HANDWRITTEN ANSWERSHEET EVALUATION APPLICATION

A PROJECT REPORT

Submitted by

ANANDHAN R (310917205003)

SRIMATHI B (310917205036)

SUBAKESHINI R (310917205038)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

JERUSALEM COLLEGE OF ENGINEERING
(An Autonomous Institution, Affiliated to Anna University, Chennai)

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2021

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"HANDWRITTEN ANSWERSHEET EVALUATION APPLICATION"** is the bonafide work of **"ANANDHAN R (310917205003), SRIMATHI B (310917205036), SUBAKESHINI R (310917205038)** who carried out the project work under my supervision.

SIGNATURE

Dr. K. SUNDARAMOORTHY

Professor and Head

Department of Information Technology

Jerusalem College of Engineering

Pallikaranai, Chennai-600100

SIGNATURE

Mr. GEORGE FERNANDEZ.I

Assistant Professor, Supervisor

Department of Information Technology

Jerusalem College of Engineering

Pallikaranai, Chennai-600100

Submitted to the project viva-voce Examination held on_____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to extend our sincere thanks and gratitude to our honorable CEO **Prof. M. MALA, M.A., M.Phil.,** and our Director **Dr. M. RAMALINGAM, B.E., M.S., PhD.,** and our Principal, **Dr.S.PRINCE AROCKIA DOSS M.E., PhD** for providing us all the facilities and resources required to complete our project successfully.

We find no words to express our deep sense of gratitude to **Dr.K.SUNDARAMOORTHY, M.E., PhD,** Professor and Head, Department of Information Technology, for being a constant source of inspiration throughout the course of our project.

We thank our Project Coordinator **Mr. D. SUDHAGAR, M.E., (PhD),** Associate Professor, for their advice and assistance in keeping our progress on schedule.

We express our deep sense of gratitude to our supervisor**, Mr.I.GEORGE FERNANDEZ, B.E, M.Tech, (PhD).,** Assistant Professor of our Department, for his valuable guidance and useful suggestions, which helped us in completing this project successfully.

Our grateful thanks are also extended to all our department faculty members for their valuable technical support on this project. Finally, we thank almighty, our parents and friends for their constant encouragement without which this project work would not be possible.

ANANDHAN R

SRIMATHI B

SUBAKESHINI R

ABSTRACT

Automatic evaluation of handwriting answers has been a difficult problem for education system for many years. To speeding up the evaluation remains as the major problem for enhancing the throughput of instructors. This paper shows an easy method for automatically evaluating the handwritten answers from the images. Our main goal is to build an application to evaluate a student's handwritten answer by assigning an evaluation score that is comparable to the human giving scores. Although many essay evaluation systems are available, short answer grading is still a tough problem. In the proposed system, build a application using tkinter, Optical Character Recognition tools are used to extract the keyword printed texts in keyword answer image and Google Vision API tools are used to extract the handwritten texts in student handwritten answer images. In the proposed model evaluates scores based on cosine similarity function. Each sentence in the evaluated answer paper carries their respective mark. The developed model can be used to evaluate and provide the marks of the student handwritten answer sheets. Our System is divided into three modules. The first and second one is extracting the data from the scanned printed text and handwritten answer images and organizing it in the proper manner and the third is applying NLTK and cosine similarity function from the above step and giving marks in screen.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AES	Automated Essay Scoring
OCR	Optical Character Recognition
API	Application Programming Interface
NLTK	Natural Language Tool Kit
JSON	JavaScript Object Notation
GUI	Graphical User Interface
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
UML	Unified Modeling Language
VGSL	Variable Graph Specification Language
TF-IDF	Term Frequency – Inverse Document Frequency

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The evaluation of handwritten text answers is a challenging process that requires great effort from the evaluators, especially when the number of answers to evaluate is high. Developing an automatic answer script evaluation system is needed because human evaluation needs concentration and might be biased, whereas an automatic answer evaluation system will be effective without these limitations. Regarding text answer evaluation, sentence similarity measures have been widely used to compare student written answers with reference texts. In this paper, we propose an automated answer evaluation system that uses our proposed cosine-based sentence similarity measures to evaluate the answers. Cosine measures have proved to be effective in comparing between free text student answers and reference texts.

