

B.Tech Major Project Report

COT-414

on

AUTOMATED EVALUATION OF SUBJECTIVE QUESTIONS

BY

NAMILAKONDA SRI HARSHA (11610186)

VUTHARKAR SAI SHRAWAN (11610220)

POTHINENI RAKESH KUMAR (11610222)

Group No.:7

Under the Supervision of

Dr. S.K.JAIN, Prof.



DEPARTMENT OF COMPUTER ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

KURUKSHETRA – 136119, HARYANA (INDIA)

May-June, 2020



CERTIFICATE

We, hereby certify that this work which is being presented in this B.Tech. Major Project (COT-414) report entitled “**Automated Evaluation Of Subjective Questions**”, in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Engineering** is an authentic record work of our own work carried out during the period from January, 2020 to May, 2020 under the supervision of Dr.S.K.Jain, Professor, Computer Engineering Department.

The matter presented in this project report has not been submitted for the award of any other degree elsewhere.

Signature of Candidate

NAMILAKONDA SRI HARSHA (11610186)

VUTHARKAR SAI SHRAWAN (11610220)

POTHINENI RAKESH KUMAR (11610222)

This is to certify that the above statements made by the candidates is correct to the best of my knowledge.

Date:

Signature of Supervisor

Dr. S.K. JAIN
(Professor, NITK)

TABLE OF CONTENTS

Section No.	TITLE	Page no.
	Abstract	4
1	Introduction	5
2	Motivation	7
3	Literature Survey	8
	3.1 Automation in Evaluation	8
	3.2 Automated Examination Management System	10
4	Composition of Question-Answer Pairs	11
5	Cosine Similarity Based Evaluation	16
6	Data Flow Diagram	19
	6.1 Level 0 DFD	19
	6.2 Level 1 DFD	20
	6.3 Level 2 DFD	21
7	Implementation Details	22
8	Results & Observations	27
9	Conclusion & Future Plan	35
	References (in IEEE format)	37
APPENDIX:		
A	COMPLETE CONTRIBUTARY SOURCE CODE	39

ABSTRACT

The examination criteria play a crucial role in the current education system and acts as an ultimate means to assess the performance of the student. The entire process of evaluation of student responses in an examination is very essential in order to judge the integration of ideas, conceptual clarity, depth of understanding and the ability to retain the information. It has been observed that almost all the activities performed by any authority responsible for conducting the examination or any examination cell are human intensive. The manual evaluation of answer sheet is a long, tedious and expensive process that consumes the significant amount of time of the instructor. The various fraudulent activities and security threats that might occur in the whole scenario is another point of concern which exhausts lot of manpower and available resources. In order to conquer all these challenges, this project aims to transform the examination process into the one that requires minimal amount of human intervention by building an automated test platform application with proper user authentication mechanism. This application can be used to conduct both objective and subjective based tests. The Question-Answer pairs are composed dynamically from the subjective information provided, through natural language processing techniques. The evaluation of subjective answers and resultant score generation takes place using the cosine similarity metric. The obtained scores are recorded in the relevant data files automatically, to perform the rank-based analysis in future. The grading system has also been incorporated in the application to segregate various responses in a better manner. This flask-based web application, as a whole, helps in the effective and efficient utilization of resources by the examination body and saves the instructor time considerably.

1. INTRODUCTION

1.1 Application Overview

The automation of examination system is an upcoming idea in academia. This web application implements the mentioned idea and also minimizes the manual effort required to setup the entire system for performance assessment of candidates in each and every way plausible. In order to utilize this application for conducting a test, the user must have a data file which contains the relevant subjective information on which the candidates need to be tested. The application handles the entire test process in a procedural manner, right from authentication of candidates for attempting the test, to recording the scores and displaying the result based on the candidate responses. It facilitates the registration process and allows only the registered candidates to use the application. It allows the candidate who logged into the application, to choose between objective test and subjective test. It also allows the candidate to choose between various subjects on which the test is intended to be taken. The candidate can also create a custom test i.e. choose a subject which has not been provided in the list by uploading the data file with relevant subject-based information. Once the candidate chooses the type of test and the subject, the Question-Answer pairs are dynamically composed from the corresponding data file and upon submission of responses by the candidate, the evaluation also gets done without any human intervention.

1.2 Technology Stack

The design of any web application can be primarily realized as "front-end" and "back-end" portions. The front-end portion can be called as 'client side' of web application, which mainly deals with user interaction. The back-end portion can be referred as 'server side' of the web application, which deals with storing and rearranging the data and implements the necessary functionalities to ensure the proper performance of all the operations on client side of the application[8]. Various programming languages used for the front-end implementation of this application are HTML, CSS and JavaScript. The front-end libraries and frameworks used are jQuery and Bootstrap. Similarly, the programming language used for the back-end implementation is Python. A suite of libraries called the Natural Language Toolkit (NLTK) in python, is used to implement the core functionality of the application. A machine learning concept called Cosine Similarity metric also plays an essential role in the application. Flask is used as back-end web framework in the application. Flask is

basically a micro web framework that does not use any specific libraries or tools. It is developed in python and based on the “Pocoo” projects, “Jinja2” and “Werkzeug” [7]

1.3 Usage Scenarios of Application

There are specific usage scenarios where this application is proven to be immensely useful. As the evaluation of the candidate responses is automated based on textual similarity, the theoretical subject based tests and various essay writing contests can extract maximum benefit out of the application. However, the tests which also includes the non-textual based responses, such as visual and tabular representation of information by the candidates, might not find this application to be productive as it currently lacks the processing capabilities of such responses.

2. MOTIVATION

In these days, almost all the institutions avail the technology to enhance the ease in tracking and streamlining various processes such as admissions, examinations, fee payment etc., by making them available online [12]. The usage of online examination system and automated evaluation is beneficial over the traditional approach in the following ways:

a) Secure Examination Process

The traditional examination process gives leeway for malpractices. If the examination is to be conducted over different examination centres, there is high demand for robust security practices, else, the examination paper becomes increasingly vulnerable. These risks can be mitigated by the use of online examination system. Once the set of questions are uploaded in the system, the required number of questions can be randomly selected and displayed, or the order of display of same set of questions can be varied by shuffling them, which reduces the chance of wrongdoing and provides greater flexibility [12].

b) Swift Result Processing and Ranking Analysis

The traditional evaluation process and result compilation consumes significant amount of time and resources of the examiners. Moreover, there are huge manual and administrative tasks overhead involved in such paper-based examinations. Through the automation in evaluation, the results are instantaneously displayed and further rank-based analysis can be easily performed. Thus, it will be immensely helpful in shortlisting and decision-making process [12].

c) Reduction in logistics costs

The logistics costs are significantly high in the paper-based examinations because of numerous factors. Whereas, the online examination system provides scalability to any extent with minimal logistics cost. The automation in the result processing further limits the required cost [12].

Because of the mentioned reasons, the online examination system and automated evaluation becomes a feasible, more affordable choice for conducting examinations. Due to the rapid advancement in the information processing capabilities, it is compatible to organize both objective and subjective based tests in this manner.

3. LITERATURE SURVEY

3.1 Automation in Evaluation

3.1.1 Automated Essay Scoring

The process of usage of computer programs that are specialized to assign grades to the candidate responses, which are written in the form of essays is called Automated Essay Scoring (AES) [6].

It is an application of natural language processing and a model of educational assessment. The main objective of Automated Essay Scoring is to obtain discrete categories, grades for instance, from the large set of textual entities through classification. It can be realized as a concept of statistical classification. The Hewlett foundation held a competition on 'Kaggle' called the Automated Student Assessment Prize (ASAP) in 2012. The main intention of this competition is to demonstrate that scores predicted by AES can be as reliable as scores given human graders or more so. In order to provide the feedback to the candidates, modern AES systems provide try to obtain the scores on essay's quality through the dimensions such as grammaticality, organization, style, coherence, persuasiveness and relevance, as specified in the recent survey. The basic procedure used in the AES systems is to construct a mathematical model that relates to manual scores given to training set of essays on the basis of their quality. It does so by evaluating the features of text in each and every essay, such as the number of subordinate clauses, adherence to prompt's, level of agreement of author and associated reasons etc., [6].

3.1.2 Short Answer Scoring in English Grammar Using Text Similarity Measurement

This work was published in 2018. An automated short answer scoring system has been constructed on the basis of similarity in text, employing a natural language processing approach. The focus of the work is on the "fill-in-the-blanks" type of questions. The answer given by the students must be from 2 to 5 words. The data set basically consists of 240 responses to 10 randomly selected questions. The responses of students are mapped with the ideal responses using the textual similarity techniques, Levenshtein distance and the cosine similarity. In both the techniques, certain values of distance are restricted. Upon comparison with the human grader scoring for assigning full marks, these techniques exhibited high agreement with maximum percentage 94 and 92 for cosine similarity and

Levenshtein distance respectively. Despite the scope being narrowed to secondary school English grammar because of the data sets used for this study, this work assists teacher by easing out the onerous grading task [1].

3.1.3 Knowledge Representation and Answer Evaluation System using Language Processing Algorithm

This work was published at International Conference in 2019. The main motive of the work is to automate the entire evaluation process and provide a comprehensive feedback to the students. In order to achieve this objective, an android application is developed, into which the ideal answer script and all the candidate responses i.e., the corresponding scripts are uploaded. With the help of Tesseract based Object Character Recognition (OCR) technique, the handwritten text in the answer scripts is extracted into the digital text, which is further divided into tokens using Natural Language Toolkit (NLTK). On the basis of comparison of tokenized ideal answer script and tokenized candidate answer script, the marks will be awarded. The marks obtained for a particular candidate is recorded in Firebase database at the back-end and a detailed performance-based report is generated, specifying the areas where the candidate lacks and the level of understanding of the subject [4].

3.1.4 Automated Essay Scoring with Ontology based on Text Mining and NLTK tools

This work was published in 2018. The aim of this work is to automatically generate the ontology related to the domain of essays through the applied natural language processing algorithms using Natural Language Tool Kit and OntoGen (a data driven and semi-automatic ontology editor) that enhances the grading process for essays by the teacher [3].

3.1.5 Automated Grading System Using Natural Language Processing

This work was published in 2018. It presents a solution for automated evaluation of theoretical subjects using the concepts of Natural language processing. As the orderly comparison of keywords, turns out to be fairly inefficient considering the different ways of expressing the same answer, this work focusses on extraction of synonymous words related to the domain using ontology. In this manner, the conceptual coverage, right word combinations, presence of keywords or the corresponding synonymous words can be checked making the entire process of evaluation, holistic. Eventually, an automated grading

system is obtained with very little error rate which can be comparable to differential error rate in various manual corrections [2].

3.2 Automated Examination Management System

3.2.1 Automated System for Management of Examination Processes

This work was published in 2019. It deals with the automation of monotonous, routine manual activities performed during the examination. The extent of automation done in this work is not only beneficial to students but also to the examiners. It helps in the utilization of workforce and infrastructural resources optimally by the examination cell. The various technologies used during the development of the system are servlets, HTML, CSS, Bootstrap, MySQL and amazon web services (to avail the cloud computing services and make the entire system reliable by storing the data in the cloud for future use). In terms of system design, there are different modules for students and administrators. The module for students basically helps in registering for an examination and receiving the notification of the upcoming examinations through email. Whereas, the module for administrators performs various management related tasks such as generation of seating arrangement for the semester examinations, appointing the examiners for practical and oral tests, sending notices to the students regarding the upcoming examinations and keeping track of inventory required and resources available for the exams [5].

4.COMPOSITION OF QUESTION-ANSWER PAIRS

Natural Language Processing (NLP) plays a massive role in the transformation of free data or unstructured data into structured data suitable for analysis [11]. In this application, the approaches of Natural language processing are used in the composition of Question-Answer pairs and after obtaining the candidate responses, a textual similarity technique called cosine similarity is used in the evaluation process and score generation consequently.

In this application, the approach used for the formation of Question-Answer pairs is different for both objective and subjective based tests. However, the extraction of noun phrases from the subjective information provided, must precede these approaches.

4.1 Extraction of noun phrases from text

The stepwise algorithm to extract noun-phrases from the given text is presented as follows:

Begin

Step 1: Define a grammar to chunk keywords i.e., noun phrases in this context. Let us label the corresponding chunk as "CHUNK"

Step 2: Specify the behaviour of the parser based on the defined grammar using set of regular expressions

Step 3: Tokenize the text provided as input into sentences

Step 4: Consider a sentence in the group of sentences, upon consideration of all the sentences, go to step 13

Step 5: Tokenize the sentence into words and perform parts of speech tagging for each word

Step 6: Assign the syntactic structure to the tagged words through parsing, with the parser obtained in step 2

Step 7: Consider a sub-tree i.e., a constituent in the syntax tree, upon consideration of all sub-trees, go to step 12

Step 8: If the subtree is not labelled as "CHUNK", go to step 11

Step 9: Traverse the sub-tree and form a noun phase by combining all the words present as the leaf nodes

Step 10: Append the obtained noun-phrase to the list which collect all the noun phrases in the text

Step 11: Go to step 7

Step 12: Go to step 4

Step 13: Stop

End

```
noun_phrases = list()
grammar = r"""
    CHUNK: {<NN>+<IN|DT>*<NN>+}
           {<NN>+<IN|DT>*<NNP>+}
           {<NNP>+<NNS>+}
    """
chunker = nltk.RegexpParser(grammar)
tokens = nltk.word_tokenize(sentence)
pos_tokens = nltk.tag.pos_tag(tokens)
tree = chunker.parse(pos_tokens)

for subtree in tree.subtrees():
    if subtree.label() == "CHUNK":
        temp = ""
        for sub in subtree:
            temp += sub[0]
            temp += " "
        temp = temp.strip()
        noun_phrases.append(temp)
```

Figure 4.1 Snippet of code that performs noun phrases extraction

Elucidation of methods used

1) nltk.RegexpParser(grammar)

The "RegexpParser" function takes the grammar defined for chunking a piece of information as a parameter and specify the behaviour of the parser using a specific set of regular expressions. Chunking can be defined as a process of amalgamation of individual pieces of information into large and meaningful units [11].

