561. stringer += "<br> Can you be specific with the department sir<br>"
562. flag =0
563. **if** ultiflag == 0:
564. stringer += "<br> I am not programmed to answer those<br>"
565. ultiflag = 0
566. **return** """<link href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css" rel = "stylesheet">"""+"""<style> a:link, a:visited {
567. padding: 14px 25px;
568. text-align: center;
569. text-decoration: none;
570. display: inline-block;
571. }

574. a:hover, a:active {
575. text-decoration: none;
576. color: hotpink;
577. }
578. </style>"""+"""<div class="container">"""+"""<div class = "page-header">"""+"<h1>AISN RESPONSE</h1>"+"</div>"+"""<div class= "jumbotron">"""+"""<p class="text-center">"""+stringer+"<a href='/speak'> Go back to AISN </a></p>"+"</div>"+"</div>"
579. **if** \_\_name\_\_ == "\_\_main\_\_":
580. app.debug = True
581. app.run(host='192.168.1.6')

**REFERENCES :**

[1] Steven Bird, Ewan Klein and Edward Loper, “Natural Language Processing with Python,” O'REILLY.

[2] Alex Berson and Stephen J. Smith, “ Data Warehousing, Data Mining & OLAP”, Tata McGraw – Hill Edition, Tenth Reprint, 2007.

[3] Jiawei Han and Micheline Kamber, “Data Mining Concepts and Techniques”, Elsevier, 2007.

[4] Y. C. Huei, "Benefits and introduction to python programming for freshmore students using inexpensive robots," Teaching, Assessment and Learning (TALE), 2014 International Conference on, Wellington, 2014, pp. 12-17. doi: 10.1109/TALE.2014.7062611

[5] V. Aswini and S. K. Lavanya, "Pattern discovery for text mining," Computation of Power, Energy, Information and Communication (ICCPEIC), 2014 International Conference on, Chennai, 2014, pp. 412-416. doi: 10.1109/ICCPEIC.2014.6915399