Handwritten Text Recognition and Evaluating the Answer sheet of short answers and comparing with Key word to find the result with Percentage of Matches is a proposed technology that is much needed in this world as of today. Before proper implementation of this technology, we have relied on writing texts with our own hands and evaluating manually which can result in errors. It's difficult for staff to correct the same set of answer sheets manually. Manual labour is required in order to maintain proper organization of the data. Throughout history, there has been lot of Revaluation of answer sheets evaluation will get reduce. Modern day technology is letting people store the data over machines, where the storage, organization and accessing of data is relatively easier. Adopting the use of Handwritten Text Recognition software, it's easier to evaluate the hand written answer sheets. Furthermore, it provides more security to the data. The aim of our project is to build a application to recognize the handwriting and evaluate the answers and provide the result.

1.2 SCOPE OF THE PROJECT

This project aims to reduce the manpower and speeding up the evaluation process and save time to generate results. The system provides the best user interface. The efficient reports can be generated by using this proposed system.

The assessment of answers is an important process that requires great effort from evaluators. This assessment process requires high concentration without any fluctuations in mood. This substantiates the need to automate answer script evaluation.

1.3 OBJECTIVES

The main objectives of this system is to provide application to automatic evaluation of answer based on the keyword provided to the application in form of the input by the moderator which will provide marks and will reduce time and manpower.

1.4 ORGANIZATION OF THE PROJECT

The Organization of the project deals with outlines of each chapter. Each chapter explains each part of the project. Each project outlines a specific topic of the project.

Chapter 1: Represents about the scope and organization of the project.

Chapter 2: Consists of literature survey, which exploits the existing methodology and its disadvantages.

Chapter 3: Depicts the software and hardware specification of the project.

Chapter 4: Presents the methodology, module design of the project.

Chapter 5: Represents the implementation and results of the project.

Chapter 6: Explain the testing process of the project.

Chapter 7: Gives the conclusion and future enhancement of the project.

1.5 SUMMARY

Thus, the various aspects and organization of the project are discussed in this chapter. Mainly the need for the food and the platform used are discussed. The motivation of the project is also explained along with the introduction about the project and a brief knowledge about the techniques used is described.

CHAPTER 2

LITERATURE SURVEY

2.1 GENERAL

There are several works of researches which made us to understand the existing system better. It is necessary to get through with the existing system before designing the proposed system. Some of those important works are discussed in this chapter.

2.2 RELATED WORKS

2.2.1 Cosine similarity to determine similarity measure: Study case in online essay assessment.

Development of technology in educational field brings the easier ways through the variety of facilitation for learning process, sharing files, giving assignment and assessment [1]. Automated Essay Scoring (AES) is one of the development systems for determining a score automatically from text document source to facilitate the correction and scoring by utilizing applications that run on the computer. AES process is used to help the lecturers to score efficiently and effectively. Besides it can reduce the subjectivity scoring problem. However, implementation of AES depends on many factors and cases, such as language and mechanism of scoring process especially for essay scoring. A number of methods implemented for weighting the terms from document and reaching the solutions for handling comparative level between documents answer and expert's document still defined. In this research, we implemented the weighting of Term Frequency - Inverse Document Frequency (TF-IDF) method and Cosine Similarity with the measuring degree concept of similarity terms in a document. Tests carried out on a number of Indonesian text-based documents that have gone through the stage of pre-processing for data extraction purposes. This process results is in a ranking of the document weight that have closeness match level with expert's document.