2) nltk.sent_tokenize(text)

The "sent_tokenize" function takes the piece of text provided as input as the parameter. It avails an instance of "PunktSentenceTokenizer" from natural language tool kit (nltk). It is already well trained to detect the characters which represents the end of a sentence [11].

3) nltk.word_tokenize(sentence)

The "word_tokenize" method is used in the extraction of tokens from a character string. It takes a sentence as input and actually returns the syllables of words present in that sentence. At most, two syllables are present in a word [11].

4) nltk.tag.pos_tag(tokens)

The "pos_tag" function tags a particular token with the corresponding "parts of speech" using an inbuilt POS tagger. It takes a group of tokens that needs to be POS tagged as a parameter [11]. The description of few POS tags specified in the given grammar is, NNS: plural noun, DT: determiner, IN: subordinating conjunction or preposition, NNP: proper noun, NN: singular noun.

5) chuker.parse(pos_tokens)

The "parse" method takes the tokens which are "parts of speech" tagged as the parameter, and build a syntax tree like structure which depicts the relationship between various tokens on the basis of syntax rules specified using the parser [11].

6) tree.subtrees()

This method is used to generate all the sub-trees of the syntax tree formed. sometimes, the generation of sub-trees can also be restricting by using a filter expression as parameter [11].

7) subtree.label()

This method is used to obtain the label attached to that particular sub-tree in the syntax tree. This label classifies the group of tokens that corresponds to a particular sub-tree. In most of the cases, this method is used to extract the corresponding sub-trees which matches with the label present in the grammar defined [11].

4.2 Composition of Question-Answer pairs for subjective test

Upon obtaining the list of noun phrases (key words) from the given subjective information, a question is formed by adding a predefined question format to a noun phrase and corresponding answer is formed by extracting all the sentences that contain this noun phrase. A Question-Answer pair in approach, is stored in the form of key-value mapping where "key" can be realized as a noun phrase extracted and "value" can be realized as one or more sentences with the presence of corresponding key word in them.

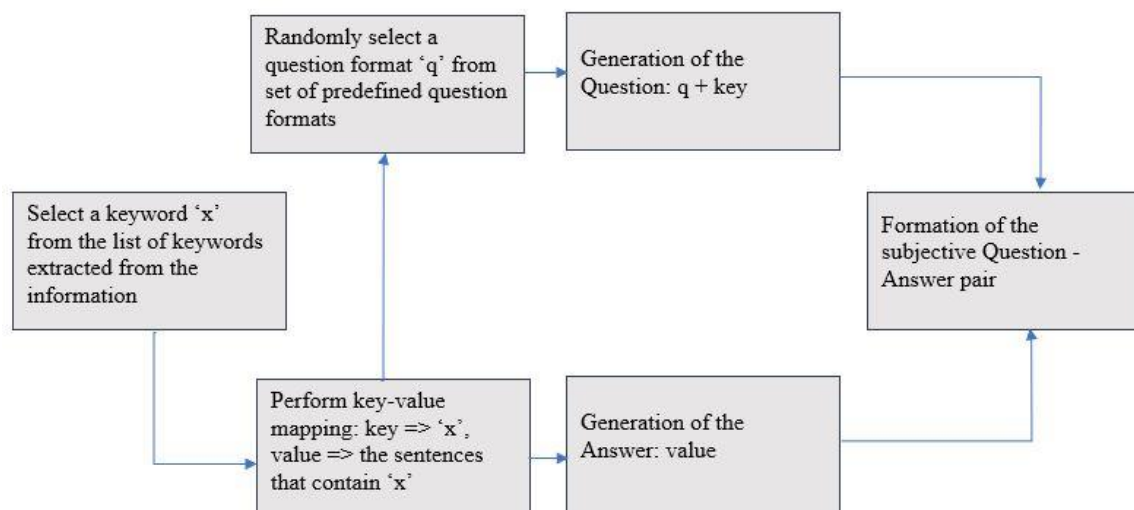


Figure 4.2 Block diagram of composition of a subjective Question-Answer pair

4.3 Composition of Question-Answer pairs for objective test

The objective questions in this application are typically "fill-in-the-blanks" type of questions, with the expected candidate response of at most two words. Initially, each and every sentence in the information provided is examined. The sentences which are beginning with an adverb hardly qualifies to be an objective question. Similarly, the sentences with number of words less than specific value (implementation dependent) can also be made unfit to form a question. The noun-phrases are extracted from the sentence which meets these conditions. If the sentence does not contain any noun phrase i.e., key word, a sensible objective question cannot be formed, therefore these sentences are not used in framing questions. An extracted noun phrase is selected from the sentence. If the number of words in that phrase are more than two, the last two words or blanked out in order to be aligned with expected of candidate response or else entire noun phrase is blanked out. In this manner, a valid sentence present in the given information is recompiled to form an objective question and a noun phrase which is partially or completely blanked out in the corresponding sentence can be considered as an answer.

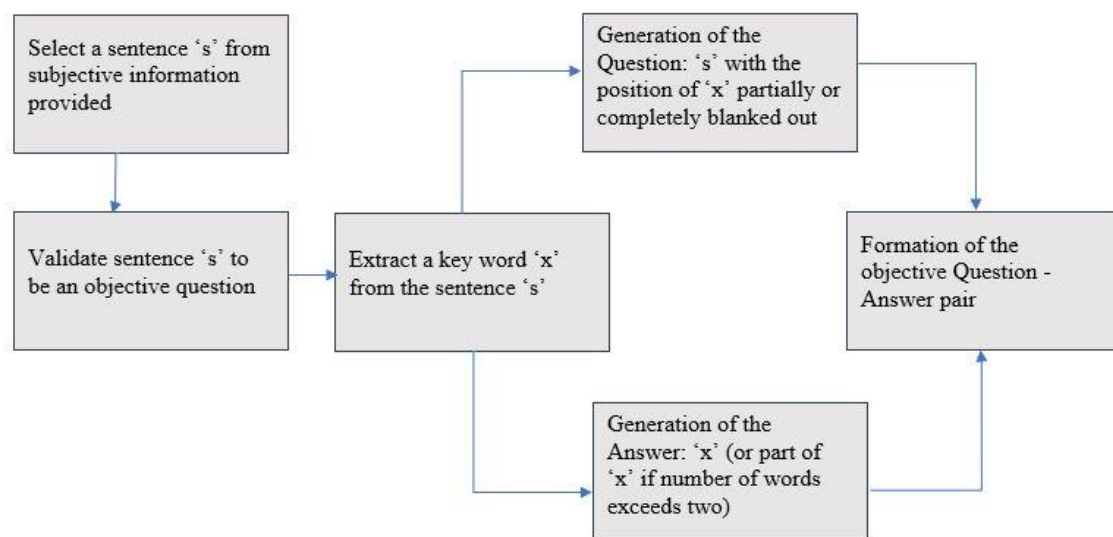


Figure 4.3 Block diagram of composition of an objective Question-Answer pair

5. COSINE SIMILARITY BASED EVALUATION

5.1 Cosine Similarity

Cosine similarity is a technique to obtain textual similarity between two documents irrespective of the size of corresponding documents. This technique overcomes the inherent flaw in other techniques which measure the similarity between documents, such as, "counting the number of common words" in both the documents under consideration (this approach fails to produce accurate similarity index because, if the size of both the documents is increased, then the number of words which are going to be common in these documents also increases even though the topics presented are totally unrelated) and "Euclidean distance" approach (It fails because of the consideration of size of documents in the similarity measurement, even though both the documents under consideration may vary in size, there are fair amount of chances that they present the same topic and they are textually similar) [9].

Mathematically, cosine similarity metric considers two vectors and measures the cosine of angle between them when projected in a multi-dimensional space. In the context of textual similarity, these vectors correspond to the arrays formed from words present in the documents. Each word in the document is considered as a dimension in space and this metric calculates the orientation(angle) of documents. It does not consider the magnitude, as in Euclidean distance approach. For instance, if the word "computer" appears 20 times in document 'A' and 50 times in document 'B', these documents are still oriented closer i.e., exhibit high similarity. In the same scenario, the Euclidean distance is more because of variability in size [9].

Cosine similarity formula

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

Where 'A_i' and 'B_i' are the components of vector 'A' and 'B' respectively [9].

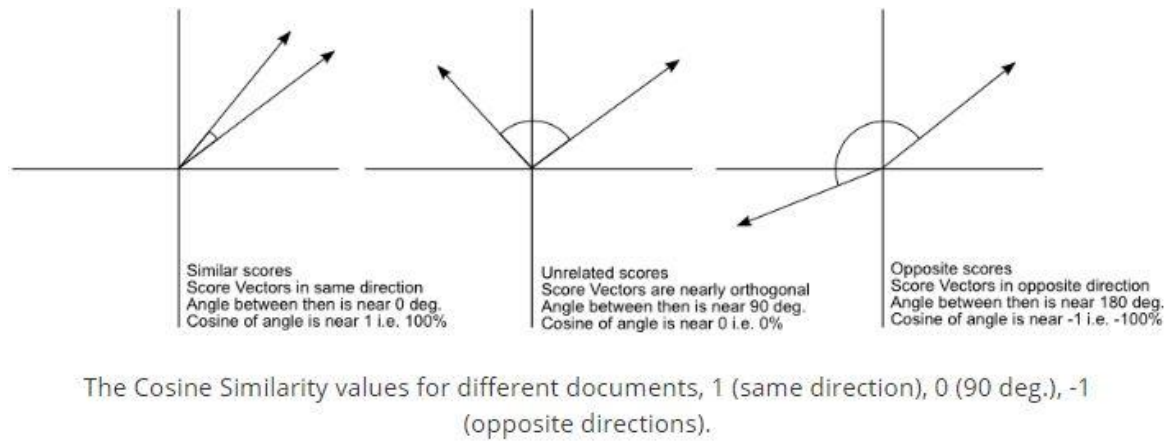


Figure 5.1 Variation in the cosine similarity for different documents (image source:[10])

5.2 Evaluation of Subjective Answers

In this application, for the objective test, evaluation is done through case insensitive comparison of candidate and ideal responses. Evaluation for the subjective test is done using the concept of cosine similarity and the respective metric. The step-wise algorithm in order measure the similarity between candidate and ideal responses, score generation is presented as follows:

Begin

Step 1: Obtain the user answer and original answer

Step 2: Tokenize both the answers initially into sentences, then into words and store them in "user answer list" and "original answer list" accordingly

Step 3: Remove the "stop words" from both these lists.

Step 4: Form a "main list" by combining both these lists such that "main list" contains the words present in "user answer list" and "original answer list"

Step 5: Consider a list among "user answer list" and "original answer list" in order form the corresponding vector to calculate cosine similarity, if both the lists are considered, go to step 11

Step 6: Declare an empty vector to basically store the boolean values '0' and '1'

Step 7: Consider each word 'x' in the "main list", if all the words are considered, go to step 10

Step 8: if the word 'x' is present in the selected list ("user answer list" or "original answer list"), append '1' to the vector, else append '0' to the vector

Step 9: Go to step 7

Step 10: Go to step 5

Step 11: Find the dot product between both the vectors, one obtained from "user answer list" and "main list", the other obtained from "original answer list" and "main list"

Step 12: Calculate the magnitude of both the vectors

Step 13: Compute the cosine similarity between both the vectors. The similarity index value obtained ranges between 0 and 1

Step 14: Obtain the score in the scale from 0 to 100, by multiplying the similarity index with 100 and return the score obtained

End

6. DATA FLOW DIAGRAMS

Notation followed: Yourdon and Peter Coad Notation

External entity to the system: User

Data stores used in the system:

- a) Subject files
- b) Details of registered users
- c) User activity log

6.1 Level 0 DFD

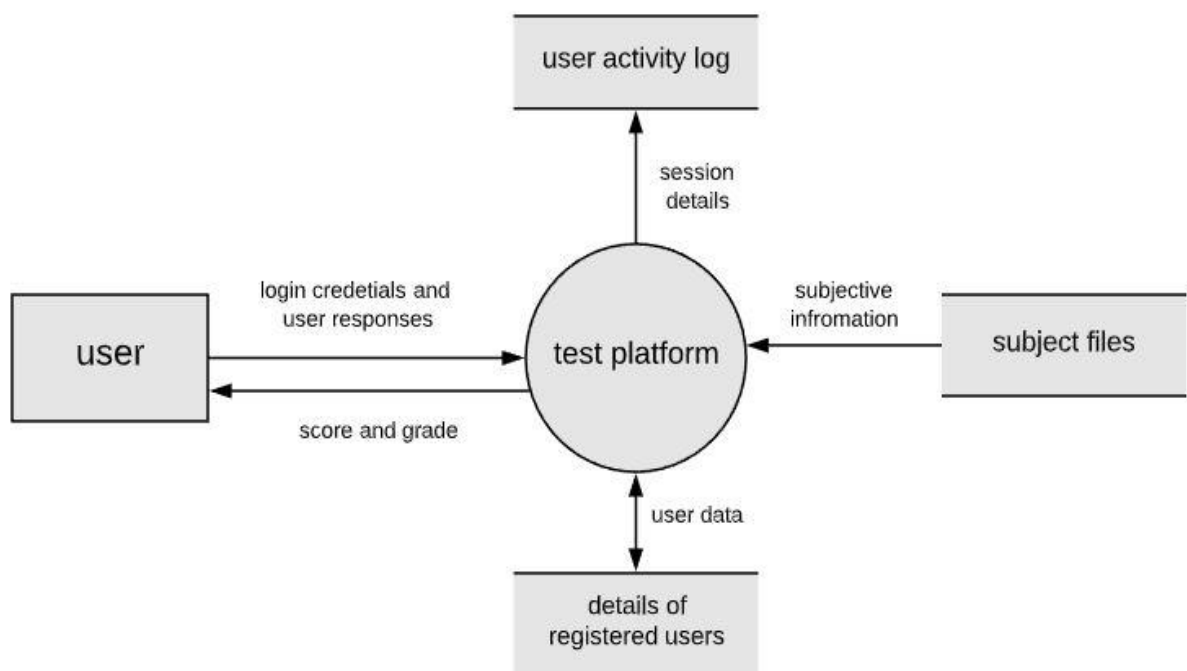


Figure 6.1 Data flow diagram of Level-0

6.2 Level 1 DFD

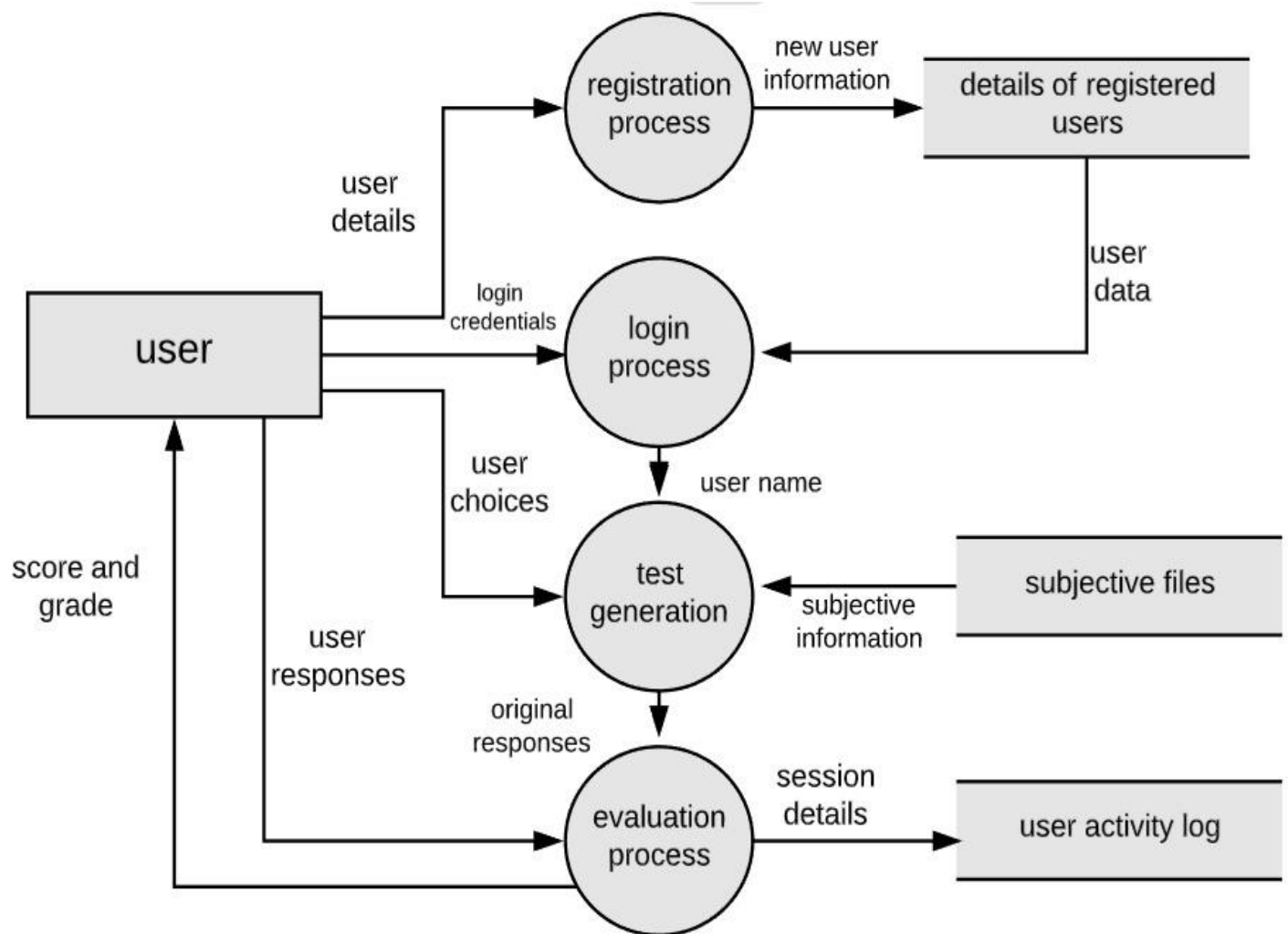


Figure 6.2 Data flow diagram of Level-1

6.3 Level 2 DFD

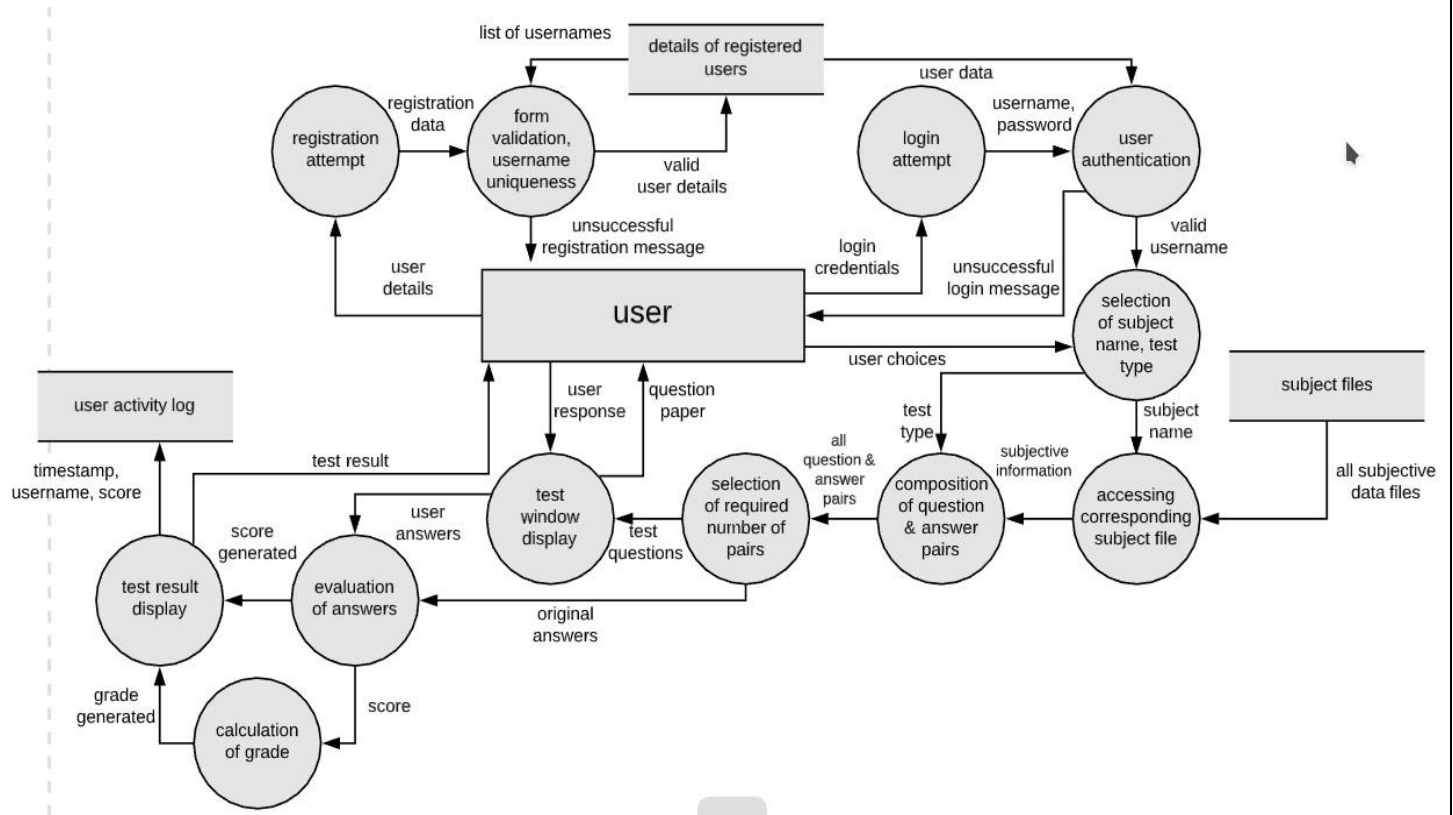


Figure 6.3 Data flow diagram of Level-2

7. IMPLEMENTATION DETAILS

7.1 Data Setup

7.1.1 Subject files

In this application, text files related to three subjects are used in order to exercise the functionalities of the application. They are: a) Software Testing b) Databases and c) Machine learning. These files contain relevant subjective information such as important definitions and overview of the concepts. Apart from these three files, custom files can also be used that contain the information upon which test should be generated.

7.1.2 User details file

This application contains a '.csv'(comma separated values) file to store the details of all the registered users in the application. For each user the following details are present in the file: Full name, username and password. This file is useful for the validation of registration form and user authentication.

7.1.3 Log files

There are two log files are present in the application that contain '.csv' (comma separated values) extension. One file stores the session details when user attempts a subjective test and the other one stores the session details during objective test. The session details stored in these log files are: Date, Month, Year, username, subject and score. The subject will be named "custom" if the user exercises the application with a subject file other than the three files mentioned. Two separate files are used to store the same kind of information in order to facilitate the further analysis based on test type.

7.2 User Interaction (Front-end implementation)

7.2.1 Logging into the system

In order to get logged in to the system, a user needs to provide respective username and password. The Login form that helps a registered user to attempt the test is present in the home page of the application. Besides the login form, the home page also consists of an

option in order to register into system. If user fails to login, a message called "invalid user name and password" is displayed.

7.2.2 User registration

The registration form instructs user to enter the Full name, username (which must be unique among all the registered users in the application), password (which the user need to enter twice for the confirmation). Upon the submission of registration form, the form data is validated and an acknowledgement message informing the status of registration is displayed based on the form data.

7.2.3 User decision making

Once the user authentication is successful, the user can choose the test type, by selecting any of the two options that corresponds to objective and subjective tests. In order to choose the subject upon which test is intended to be given, there are four options for the user, three options are related to the three subject files (software testing, databases, machine learning) which are already present in the application. The fourth option is to be selected when the user wants to attempt the test on a different information file and this file must to be uploaded to the application through this form. Whereas, the application knows the path to obtain the other three files.

7.2.4 Test window

For the objective test, the test window contains three fill-in-the-blanks type of questions which expects one word or two words response from the user for each question. For the subjective test, the test window displays two questions and there is no application-specific constraint to on the length of user response for each question.

7.2.5 Result display

The result of the test is displayed to user upon submission of user responses. The results page in the application contains the score obtained (which is rounded off to three decimal places), grade (on the basis of generated score) and a message which informs the user about the status of operation that attempts to store the session information in log files.

7.3 Behind-the-scenes activities (Back-end implementation)

7.3.1 Login form validation

Upon submission of login form by the user, the application receives the login credentials. It accesses the file that contain the user details and iterates over all the entries. For each iteration, it compares the corresponding username present in the entry with the username received, if both the usernames match, then the corresponding passwords are compared in the similar manner. If the login credentials match with an entry in the file, then application redirects to the page that allows the user to make choices for test generation. If the login credentials found out to be invalid, the page that displays relevant status message is rendered.

7.3.2 Registration form validation

Upon submission of the registration form, the application ensures that no field is empty, then it checks whether the password which was entered twice in the form is exactly same. If the form data is failed to meet any of these conditions, the relevant alert message is sent to user to modify the content and resubmit the form. Upon meeting these conditions, the application accesses the file that contains user details and iterates over all the entries to access respective 'username' field. If the username received in form data is proven to be unique, then a new entry is created with the required details, received as form data. Thus, the registration process be successful. If the username received is not unique, the registration process won't be successful. Nevertheless, the page that displays status of registration is rendered.

7.3.3 Test generation

On the basis of subject chosen by the user, the application obtains corresponding text file either from a specific path declared or from the decision-making form (where the user uploaded the required file previously). Based on type of test chosen, the obtained file is sent to the module that composes Question-Answer pairs either of objective type or subjective type. The key-value mapping is performed in the process of test generation. The required number of Question-Answer pairs are randomly chosen i.e., three for objective type and two for subjective type. The page that contains the test window is rendered along with the questions present in these pairs.

7.3.4 Evaluation process

The ideal responses are obtained by the application during the process of test generation. The user responses are obtained upon the completion of test (i.e., submission of data from the "test window" by the user). In case of subjective test, for each question, the ideal response and user response are sent to the module that computes the similarity between both the responses and generates score on the scale of 0 to 100. Then the scores for these two questions are summed up and divided by 2 for obtaining the final score. In case of objective test, for each question, case insensitive comparison is performed between user response and ideal response. If both the responses for a question, turned out to be same, then the score obtained is 100 or else it is 0. The scores obtained for these three questions are added together and divided by 3 in order to obtain the final score. The final score generated elucidates the performance of the student in that test. In order, to categorize various performances, the grading system has been incorporated. The grades computed for the final score obtained can be understood as follows:

Score	Grade
0-39	FAIL
40-54	D
55-64	C
65-74	B
75-84	A
85-100	A+

Table 7.1 Scores and the corresponding grades for result generation

After generation of final score and corresponding grade, the process that stores the session details of user is invoked. Later on, the result page is rendered.

7.3.5 User session information storage

During the termination of user session, the application tries to store the mentioned details related to the session. The status of this operation is also displayed along with generated score and grade once the application renders the page that displays the test result.