2.2.2 Cloud based Text extraction using Google Cloud Vison for Visually Impaired applications.

As number of visually impaired people increased year by year due to accidents and biological disorders [6]. Hence there is a need for assistive device that can helpful for their day to day activities. Hence, with recent advancement in the technology, they intend to implement assistive device for visually impaired person such smart reader that is capable of capturing an image from a camera and extract the text from the captured image and further to convert the text to speech as voice-based output to assist the visually impaired people. The captured image is analysed using Google Cloud Vision API Optical Character recognition (OCR). In order to extract text, we use image pre-processing methods to remove any noise or blur in the captured image so that the accuracy can be increased. Further, they include software-based text to speech to convert the text to speech as voice output

2.2.3 Automated Text Extraction from Images using OCR System.

Digital images are getting popular rapidly. Every day, many images have been generated by many groups like students, engineer, doctors, according to their varying needs. They can access images based on its primitive features or associated text [3]. Text present in such images can provide meaningful information. We aim to retrieve the content and summarize the visual information automatically from images. Optical character recognition system that involves several algorithms are required for this purpose. Tesseract is currently the most accurate optical character recognition engine which was developed by HP Labs and is currently owned by Google. In this paper, we extract text from images using text localization, segmentation and binarization techniques. Text extraction can be achieved by applying text detection that identifies image parts containing text, text localization finds the exact position of the text, text segmentation separates the text from its background and binarization process converts the coloured images into binary. On this binary image, character recognition

is applied to convert it into ASCII text. Text extraction is used in creating e-books from scanned books, image searching from a collection of visual data etc.

2.2.4 Using Google Cloud Vision in assistive technology scenarios.

Google Cloud Vision is an image recognition technology that allows us to remotely process the content of an image and to retrieve its main features. By using specialized REST API, called Google Cloud Vision API [5], developers exploit such a technology within their own applications. Currently, this tool is in limited preview and its services are accessible for trusted tester users only. From a developer's perspective, in this paper, we intend to use such software resources in order to achieve assistive technology solutions for people with disabilities. Specifically, we investigate some potential benefits of Cloud Vision tool towards the development of applications for users who are blind.

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

One of the common learning activities used in educational levels and disciplines is essay writing [7]. The problems of the essay writing activities are time-consuming, concerns in producing immediate result and/or feedback from teachers to students, and the teachers tend to be subjective in grading the essay activities. The study aims to apply the preliminary approach for automatically generating the domain concept ontology in essays using Onto Gen and applied natural language processing algorithms using NLTK (Natural Language Tool Kit) that enhance the teachers essay grading.

2.3 SUMMARY

Thus, in this chapter, the literature survey of the project has been explained. It gives brief information about the related works similar to this project and knowledge about the techniques used in those papers. By knowing the Information, the current disadvantages and the drawbacks of the projects can be altered and modified.

CHAPTER 3

SYSTEM ANALYSIS AND REQUIREMENTS

3.1 GENERAL

This chapter clearly shows the system requirements used in the system design. System analysis as “the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way”. Another view sees system analysis as a problem-solving technique that breaks down a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose. The Proposed system requires the general hardware and software requirements. A brief description of the software requirements are presented here.

3.2 EXISTING SYSTEM

A natural direction to evaluate the handwritten answers is to convert into textual content and then exploit the advances in the text-based automatic evaluation. While the optical character recognizer (OCR) can reliably recognize printed text, offline handwritten text recognizer for unconstrained vocabulary are not robust enough for the practical use due to the inherent complexity of a handwritten word image. Automated evaluation of assessments is an active area of research in the text domain. A multitude of measures were proposed for computing similarity between the reference answer and the candidate answer. Handwritten Text Recognition and Evaluating the Answer sheet for Mathematical answers is already existing system which is used only for evaluating the Mathematical evaluation.

3.2.1 CHALLENGES IN EXISTING SYSTEM

Even though traditional OCR tools have been in the market since the 70s, there are still not many tools that can handle handwriting recognition. As everyone has their own style of writing, traditional OCR tools cannot perceive everyone’s handwriting. Besides computer vision technology, highly complex deep learning algorithms are

required to identify all these variations successfully. Below is a list of challenges that handwriting recognition tools frequently encounter.

Higher image quality is critical for handwriting recognition, however OCR solutions need to deal with a variety of quality and variety of individual handwritings, including different styles and different alphabets characters might be skewed which makes recognition harder.

3.3 SYSTEM REQUIREMENTS

The hardware and software specification required for proposed system are discussed below.