8. RESULTS AND OBSERVATIONS

8.1 The home page of the application which contains login form and option to register into the application

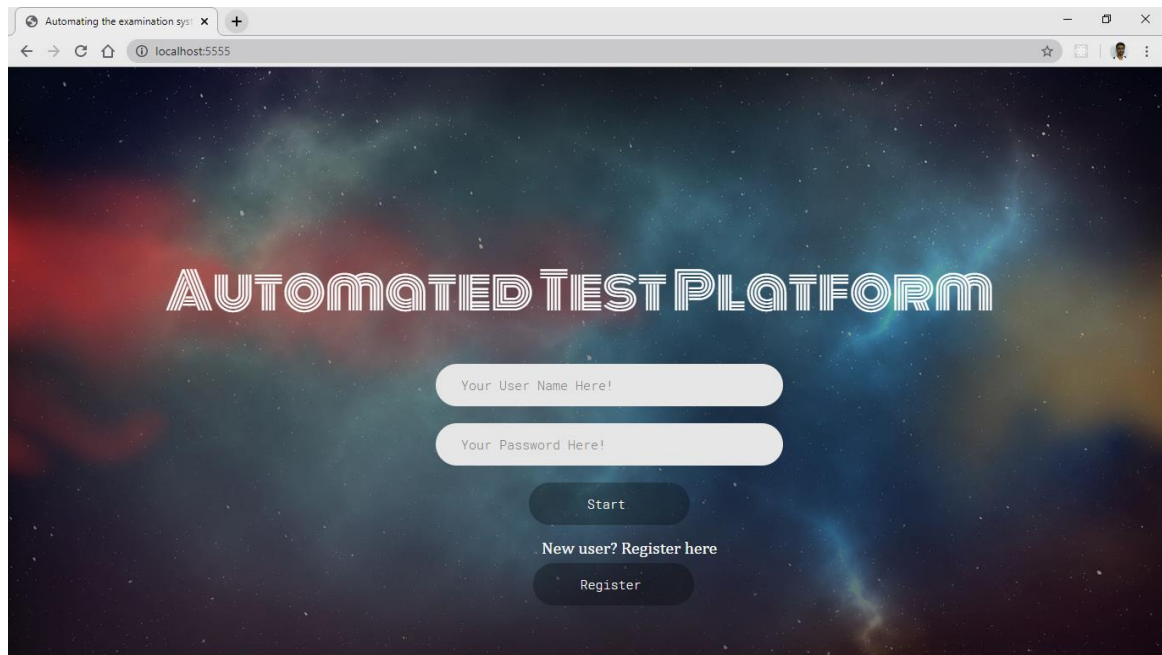


Figure 8.1 Snippet of the home page

8.2 When the login credentials provided by user is found to be invalid

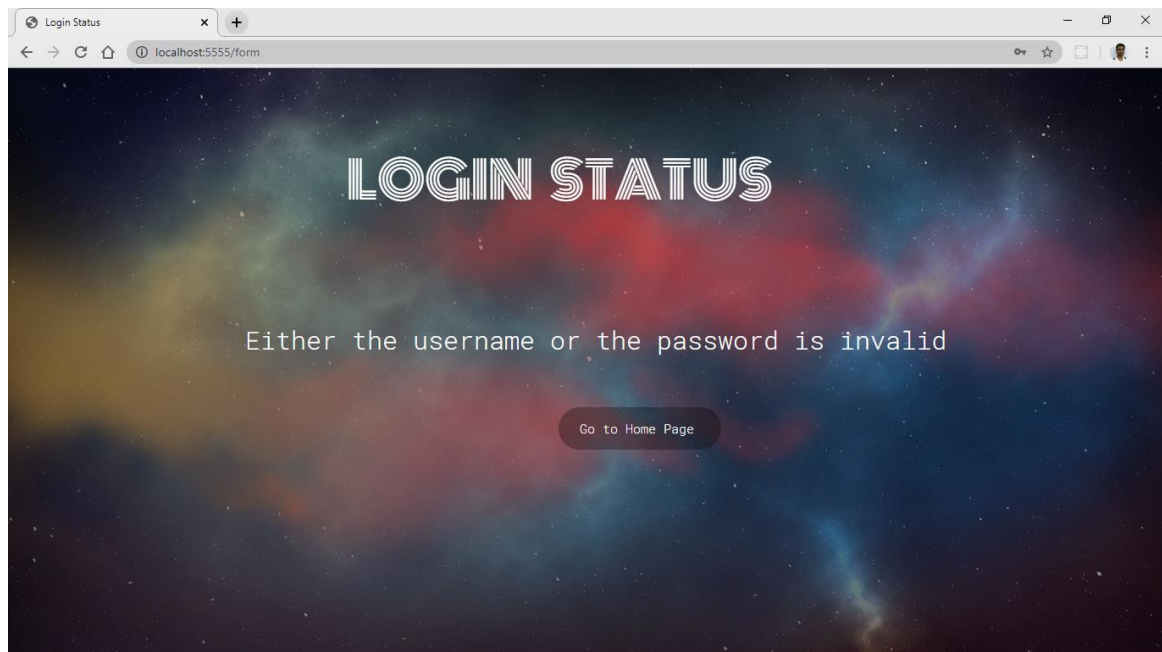
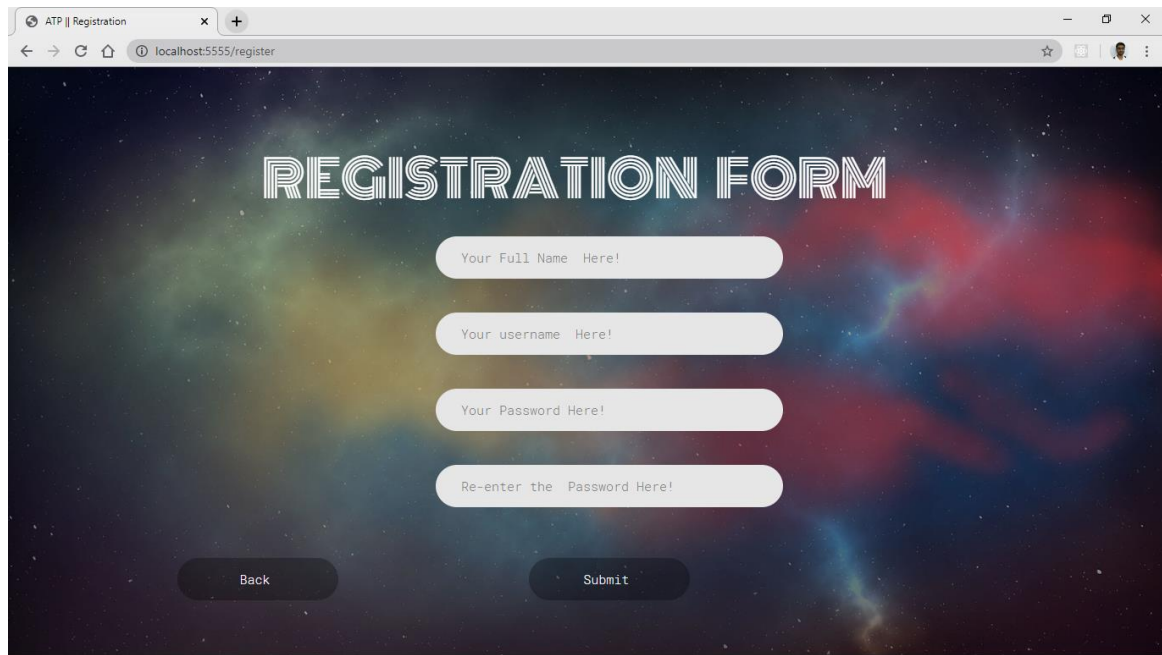


Figure 8.2 Snippet of login status

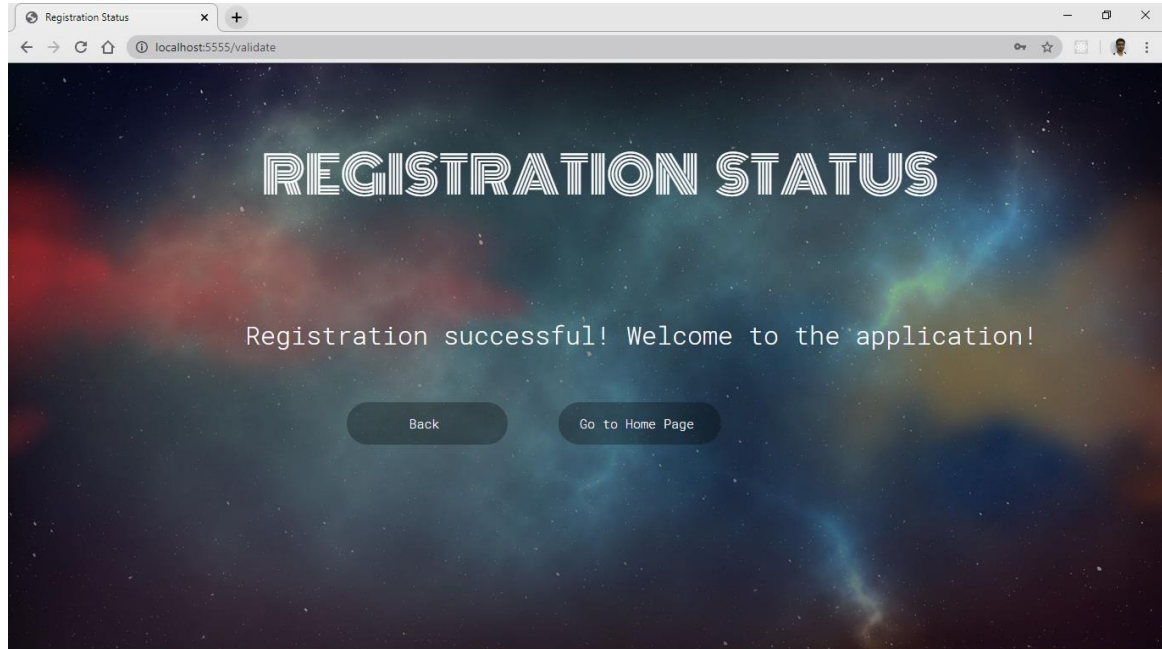
8.3 Registration form for the application



A screenshot of a web browser showing a registration form. The browser's address bar displays 'localhost:5555/register'. The page has a dark, space-themed background with colorful nebulae. The title 'REGISTRATION FORM' is centered at the top in a large, white, outlined font. Below the title are four input fields, each with a light gray background and rounded corners. The first field is labeled 'Your Full Name Here!', the second 'Your username Here!', the third 'Your Password Here!', and the fourth 'Re-enter the Password Here!'. At the bottom of the form are two dark gray buttons with white text: 'Back' on the left and 'Submit' on the right.

Figure 8.3 Snippet of registration form

8.4 When the user registration is successful



A screenshot of a web browser showing the successful registration status page. The browser's address bar displays 'localhost:5555/validate'. The page has the same dark, space-themed background as the registration form. The title 'REGISTRATION STATUS' is centered at the top in a large, white, outlined font. Below the title, the text 'Registration successful! Welcome to the application!' is displayed in a white, monospaced font. At the bottom of the page are two dark gray buttons with white text: 'Back' on the left and 'Go to Home Page' on the right.

Figure 8.4 Snippet of successful registration status

8.5 When the user tries to register into the application with existing user name

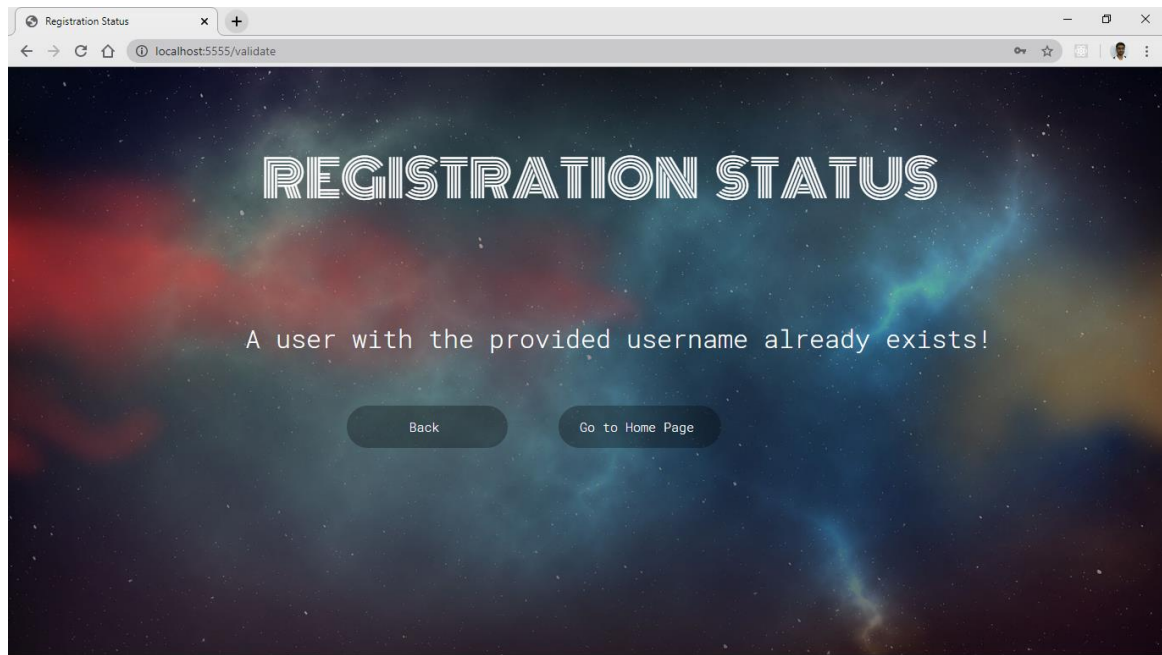


Figure 8.5 Snippet of unsuccessful registration status

8.6 Page through which user performs decision making for test generation

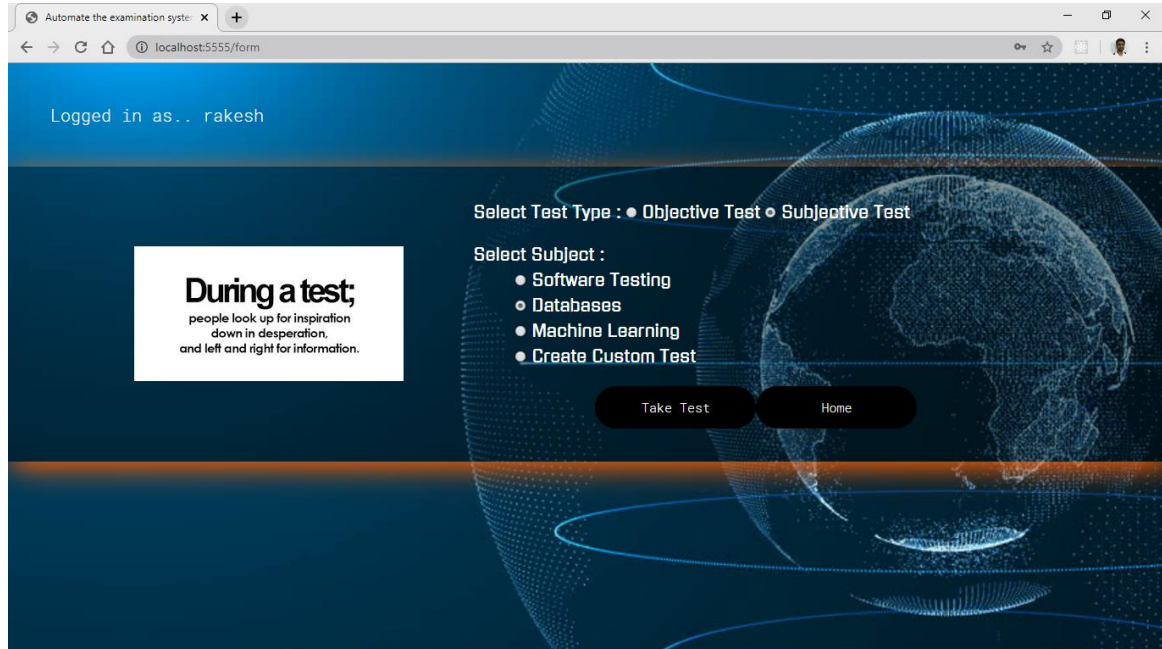


Figure 8.6 Snippet of form to generate test

8.7 Uploading the relevant text file when user chooses to create a custom test

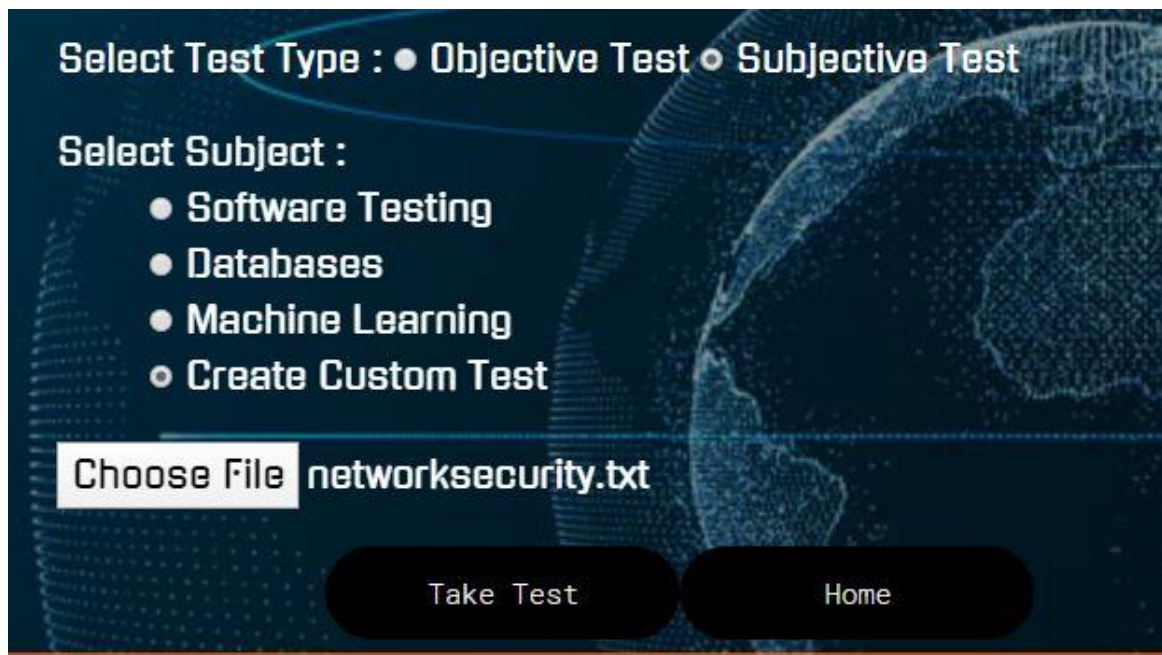


Figure 8.7 Snippet to create custom test

8.8 An instance of subjective test

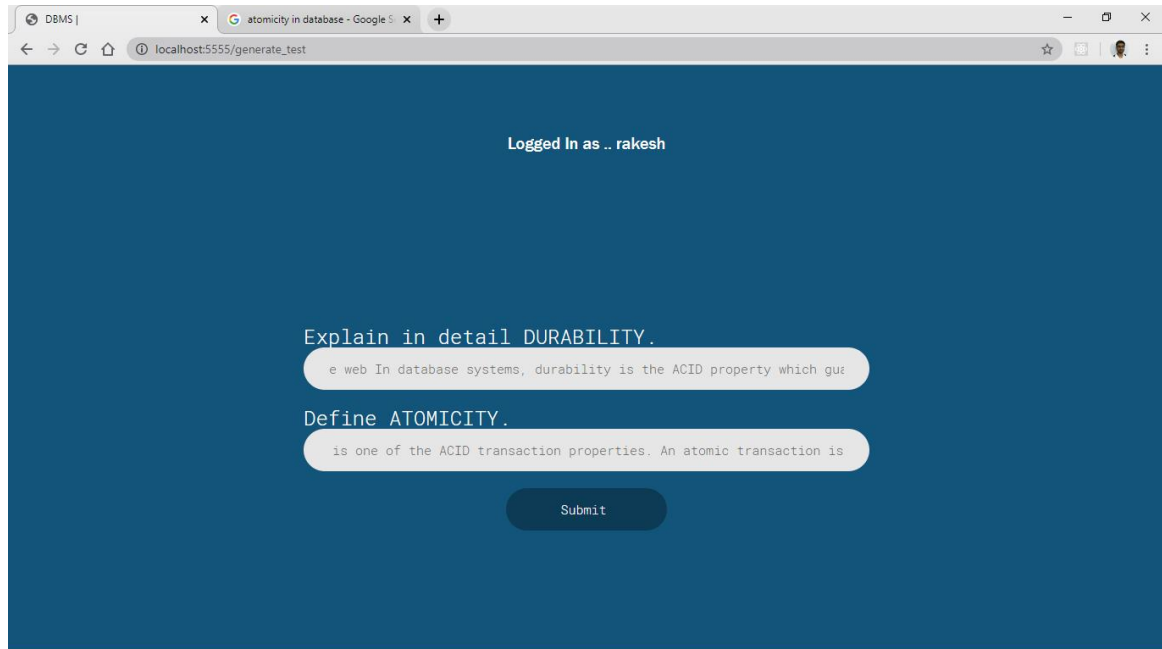


Figure 8.8 Snippet of subjective test

8.9 An instance of objective test

Isolation is typically defined at database level as a property that defines how/when the changes made by one operation is visible to others. Isolation is one of the -----.

----- in producing information, which is based on facts.

In database systems, durability is the ----- property which guarantees that transactions that have committed will survive permanently.