3.3.1 SOFTWARE REQUIREMENTS

Software requirements deals with defining resource requirements and prerequisites that needs to be installed on a computer to provide functioning of an application. The minimal software requirements are as follows,

- Operating System : Windows, Linux, MacOS
- IDE: Spyder
- Coding Language: Python3
- Software tools: Tesseract OCR Engine, PIP, tkinter interface

3.3.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. The minimal hardware requirements are as follows,

- Processor : corei3/i5
- Hard Disk : 500GB
- RAM : 3GB and above

3.4 REQUIREMENT ANALYSIS

3.4.1 PYTHON3

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects 15.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release." No more security patches or other improvements will be released for it. The Python 2's end-of-life, only Python 3.5.x are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

3.4.2 GOOGLE VISION API

Google Cloud offers two computer vision products that use machine learning to help you understand your images with industry-leading prediction accuracy.

Google Cloud's Vision API offers powerful pre-trained machine learning models through REST and RPC APIs. Assign labels to images and quickly classify them into millions of predefined categories. Detect objects and faces, read printed and handwritten text, and build valuable metadata into your image catalog.

The Google Cloud Vision API is in general availability and there is a free tier, where you are allowed 1,000 units per Feature Request per month free. Beyond that there is a tiered pricing model based on the number of units that you use in a month.

The Google Cloud Vision API allows developers to easily integrate vision detection features within applications, including image labeling, face and landmark detection, optical character recognition (OCR), and tagging of explicit content using the Vision API with Python.

3.4.3 TESSERACT

Tesseract is an open source text recognition (OCR) Engine, available under the Apache 2.0 license. It can be used directly, or (for programmers) using an API to extract printed text from images. It supports a wide variety of languages. Tesseract doesn't have a built-in GUI, but there are several available from the 3rd Party page. Tesseract is compatible with many programming languages and frameworks through wrappers that can be found here. It can be used with the existing layout analysis to recognize text within a large document, or it can be used in conjunction with an external text detector to recognize text from an image of a single text line.

Tesseract 4.00 includes a new neural network subsystem configured as a text line recognizer. It has its origins in Corpus' Python-based LSTM implementation but has been redesigned for Tesseract in C++. The neural network system in Tesseract pre-dates TensorFlow but is compatible with it, as there is a network description language called Variable Graph Specification Language (VGSL), that is also available for TensorFlow.

To recognize an image containing a single character, we typically use a Convolutional Neural Network (CNN). Text of arbitrary length is a sequence of characters, and such problems are solved using RNNs and LSTM is a popular form of RNN. Read this post to learn more about LSTM.

Install the python wrapper for tesseract after this using pip.

$ pip install pytesseract.

3.5 SUMMARY

This chapter describes about the existing system issues and the system requirements. It gives the complete overview of the existing system (i.e.) the issues and the disadvantages in the system. This also provides the detailed description of the system requirements with these requirements only the project can be done.

CHAPTER 4

DESIGN AND IMPLEMENTATION OF PROPOSED SYSTEM

4.1 GENERAL

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. This chapter gives the overall view of the module's description and the proposed architecture of the project.

4.2 PROPOSED SYSTEM

Many schemes and methods are currently available for evaluation of essays. But automatic evaluation and grading cannot be adapted for descriptive answers. In this approach, a novel method for automatic assessment for descriptive text answers is proposed.

To overcome the challenges on Handwritten text recognition Google Vision API turned out to be a great tool to get a handwritten text from a photo. To access the service, start with the registration to Google Cloud. Google requires authentication, but it's simple and painless we will only need to store a JSON file that's including API key, which you can get directly from the Google Cloud Platform. Extract the answer key in printed form using Tesseract OCR. Evaluating the extracted handwritten short descriptive answer comparing with Key word to find the result with Percentage of Matches is a proposed technology using Cosine Similarity function. Using Tkinter GUI toolkit access the image from user.

Our System is divided into three modules. The first and second one is extracting the data from the scanned printed text and handwritten answer images and organizing it in the proper manner and the second is applying NLTK and cosine similarity function from the above step and giving marks in screen.