Submit

Figure 8.9 Snippet of objective test

8.10 The result pages that display different grades based on the obtained score

Software Testing | Analysis

YOUR SCORE : 92.864%

Your score has been saved! GRADE: A+

Log Out

Figure 8.10.1 Snippet of result with A+ grade

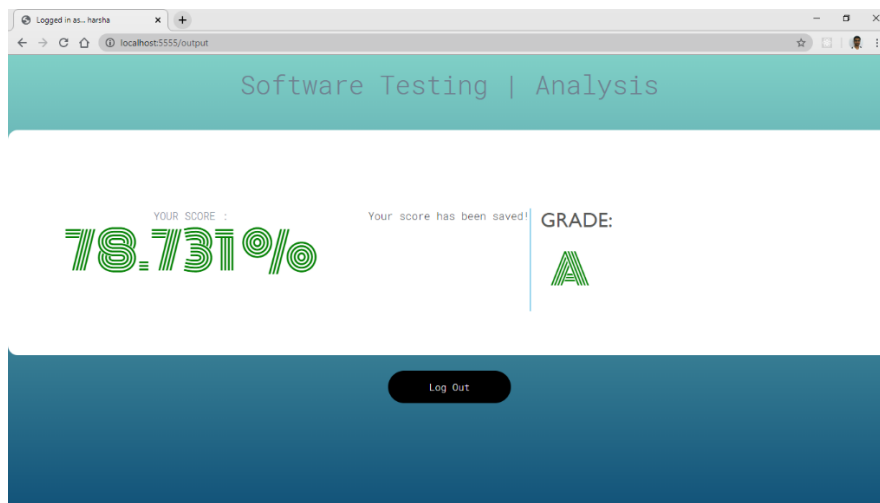


Figure 8.10.2 Snippet of result with A grade

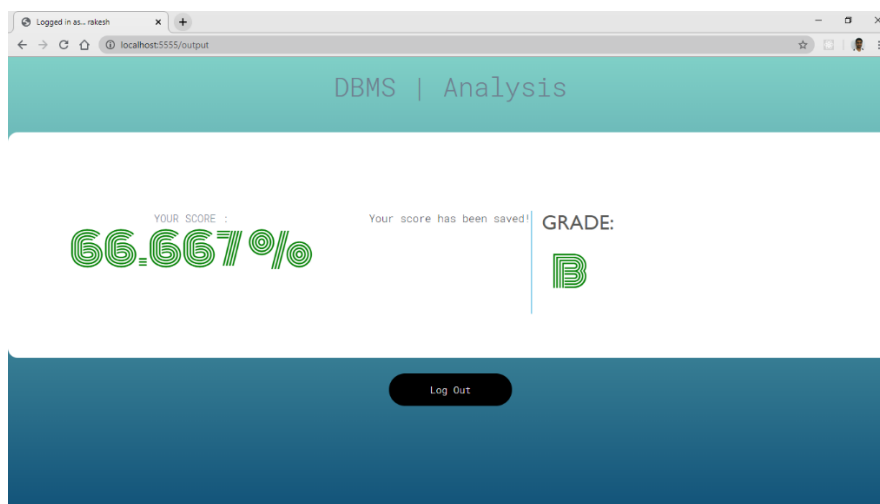


Figure 8.10.3 Snippet of result with B grade

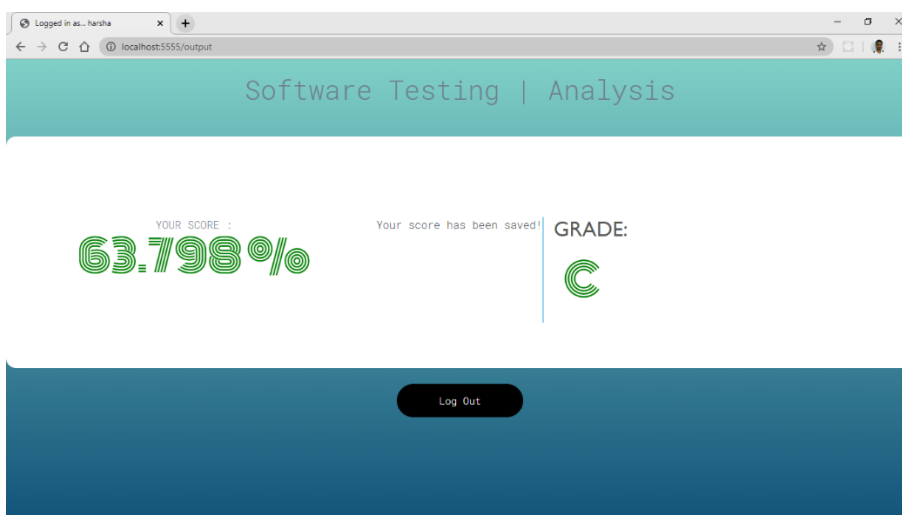


Figure 8.10.4 Snippet of result with C grade

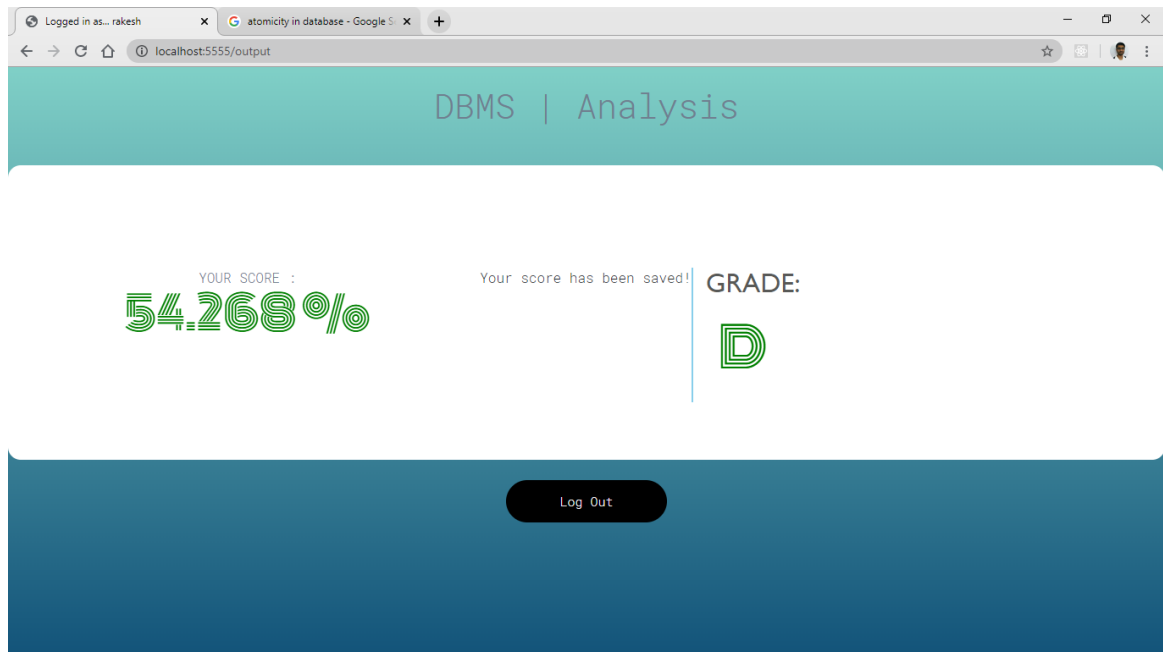


Figure 8.10.5 Snippet of result with D grade

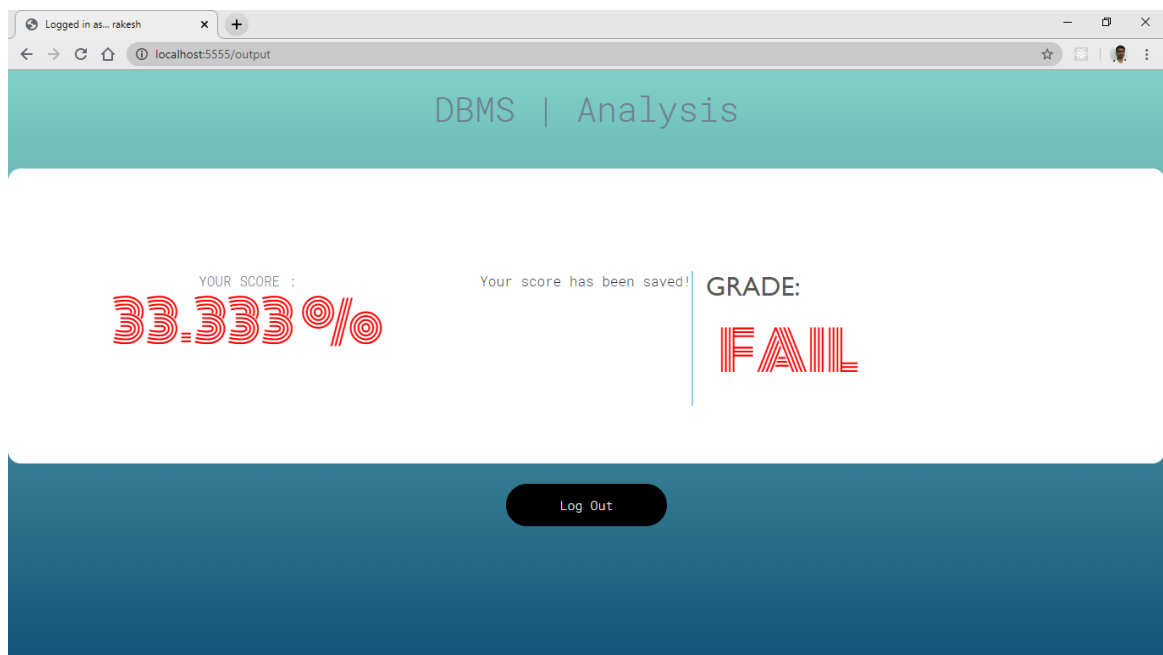


Figure 8.10.6 Snippet of result where the candidate failed to clear the test

8.11 An instance of file that stores details of registered users

```
atp > static > data > db > user_log.csv
1 Fullname,Username,password
2 Sriharsha Namilakonda,harsha,harsha123
3 Rakesh Kumar,rakesh,rakesh123
4 Srawan Vutharkar,srawan,srawan123
5 zoraver lawrence,zoraver,zoraver123
6 Himanshu Rohilla,himanshu,himanshu123
7 Deepankar Rao,deepankar,deepankar123
8 Chinmay Charith,charith,charith123
9 Abhimanyu Sripad,sripad,sripad123
10 Shashank Vakula,shashank,shashank123
11
```

Figure 8.11 Snippet of user data

8.12 An instance of file that stores details of user sessions

```
atp > static > data > db > user_data_log_subjective.csv
1 Date,Month,Year,Username,Subject,Score
2 18,5,2020,HARSHA,DBMS,65.157
3 18,5,2020,HARSHA,DBMS,63.411
4 18,5,2020,SRAWAN,DBMS,0.0
5 19,5,2020,RAKESH,DBMS,78.248
6 19,5,2020,RAKESH,DBMS,27.894
7 19,5,2020,HARSHA,DBMS,0.0
8 19,5,2020,HARSHA,ML,61.845
9 19,5,2020,HARSHA,ML,76.549
10 19,5,2020,DEEPANKAR,DBMS,64.656
11 19,5,2020,HIMANSHU,DBMS,86.928
12 19,5,2020,HARSHA,DBMS,31.402
13 19,5,2020,HARSHA,DBMS,97.593
14 19,5,2020,CHARITH,DBMS,78.516
15 19,5,2020,HARSHA,DBMS,93.71
16 24,5,2020,RAKESH,DBMS,54.268
17 24,5,2020,RAKESH,SOFTWARE TESTING,92.864
18 24,5,2020,RAKESH,DBMS,88.044
19 24,5,2020,HARSHA,DBMS,50.6
20 24,5,2020,HARSHA,SOFTWARE TESTING,69.002
21 24,5,2020,HARSHA,SOFTWARE TESTING,63.798
22 24,5,2020,HARSHA,DBMS,64.142
23 24,5,2020,HARSHA,SOFTWARE TESTING,85.158
24 24,5,2020,HARSHA,SOFTWARE TESTING,78.731
```

Figure 8.12 Snippet of user session data

9. CONCLUSION AND FUTURE PLAN

9.1 Conclusive Note

The automated test generation and evaluation is highly advantageous in the academia. It reduces the monotony for the examiners in the performance assessment as much as possible and it gets rid of onerous grading and rank-based statistical computation. In the current scenario, the automation in examination and evaluation is mostly limited to objective type of tests because of the simplicity in assessing the candidate responses. Considering the advancement of technology in rapid pace, various mechanisms that compute textual similarity are available and these techniques are getting improved in order to provide a suitable match to human grading.

This application can be customized according to needs of instructor, for instance, if the instructor decides to conduct test on fixed questions and provides the subjective information along with it, the corresponding answers can be extracted on the basis of key words present in the question. Consequently, the relevant question-answer pairs are composed that aid in the automated evaluation. This application also provides significant assistance to students in the preparation of a particular exam and to test their level of understanding on a topic.

9.2 Advantages and Disadvantages of the application

Advantages:

1. This application reduces the time and effort required by the examiners, so that the instructor can completely focus on pedagogical activities
2. It helps in the efficient utilization of available resources by the examination cell and minimizes the logistics cost
3. It follows the standardized evaluation mechanism for all the candidates and gets rid of differential human-to-human grading
4. It reduces the possibility of occurrence of malpractice and fraudulent activities to a considerable extent
5. As the result computation in this application is instantaneous, the candidate can assess the areas of improvement swiftly

Disadvantages:

1. As this application composes the questions for the test upon identification of keywords present in the information, the questions related to conceptual reasoning are not generated as required
2. This application lacks the human-touch i.e., human intellect to judge the creativity in the candidate responses. so, in the scenarios where out-of-the-box responses are expected from the candidates, the evaluation better be manual
3. This application currently lacks the capability to process the non-textual entities, which limits the usage situations. Moreover, the presentation of non-textual entities using pen and paper method is easy compared to any computerized tool

9.3 Future work

The application provides the score and grade of an instance of test to the candidate at present, the feedback mechanism needs to be improved with various data visualization techniques. For instance, progress of the candidate on a particular subject can be depicted by plotting all the scores obtained till then on a graph. The scalability of this application should be increased. The application should incorporate more functionalities in order to ease the customization according to needs of an instructor. The security practices followed in the application must be strengthened. The answer generation mechanism is solely reliant on the keyword in the current implementation application, it should be improved by considering the synonymous words of the corresponding keyword in the formation of ideal responses. This can be done using the "Synset" interface in natural language tool kit. The robustness in the evaluation mechanism (i.e., the technique for measuring similarity between two documents) needs to be intensified in order to process all kind of textual and non-textual entities.

REFERENCES

- [1] Akeem Olowolayemo, Santhy David Nawi , Teddy Mantoro, "Short Answer Scoring in English Grammar Using Text Similarity Measurement", International Conference on Computing, Engineering, and Design (ICCED), September 2018.
- [2] Amit Rokade, Bhushan Patil , Sana Rajani, Surabhi Revandkar, Rajashree Shedge, "Automated Grading System Using Natural Language Processing", Second International Conference on Inventive Communication and Computational Technologies (ICICCT), April 2018.
- [3] Jennifer O. Contreras, Shadi Hilles, Zainab Binti Abubakar, "Automated Essay Scoring with Ontology based on Text Mining and NLTK tools", International Conference on Smart Computing and Electronic Enterprise (ICSCEE), July 2018.
- [4] V. Sridevi, Suresh Kumar S, B. Supraja, S. Udhayakumar, "Knowledge Representation and Answer Evaluation System using Language Processing Algorithm", International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), March 2019.
- [5] Bilal H. Hungund, Anas Shaikh, Ghazi Owais, Aabidi Sayyed Ali, Pratibha Dumane, "Automated System for Management of Examination Processes", International Conference on Nascent Technologies in Engineering (ICNTE), January 2019.
- [6] Automated Essay Scoring - Wikipedia, the free encyclopedia, [online] Available: https://en.wikipedia.org/wiki/Automated_essay_scoring
- [7] Flask (web framework) - Wikipedia, the free encyclopedia, [online] Available: [https://en.wikipedia.org/wiki/Flask_\(web_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))
- [8] Frontend vs Backend - GeeksForGeeks Article, [online] Available: <https://www.geeksforgeeks.org/frontend-vs-backend/>
- [9] Selva Prabhakaran, "Cosine Similarity – Understanding the math and how it works (with python codes)", [online] Available: <https://www.machinelearningplus.com/nlp/cosine-similarity/>

[10] Christian S. Perone, "Machine Learning :: Cosine Similarity for Vector Space Models (Part III)", Terra Incognita, [online] Available:

<http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/>

[11] Natural Language Toolkit - NLTK 3.5 documentation, [online] Available: <https://www.nltk.org/>

[12] Benefits of Online Examination System, epravesh – Educational Technology for digital assessments, Exams, Admissions and trends, [online] Available:

<https://www.blog.epravesh.com/reasons-online-examination-system-schoolscolleges-university/>

APPENDIX

A. COMPLETE CONTRIBUTARY SOURCE CODE

1. Front-end code

a) form.html

```
<html lang="en">

<head>

    <title>Automate the examination system</title>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link rel="icon" type="image/png" href="images/icons/favicon.ico"/>

    <link rel="stylesheet" type="text/css" href="../static/form-
static/vendor/select2.min.css">

    <link rel="stylesheet" type="text/css" href="../static/form-static/css/util.css">

    <link rel="stylesheet" type="text/css" href="../static/form-static/css/main.css">

    <link rel="stylesheet" href="../static/css/style.css">

</head>

<body class="body2">

    <div class="head-space">

        <h5>Logged in as.. {{username}}</h5>

    </div>

    <div class="contact1 myownfont" style="box-shadow: 5px 8px 32px rgb(255, 94,
0);">

        <div class="contact1-pic js-tilt" data-tilt>
```

```

    </div>

    <form style="color:white" class="contact1-form validate-form"
action="/generate_test" method="POST" enctype="multipart/form-data">

    <!--          <div class="wrap-input1 validate-input" data-validate="Subject
Name is required">

                <input autocomplete="off" class="input1" type="text"
name="subjectname" id="subjectname" placeholder="Subject Name">

                <span class="shadow-input1"></span>

            </div> -->

    <div class="wrap-input1 validate-input" data-validate="">

        Select Test Type :

        <span>

            <input type="radio" name="test_id" value="obj">
Objective Test

            <input type="radio" name="test_id" value="subj">
Subjective Test

        </span>

    </div>

    <div class="wrap-input1 validate-input" data-validate="">

        Select Subject : <br>

        <blockquote style="padding-left: 50px">

            <input type="radio" name="subject_id" value="se"
id="myCheck4" onclick="myFunction2()"> Software Testing <br>

```



```
        <input type="radio" name="subject_id" value="db"
id="myCheck3" onclick="myFunction2()"> Databases <br>
```

```
        <input type="radio" name="subject_id" value="ml"
id="myCheck2" onclick="myFunction2()"> Machine Learning <br>
```

```
        <input type="radio" name="subject_id"
value="custom" id="myCheck" onclick="myFunction()"> Create Custom Test
```

```
    </blockquote>
```

```
</div>
```

```
    <div id="file_br" class="wrap-input1 validate-input" data-
validate="Test file is required" style="display:none">
```

```
        <input type="file" name="file" />
```

```
    </div>
```

```
    <div class="container-contact1-form-btn">
```

```
        <button class="contact1-form-newbtn">
```

```
            <span>Take Test<i class="" aria-hidden="true"></i></span>
```

```
        </button>
```

```
        <a href="/" class="contact1-form-newbtn-home">Home</a>
```

```
    </div>
```

```
</form>
```

```
</div>
```

```
</body>
```

```
<!-- JavaScript -->
```

```
function formHandler(){

    // event handler to handle the name field

    var nameField = document.getElementById("testname");

    nameField.onfocus = function(){

        if(nameField.placeholder == "Test Name"){

            nameField.placeholder = "";

        }

    };

    nameField.onblur = function(){

        if(nameField.placeholder == ""){

            nameField.placeholder = "Test Name";

        }

    };

}

// call fucntion after the window has loaded

window.onload = function(){

    formHandler();

}
```

```
</script>
```

```
<script>
```

```
function myFunction2() {

    var checkBox2 = document.getElementById("myCheck2");
```

```

        var checkBox3 = document.getElementById("myCheck3");

        var checkBox4 = document.getElementById("myCheck4");

        document.getElementById("file_br").style.display="none";

    }

</script>

</html>

```

b) index.html

```

<html class="no-js" lang="">

<head>

    <meta charset="utf-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

    <title>Automating the examination system</title>

    <meta name="description" content="" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <!-- CSS -->

    <link rel="stylesheet" href="../static/css/jquery.fancybox.css" />

    <link

        rel="stylesheet"

        type="text/css"

        href="../static/form-static/css/util.css"

    />

    <link