4.3 ARCHITECTURE DIAGRAM

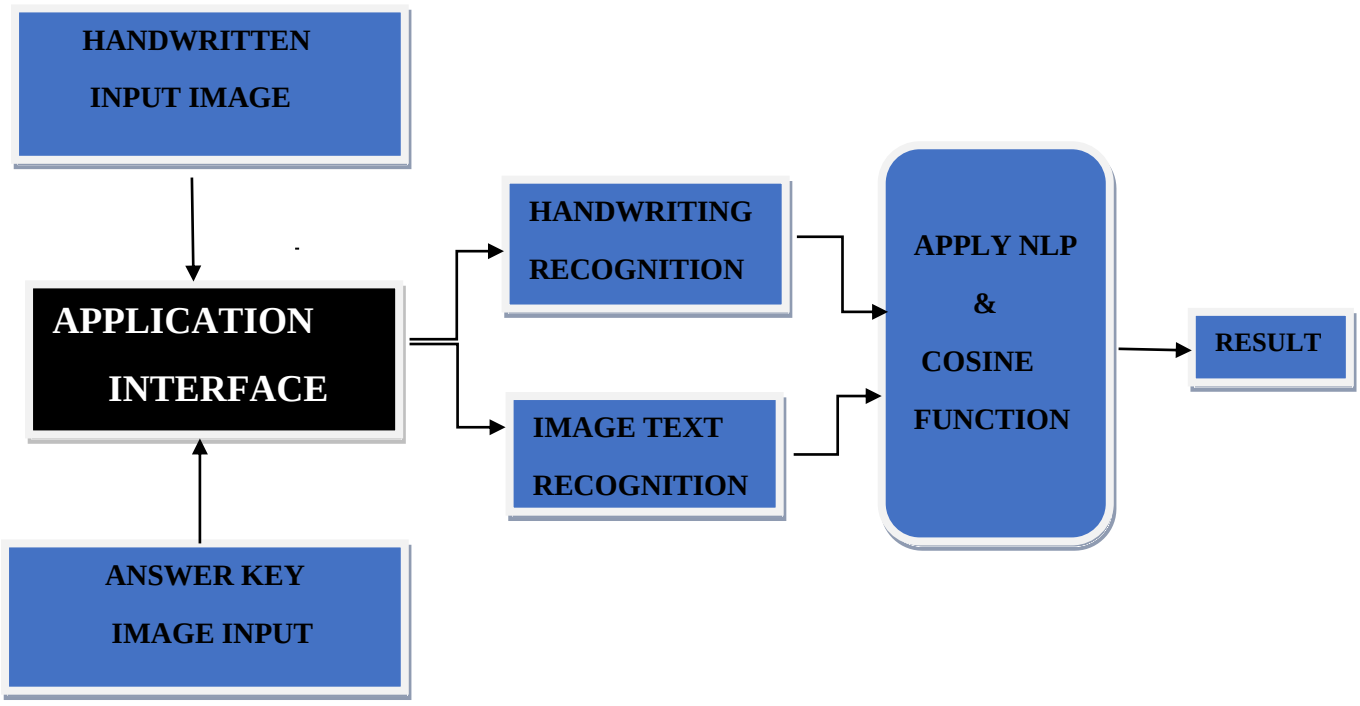


Fig 4.1 System Architecture

4.4 MODULES

4.4.1 TEXT EXTRACTION MODULE

Pytesseract or Python-tesseract is an Optical Character Recognition (OCR) tool for python. It will read and recognize the text in images, license plates, etc. Here, we will use the tesseract package to read the text from the given image.

Mainly, 3 simple steps are involved here as shown below:-

- Loading an Image saved from the computer or download it using a browser and then loading the same. (Any Image with Text).
- Binarizing the Image (Converting Image to Binary).
- We will then Pass the Image through the OCR system.

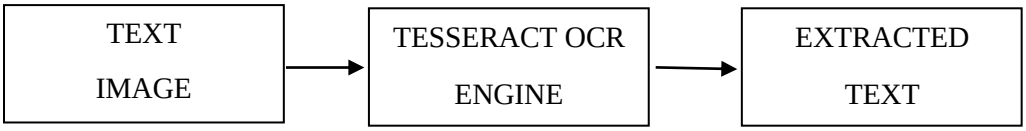


Fig 4.2 Flow diagram of Text Extraction Module

4.4.2 HANDWRITTEN TEXT EXTRACTION MODULE

Google Cloud Platform is a suite of public cloud computing services offered by Google. The platform includes a range of hosted services for compute, storage and application development that run on Google hardware. Google Cloud Platform services can be accessed by software developers, cloud administrators and other enterprise IT professionals over the public internet or through a dedicated network connection.

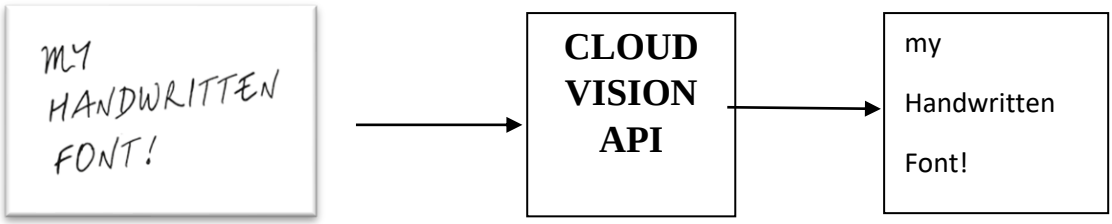


Fig 4.3 Flow Diagram for handwritten text extraction

For artificial intelligence (AI), Google offers its Cloud Machine Learning Engine, a managed service that enables users to build and train machine learning models. Various APIs are also available for the translation and analysis of speech, text, images and videos.

A project organizes all your Google Cloud resources. A project consists of the following components:

- a set of collaborators
- enabled APIs (and other resources)
- monitoring tools

- billing information
- authentication and access controls

To set up a Cloud Console project.

- Create or select a project.
- Enable the Vision API for that project.
- Create a service account.
- Download a private key as JSON.

To use services provided by Google Cloud, you must create a project.

Create a service account and download the private key file

Create a service account:

1. In the Cloud Console, go to the Create service account page.
2. Select a project.
3. In the Service account name field, enter a name. The Cloud Console fills in the Service account ID field based on this name.

   In the Service account description field, enter a description. For example, Service account for quick start.
4. Click Create.
5. Click the Select a role field.

   Under Quick access, click Basic, then click Owner.
6. Click Continue.
7. Click Done to finish creating the service account.

   Do not close your browser window. You will use it in the next step.

Create a service account key:

1. In the Cloud Console, click the email address for the service account that you created.
2. Click Keys.

3. Click Add key, then click Create new key.
4. Click Create. A JSON key file is downloaded to your computer.
5. Click Close.

Use the service account key file in your environment

Provide authentication credentials to your application code by setting environment variable GOOGLE_APPLICATION_CREDENTIAL. Replace [PATH] with the file path of the JSON file that contains your service account key.

The Vision API can detect and extract text from images:

DOCUMENT_TEXT_DETECTION extracts text from an image (or file); the response is optimized for dense text and documents. The JSON includes page, block, paragraph, word, and break information.

4.4.3 COMPARISON MODULE

Implementation of this Module using cosine similarity function. It is useful in classifying data on the number of objects that have a certain similarity, as research clustering based on cosine similarity measure.

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. Similarity = (A.B) / (||A||. ||B||) where A and B are vectors.
Cosine similarity and nltk toolkit module are used in this program. To execute this program nltk must be installed in your system. In order to install nltk module follow the steps below –
1. Open terminal.
2. pip install nltk
3. python3
4. import nltk
5. nltk.download('all')

Functions used:

nltk.tokenize: It is used for tokenization. Tokenization is the process by which big quantity of text is divided into smaller parts called tokens. word_tokenize(X) split the given sentence X into words and return list.

nltk.corpus: In this program, it is used to get a list of stopwords. A stop word is a commonly used word (such as “the”, “a”, “an”, “in”).

Then, use cosine_similarity() to get the final output. It can take the document term matrix as a pandas data frame as well as a sparse matrix as inputs.