```

```
rel="stylesheet"

type="text/css"

href="../static/form-static/css/main.css"

/>

<!-- Javascript -->

<script src="../static/js/bootstrap.min.js"></script>

</head>

<!-- Body content -->

<body>

<div class="clouds">

<div class="row"></div>

<div class="row">

<div class="content">

<center>

<h1 class="tagline">Automated Test Platform</h1>

</center>

</div>

</div>

<div class="row1">

<div class="registration-content">

<center>

<p

style="
```

```

        color: white;

        font-size: 20px;

        position: absolute;

        top: 550px;

        left: 630px;

        font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
    "
>

    New user? Register here <br>

    <div class="container-contact1-form-btn" style="position: absolute; top:585px;
left:620px">

        <button class="contact1-form-btn" id="RegisterButton">

            <span>Register<i class="" aria-hidden="true"></i></span>

        </button>

    </div>

</p>

</center>

</div>

</div>

<center>

    <div class="row2" style="position: absolute; top: 350px; left: 300px;">

        <form

```

```
class="contact1-form validate-form"

action="/form"

method="POST"

enctype="multipart/form-data"

>

<div

class="wrap-input1 validate-input"

data-validate="Name is required"

>

<input

autocomplete="off"

class="input1"

type="text"

name="username"

id="username"

placeholder="Your User Name Here!"

/><br />

<input

autocomplete="off"

class="input1"

type="password"

name="password"

id="password"
```

```

        placeholder="Your Password Here!"

    />

    <span class="shadow-input1"></span>

</div>

<div class="container-contact1-form-btn">

    <button class="contact1-form-btn">

        <span>Start<i class="" aria-hidden="true"></i></span>

    </button>

</div>

</form>

</div>

</center>

</div>

</body>

</html>

<!-- JavaScript -->

function formHandler() {

    // event handler to handle the name field

    var nameField = document.getElementById("username");

    nameField.onfocus = function () {

        if (nameField.placeholder == "Your Name") {

            nameField.placeholder = "";

```

```

    }

};

nameField.onblur = function () {

    if (nameField.placeholder == "") {

        nameField.placeholder = "Your Name";

    }

};

}

// call fucntion after the window has loaded

window.onload = function () {

    formHandler();

};

document.getElementById("RegisterButton").onclick = function () {

    location.href = "/register";

};

</script>

```

c) loginutil.html

```

<html class="no-js" lang="">

<head>

    <meta charset="utf-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

    <title>Login Status</title>

```



```
<meta name="description" content="" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<!-- CSS -->

<link

    rel="stylesheet"

    type="text/css"

    href="../static/form-static/css/util.css"

/>

<link

    rel="stylesheet"

    type="text/css"

    href="../static/form-static/css/main.css"

/>

</head>

<!-- Body content -->

<body>

    <div class="clouds">

        <div class="row"></div>

        <div class="row">

            <div class="content">

                <center>

                    <h4

                        class="tagline"
```

```

        style="position: absolute; top: 100px; left: 400px;"

    >

    LOGIN &nbsp; STATUS

</h4>

</center>

</div>

</div>

<center>

<div class="row3" style="position: absolute; top: 300px; left: 280px;">

    <span class="MessageDisplay" style="color: white; font-size: 30px;">

        {{ result }}</span>

    >

</div>

</center>

<div style="position: absolute; top:400px; left:650px">

    <div class="container-contact1-form-btn">

        <button class="contact1-form-btn" id="HomeButton">

            <span>Go to Home Page<i class="" aria-hidden="true"></i></span>

        </button>

    </div>

</div>

</div>

```

```
</div>
```

```
</body>
```

```
</html>
```

```
<!-- JavaScript -->
```

```
<script>
```

```
document.getElementById("HomeButton").onclick = function () {
```

```
    location.href = "/home";
```

```
};
```

```
</script>
```

d) objective_test.html

```
<html lang="en">
```

```
<head>
```

```
    <title>{{ testname }} | {{ topicname }}</title>
```

```
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
    <link rel="icon" type="image/png" href="images/icons/favicon.ico"/>
```

```
    <link rel="stylesheet" type="text/css" href="../static/form-static/css/util.css">
```

```
    <link rel="stylesheet" type="text/css" href="../static/form-static/css/main.css">
```

```
    <link rel="stylesheet" href="../static/css/style.css">
```

```
</head>
```

```
<body class="contact2">
```

```
<h5 style=" color: white; font-family: 'Franklin Gothic Medium', 'Arial Narrow',  
Arial, sans-serif">Logged In as .. {{username}}</h5>
```

```
<div class="contact1new">
```

```
<form style="color:white" class="contact1-form validate-form"  
action="/output" method="POST" enctype="multipart/form-data">
```

```
<div class="wrap-input1 validate-input" data-validate="">
```

```
<h4>{{question1}}</h4>
```

```
<input autocomplete="off" class="input1" type="text"  
name="answer1" id="answer1" placeholder="One/Two Word Answer">
```

```
<span class="shadow-input1"></span>
```

```
</div>
```

```
<div class="wrap-input1 validate-input" data-validate="">
```

```
<h4>{{question2}}</h4>
```

```
<input autocomplete="off" class="input1" type="text"  
name="answer2" id="answer2" placeholder="One/Two word Answer">
```

```
<span class="shadow-input1"></span>
```

```
</div>
```

```
<div class="wrap-input1 validate-input" data-validate="">
```

```
<h4>{{question3}}</h4>
```

```
<input autocomplete="off" class="input1" type="text"  
name="answer3" id="answer3" placeholder="One/Two word Answer">
```

```
<span class="shadow-input1"></span>
```

```
</div>
```

```
<div class="container-contact1-form-btn">
```

```

        <button class="contact1-form-btn">

        <span>Submit<i class="" aria-
hidden="true"></i></span>

    </button>

</div>

<br>

</form>

</div>

</body>

</html>

```

```
<!-- JavaScript -->
```

```
<script src="../static/js/main.js"></script>
```

e) output.html

```

<html lang="en">

<head>

    <title>Logged in as... {{username}}</title>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <link rel="icon" type="image/png" href="images/icons/favicon.ico" />

    <link rel="stylesheet" href="../static/css/style.css" />

<style>

    .vl {

```

```

border-left: 2px solid skyblue;

height: 150%;

}

</style>

</head>

<body class="body3">

<br />

<center>

<h5 class="result_dec_header">{{ subjectname}} | Analysis</h5>

<br />

<h5>{{ topicname}}</h5>

<br />

</center>

<div class="contact1new-ultra">

<div class="row">

<div class="col result_dec">

<center>

<span

>YOUR SCORE :

<h5 id="fs" class="h1_beta">{{ show_score}} %</h5></span

>

</center>

</div>

```

{{status}}

<div class="vl"></div>

<div class="col">

<span

><h2

style="

color: #585957;

font-family: 'Gill Sans', 'Gill Sans MT', Calibri,

'Trebuchet MS', sans-serif;

"

>

GRADE:

</h2></span

>

<div class="col result_dec">

 <h5 id="grade" class="h1_beta"></h5>

</div>

</div>

<center>

```

<div class="container-contact1-form-btn">

    <a href="/" class="contact1-form-newbtn">Log Out</a>

</div>

</center>

</body>

</html>

<script>

var font_s = {{ show_score }};

if (font_s < 40){

    document.getElementById("fs").style.color="red";

}

else{

    font_s += "px";

    document.getElementById("fs").style.fontSize=font_s;

}

window.onload = function(){

    calculateGrade();

}

</script>

<!-- JavaScript -->

<script src="../static/js/main.js"></script>

```


f) registration.html

```
<html class="no-js" lang="">

<head>

  <meta charset="utf-8" />

  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

  <title>ATP || Registration</title>

  <meta name="description" content="" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <!-- CSS -->

  <link

    rel="stylesheet"

    type="text/css"

    href="../static/form-static/css/util.css"

  />

  <link

    rel="stylesheet"

    type="text/css"

    href="../static/form-static/css/main.css"

  />

</head>

<!-- Body content -->

<body>
```

```

<div class="clouds">

<div class="row"></div>

<div class="row">

<div class="content">

<center>

<h2
class="tagline"
style="position: absolute; top: 100px; left: 300px;"
>
REGISTRATION &nbsp; FORM

</h2>

</center>

</div>

</div>

<center>

<div class="row2" style="position: absolute; top: 200px; left: 300px;">

<form
class="contact1-form validate-form"
action="/validate"
method="POST"
enctype="multipart/form-data"
name="RegForm"

```

```
onsubmit="return formvalidate()"
```

```
>
```

```
<div
```

```
class="wrap-input1 validate-input"
```

```
data-validate="Full Name is required"
```

```
>
```

```
<input
```

```
autocomplete="off"
```

```
class="input1 "
```

```
type="text"
```

```
name="fullname"
```

```
id="fullname"
```

```
placeholder="Your Full Name Here!"
```

```
/><br /><br />
```

```
<input
```

```
autocomplete="off"
```

```
class="input1 "
```

```
type="text"
```

```
name="username"
```

```
id="username"
```

```
placeholder="Your username Here!"
```

```
/><br /><br />
```

```
<input  
  autocomplete="off"  
  class="input1"  
  type="password"  
  name="password"  
  id="password"  
  placeholder="Your Password Here!"  
/><br /><br />
```

```
<input  
  autocomplete="off"  
  class="input1"  
  type="password"  
  name="rpassword"  
  id="rpassword"  
  placeholder="Re-enter the Password Here!"  
/><br /><br />
```

```
<span class="shadow-input2"></span>  
  
</div>  
  
<div class="container-contact1-form-btn">  
  
  <button class="contact1-form-btn">  
  
    <span>Submit<i class="" aria-hidden="true"></i></span>
```

```

        </button>

    </div>

</form>

</div>

</center>

<div style="position: absolute; top: 580px; left: 200px;">

    <div class="container-contact1-form-btn">

        <button class="contact1-form-btn" id="BackButton">

            <span>Back<i class="" aria-hidden="true"></i></span>

        </button>

    </div>

</div>

</div>

</body>

</html>

<!-- JavaScript -->

<script>

    document.getElementById("BackButton").onclick = function () {

        location.href = "/home";

    };

    function formvalidate() {

        var fullname = document.forms["RegForm"]["fullname"];

```

```
var username = document.forms["RegForm"]["username"];

var password = document.forms["RegForm"]["password"];

var rpassword = document.forms["RegForm"]["rpassword"];

if (fullname.value == "") {

    window.alert("Please enter your full name!");

    fullname.focus();

    return false;

} else if (password.value == "" || rpassword.value == "") {

    window.alert("The password fields must not be left empty!");

    password.focus();

    return false;

} else if (password.value != rpassword.value) {

    window.alert("The password mismatch has occurred!");

    password.focus();

    return false;

}

return true;

}
```

</script>

g) subjective_test.html

```
<html lang="en">

<head>

    <title>{{testname}} | {{topicname}}</title>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link rel="icon" type="image/png" href="images/icons/favicon.ico"/>

    <link rel="stylesheet" type="text/css" href="../static/form-static/css/util.css">

    <link rel="stylesheet" type="text/css" href="../static/form-static/css/main.css">

    <link rel="stylesheet" href="../static/css/style.css">

</head>

<body class="contact2">

    <h5 style="color: white; font-family: 'Franklin Gothic Medium', 'Arial Narrow',
    Arial, sans-serif">Logged In as .. {{username}}</h5>

    <div class="contact1new">

        <form style="color:white" class="contact1-form validate-form"
        action="/output" method="POST" enctype="multipart/form-data">

            <div class="wrap-input1 validate-input" data-validate="">

                <h4>{{question1}}</h4>

                <input autocomplete="off" class="input1" type="text"
                name="answer1" id="answer1" placeholder="Short Descriptive Answer">

                <span class="shadow-input1"></span>

            </div>

            <div class="wrap-input1 validate-input" data-validate="">
```

```

<h4>{{question2}}</h4>

<input autocomplete="off" class="input1" type="text"
name="answer2" id="answer2" placeholder="Short Descriptive Answer">

<span class="shadow-input1"></span>

</div>

<div class="container-contact1-form-btn">

<button class="contact1-form-btn">

<span>Submit<i class="" aria-
hidden="true"></i></span>

</button>

</div>

<br>

</form>

</div>

</body>

</html>

<!-- JavaScript -->

<script src="../static/js/main.js"></script>

```


2. Backend code

a) cosine_similarity.py

```
# Load packages

import os

import math

import nltk as nlp

from nltk.corpus import stopwords


def find_mod_vector(vector):

    summ = 0

    for x in vector:

        summ += x**2

    return math.sqrt(summ)


def compute_vector(small_list, main_list):

    myvector = list()

    for i in range(len(main_list)):

        key = main_list[i]

        if key in small_list:

            myvector.append(1)

        else:

            myvector.append(0)

    return myvector
```

```

def get_vector(string_str):

    sentences = nlp.sent_tokenize(string_str)

    t_list = list()

    for s in sentences:

        temp = nlp.word_tokenize(s)

        for x in temp:

            t_list.append(x)

    return t_list

def evaluate_subj_answer(original_answer, user_answer):

    score_obt = 0

    orig_list = get_vector(original_answer)

    user_list = get_vector(user_answer)

    main_list = orig_list + user_list

    vector1 = compute_vector(orig_list, main_list)

    vector2 = compute_vector(user_list, main_list)

    v1 = find_mod_vector(vector1)

    v2 = find_mod_vector(vector2)

    v1_v2 = np.dot(vector1, vector2)

    dist = v1_v2 / (v1 * v2)

```

```
score_obt = dist * 100
```

```
return score_obt
```

b) objective_question.py

```
# Import packages
```

```
import nltk
```

```
from textblob import TextBlob
```

```
class ObjectiveQuestion:
```

```
    def __init__(self, title):
```

```
        self.title = title
```

```
        try:
```

```
            with open(title, mode="r") as fp:
```

```
                self.summary = fp.read()
```

```
        except FileNotFoundError as e:
```

```
            print(e)
```

```
    def generate_trivia_sentences(self):
```

```
        sentences = nltk.sent_tokenize(self.summary)
```

```
        trivia_sentences = list()
```

```
        for sentence in sentences:
```

```
            trivia = self.evaluate_sentence(sentence)
```

```
            if trivia:
```

```
                trivia_sentences.append(trivia)
```

```

    return trivia_sentences

@staticmethod
def get_similar_words(word):

    # In the absence of a better method, take the first synset

    synsets = wn.synsets(word, pos="n")

    # If there aren't any synsets, return an empty list

    if len(synsets) == 0:

        return []

    else:

        synset = synsets[0]

        # Get the hypernym for this synset (again, take the first)

        hypernym = synset.hypernyms()[0]

        # Get some hyponyms from this hypernym

        hyponyms = hypernym.hyponyms()

        # Take the name of the first lemma for the first 8 hyponyms

        similar_words = []

        for hyponym in hyponyms:

            similar_word = hyponym.lemmas()[0].name().replace("_", " ")

            if similar_word != word:

                similar_words.append(similar_word)

```

```

        if len(similar_words) == 8:

            break

    return similar_words

def evaluate_sentence(self, sentence):

    # If sentence starts with an adverb or is less than 4 words long probably not the best
    fit

    tags = nltk.pos_tag(sentence)

    if tags[0][1] == "RB" or len(nltk.word_tokenize(sentence)) < 4:

        return None

    noun_phrases = list()

    grammer = r"""

    CHUNK: {<NN>+<IN|DT>*<NN>+}

            {<NN>+<IN|DT>*<NNP>+}

            {<NNP>+<NNS>*}

    """

    chunker = nltk.RegexpParser(grammer)

    pos_tokens = nltk.tag.pos_tag(tokens)

    tree = chunker.parse(pos_tokens)

    for subtree in tree.subtrees():

        if subtree.label() == "CHUNK":

            temp = ""

            for sub in subtree:

                temp += sub[0]

            temp += " "

```

```

temp = temp.strip()

noun_phrases.append(temp)

replace_nouns = []

for word, _ in tags:

    for phrase in noun_phrases:

        if phrase[0] == "\":

            # If it starts with an apostrophe, ignore it

            # (this is a weird error that should probably be handled elsewhere)

            break

        if word in phrase:

            # Blank out the last two words in this phrase

            [replace_nouns.append(phrase_word) for phrase_word in phrase.split()[-2:]]

            break

    # If we couldn't find the word in any phrases

    if len(replace_nouns) == 0:

        replace_nouns.append(word)

    break

if len(replace_nouns) == 0:

    # Return none if we found no words to replace

    return None

val = 99

for i in replace_nouns:

    if len(i) < val:

```

```

        val = len(i)

    trivia = {

        "Answer": " ".join(replace_nouns),

        "Anser_key": val

    }

    if len(replace_nouns) == 1:

        # If we're only replacing one word, use WordNet to find similar words

        trivia["similar_words"] = self.get_similar_words(replace_nouns[0])

    else:

        # If we're replacing a phrase, don't bother - it's too unlikely to make sense

        trivia["similar_words"] = []

    # Blank out our replace words (only the first occurrence of the word in the sentence)

    replace_phrase = " ".join(replace_nouns)

    blanks_phrase = ("_____ " * len(replace_nouns)).strip()

    expression = re.compile(re.escape(replace_phrase), re.IGNORECASE)

    sentence = expression.sub(blanks_phrase, str(sentence), count=1)

    trivia["Question"] = sentence

    return trivia

```

c) subjective_question.py

```
# Load packages

import nltk as nlp

# Question patterns

question_formats = [

    "Explain in detail ",

    "Define ",

    "Write a short note on ",

    "What do you mean by "

]

# Grammer to chunk keywords

grammer = r"""

CHUNK: {<NN>+<IN|DT>*<NN>+}

{<NN>+<IN|DT>*<NNP>+}

{<NNP>+<NNS>+}

"""

def generate_subj_question(filepath):

    # Open file and load data

    try:

        fp = open(filepath, mode="r")

        data = fp.read()

        fp.close()

    except FileNotFoundError as e:

        print(e)
```



```

sentences = nlp.sent_tokenize(data)

que_ans_dict = dict()


# Train the regex parser on the above grammar
cp = nlp.RegexpParser(grammar)


# Select imp sentence to generate questions
for sentence in sentences:

    tagged_words = nlp.pos_tag(nlp.word_tokenize(sentence))


    # Parse the words to select the imp keywords to generate questions
    tree = cp.parse(tagged_words)

    for subtree in tree.subtrees():

        if subtree.label() == "CHUNK":

            temp = ""

            # Traverse through the subtree

            for sub in subtree:

                temp += sub[0]

                temp += " "

            temp = temp.strip()

            temp = temp.upper()

            if temp not in que_ans_dict:

                if len(nlp.word_tokenize(sentence)) > 20:

```

```

        que_ans_dict[temp] = sentence

    else:

        que_ans_dict[temp] += sentence

# Get a list of all keywords

keyword_list = list(que_ans_dict.keys())

que_ans_pair2 = list()

# Form questions

for _ in range(3):

    rand_num = np.random.randint(0, len(keyword_list))

    selected_key = keyword_list[rand_num]

    answer = que_ans_dict[selected_key]

    # Get a question format

    rand_num %= 4

    que_format = question_formats[rand_num]

    question = que_format + selected_key + "."

    # Make a dictionary and append to the main list repo

    que_ans_pair2.append({"Question": question, "Answer": answer})

return que_ans_pair2

```

d) util.py

```

# Import packages

import os

from datetime import datetime

from atp.objective_question import ObjectiveQuestion

```

```

# Path for backup

subjective_path = str(os.getcwd()) + "/atp/static/data/db/user_data_log_subjective.csv"

objective_path = str(os.getcwd()) + "/atp/static/data/db/user_data_log_objective.csv"


def back_up_data(username, subject_name, score_obt, flag):

    # Transform username and subject_name for backup

    username = "_".join([x.upper() for x in username.split()])

    subject_name = subject_name.strip().upper()


    if flag == "objective":

        filepath = objective_path

    else:

        filepath = subjective_path


    # Create a row for backup

    column_names = ["Date", "Month", "Year", "Username", "Subject", "Score"]

    row = [date, month, year, username, subject_name, score_obt]

    try:

        with open(filepath, 'a+', newline='') as fp:

            fp_writer = csv.writer(fp)

            fp_writer.writerow(row)

    except Exception as e:

```

```
        print(e)

    return True

def get_question_answer_pairs(pair, flag):

    if flag == "obj":

        length = 3

    elif flag == "subj":

        length = 2

    else:

        print("Error! Wrong `test_id` passed.")

        return None


    que = list()

    ans = list()


    while len(que) < length:

        rand_num = np.random.randint(0, len(pair))

        if pair[rand_num]["Question"] not in que:

            que.append(pair[rand_num]["Question"])

            ans.append(pair[rand_num]["Answer"])

        else:

            continue

    return que, ans
```

```

def generate_trivia(filename):

    questions = list()

    obj_a = ObjectiveQuestion(filename)

    questions.append(obj_a.generate_trivia_sentences())

    que_ans_pair = list()

    for lis in questions:

        for que in lis:

            if que["Anser_key"] > 3:

                que_ans_pair.append(que)

            else:

                continue

    return que_ans_pair

```

```

def relative_ranking(subjectname, flag):

    subjectname = subjectname.upper()

    max_score = 100

    min_score = 0

    mean_score = 50

    return max_score, mean_score, min_score

```

e) runserver.py

```

"""

```

This script runs the ATP application using a development server.

```

"""

```

```

from atp import app

```

```

if __name__ == "__main__":

    HOST = environ.get("SERVER_HOST", "localhost")

    try:

        PORT = int(environ.get("SERVER_PORT", "5555"))

    except ValueError:

        PORT = 1234

    app.run(HOST, PORT, debug=True)

```

f) views.py

```

# Import packages

import click

import flask

import csv

import pandas as pd

from atp import app

from atp.objective_question import ObjectiveQuestion

from atp.subjective_question import generate_subj_question

from atp.cosine_similarity import evaluate_subj_answer

from atp.util import generate_trivia, get_question_answer_pairs

from atp.util import relative_ranking, back_up_data


# Global placehodlers

global_name_list = list()

global_answer_list = list()

```

```

global_test_id = list()

user_path = str(os.getcwd()) + "/atp/static/data/db/user_log.csv"


@app.route('/')

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

def home():

    return render_template(

        "index.html",

        date=datetime.now().day,

        month=datetime.now().month,

        year=datetime.now().year

    )

@app.route("/form", methods=['GET', 'POST'])

def form():

    """ Prompt user to start the test """

    global_name_list.clear()

    user_name = request.form["username"]

    pswd = request.form["password"]


    flag = 0

    if user_name == "":

        user_name = "Admin"


    with open(user_path, mode = "r") as fp:

```

```

fp_reader = csv.reader(fp)

next(fp)

for line in fp_reader:

    if not len(line):

        break

    if line[1] == user_name and line[2] == pswd:

        flag=1

        break

if flag == 1:

    global_name_list.append(user_name)

    return render_template(

        "form.html",

        username = user_name

    )

else:

    message = "Either the username or the password is invalid"

    return render_template(

        "loginutil.html",

        result = message

    )

@app.route("/validate", methods=['GET','POST'])

def validate():

```



```
''' Prompt the validation of the registration form'''
```

```
flag = 0
```

```
full_name = request.form["fullname"]
```

```
user_name = request.form["username"]
```

```
password = request.form["password"]
```

```
with open(user_path, mode = "r") as fp:
```

```
    fp_reader = csv.reader(fp)
```

```
    next(fp)
```

```
    for line in fp_reader:
```

```
        if not len(line):
```

```
            break
```

```
        if line[1] == user_name:
```

```
            flag=1
```

```
            break
```

```
if flag == 0:
```

```
    row = [full_name,user_name,password]
```

```
    try:
```

```
        with open(user_path, 'a+',newline=") as fp:
```

```
            fp_writer = csv.writer(fp)
```

```
            fp_writer.writerow(row)
```

```
    except Exception as e:
```

```
        print(e)
```

```

message = ""

if flag == 0:

    message = "Registration successful! Welcome to the application!"

else:

    message = "A user with the provided username already exists!"


return render_template(

    "validation.html",

    result = message

)


@app.route("/register", methods=['GET','POST'])
def register():

    """ Prompt the user to register for the test"""

    return render_template(

        "registration.html"

    )


@app.route("/generate_test", methods=['GET', 'POST'])
def generate_test():

    subject_id = request.form["subject_id"]

    if subject_id == "se":

        global_name_list.append("Software Testing")

```

```

    filename = str(os.getcwd()) + "/sample_test_data/software-testing.txt"

    print(filename)

elif subject_id == "db":

    global_name_list.append("DBMS")

    filename = str(os.getcwd()) + "/sample_test_data/dbms.txt"

elif subject_id == "ml":

    global_name_list.append("ML")

    filename = str(os.getcwd()) + "/sample_test_data/ml.txt"

elif subject_id == "custom":

    file = request.files["file"]

    file.save(secure_filename(file.filename))

    global_name_list.append("Custom")

else:

    print("Error! Wrong `subject_id` received from the HTML form.")

    return None


test_id = request.form["test_id"]

global_test_id.append(test_id)


if test_id == "obj":

    que_ans_pair = generate_trivia(filename)

    question_list, answer_list = get_question_answer_pairs(que_ans_pair, test_id)

    for ans in answer_list:

        global_answer_list.append(ans)

```

```

return render_template(
    "objective_test.html",
    username=global_name_list[0],
    testname=global_name_list[1],
    question1=question_list[0],
    question2=question_list[1],
    question3=question_list[2]
)

elif test_id == "subj":
    que_ans_pair = generate_subj_question(filename)
    question_list, answer_list = get_question_answer_pairs(que_ans_pair, test_id)
    for ans in answer_list:
        global_answer_list.append(ans)

return render_template(
    "subjective_test.html",
    username=global_name_list[0],
    testname=global_name_list[1],
    question1=question_list[0],
    question2=question_list[1]
)

else:
    print("Error! Wrong `test_id` received from the HTML form.")

```

```

        return None

@app.route("/output", methods=["GET", "POST"])
def output():

    default_ans = list()

    user_ans = list()

    if global_test_id[0] == "obj":

        # Access objective answers

        user_ans.append(str(request.form["answer1"]).strip().upper())

        user_ans.append(str(request.form["answer2"]).strip().upper())

        user_ans.append(str(request.form["answer3"]).strip().upper())

    elif global_test_id[0] == "subj":

        # Access subjective answers

        user_ans.append(str(request.form["answer1"]).strip().upper())

        user_ans.append(str(request.form["answer2"]).strip().upper())

    else:

        print("Error! Wrong `test_id` found in `global_test_id`!!")

    for x in global_answer_list:

        default_ans.append(str(x).strip().upper())

    username = global_name_list[0]

    subjectname = global_name_list[1]

    # Evaluate the user response

    total_score = 0

```

```

flag = ""

if global_test_id[0] == "obj":

    flag = "objective"

    # evaluate objective answer

    for i in range(len(user_ans)):

        if user_ans[i] == default_ans[i]:

            total_score += 100

    total_score /= 3

    total_score = round(total_score, 3)

    # back up the user details and score for rank analysis

    score_status = "Failed to save your score!"

    if back_up_data(username, subjectname, total_score, flag):

        score_status = "Your score has been saved!"

elif global_test_id[0] == "subj":

    flag = "subjective"

    # evaluate subjective answer

    for i in range(len(default_ans)):

        temp_score = evaluate_subj_answer(default_ans[i], user_ans[i])

        if np.isnan(temp_score):

            temp_score=0.0

        total_score+=temp_score

    total_score /= 2

    total_score = round(total_score, 3)

    if np.isnan(total_score):

```

```

        total_score=0.0

    # back up the user details and score for rank analysis

    score_status = "Failed to save your score!"

    if back_up_data(username, subjectname, total_score, flag):

        score_status = "Your score has been saved!"

    max_score, mean_score, min_score = relative_ranking(subjectname, flag)

    # Clear the global variables for next instance

    global_name_list.clear()

    global_answer_list.clear()

    global_test_id.clear()

    global_test_id.clear()

    user_ans.clear()

    default_ans.clear()


    return render_template(

        "output.html",

        show_score=total_score,

        username=username,

        subjectname=subjectname,

        status=score_status,

        max_score=max_score,

        mean_score=mean_score,

        min_score=min_score

    )

```