4.5 METHODOLOGY

Agile Methodology for System Application Development is one needs to understand the process of development. Any application installed on a smartphone, a game or a social networking application requires many factors to be in place for its proper functioning. The factors which need to be in place are Idea of the application, Design, Development, Execution and Testing. Mobile application development is the product of a large amount of the work done by specialists from different fields, and most importantly the constant negotiation between the user. Any software, including mobile applications, can be considered as an outcome of good communication. Agile evaluation app development methodology is one of the most effective approaches to all the software development businesses, it ensures a proper channel of communication, which helps both the clients and App Developers execute the desired mobile application or in fact any software. The characteristics of Agile Methodology make an easy job for handwritten answer evaluation application development so that the mobile application outcome is adaptable after its release. This process is completely sequential, only after a step is completed next step will be implemented, a developer cannot go back to a previous step. There is absolutely no room for errors or changes, so a project outcome and an extensive plan must be set in the beginning and then it should be followed carefully.

Fig 4.4 Methodology – Agile

Agile will be very much useful when you need rapid production rather that the quality of the product. When it comes to the role of using agile project management on mobile application development then one needs to take into account. Fig 4.4 represents the flow diagram of Agile Methodology [17]

Thus, the use of agile and scrum methodology includes programming, development and project management with breakdown of the software development life cycle into smaller modules. The Agile methodology is an incremental and iterative mobile application development approach, where the complete app development process cycle is divided into multiple sub-modules, considered as mini-projects.

4.6 UML DIAGRAM

UML stands for Unified Modeling Language. The Unified Modeling Language (UML) is a general-purpose, develop modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

The purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

The Creation of UML was originally motivated by the desire to standardize the disparate notational systems and approaches to software design. It was developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in 1994-1995, with further development led by them through 1996.

In 1997, UML was adopted as a standard by the Object Management Group (OMG), and has been managed by this organization ever since. In 2005, UML was also published by the International Organization for Standardization (ISO) as an approved ISO standard. Since then the standard has been periodically revised to cover the latest revision of UML.

4.6.1 USECASE DIAGRAM

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components a user or another system that will interact with the system modelled. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

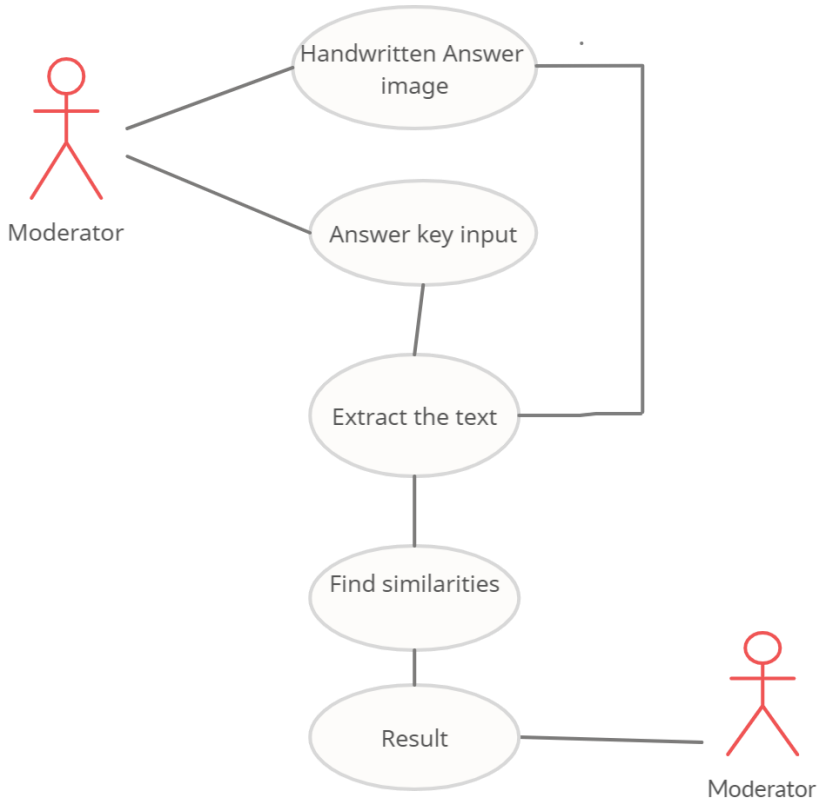


Fig 4.5 Use case Diagram

4.6.2 SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

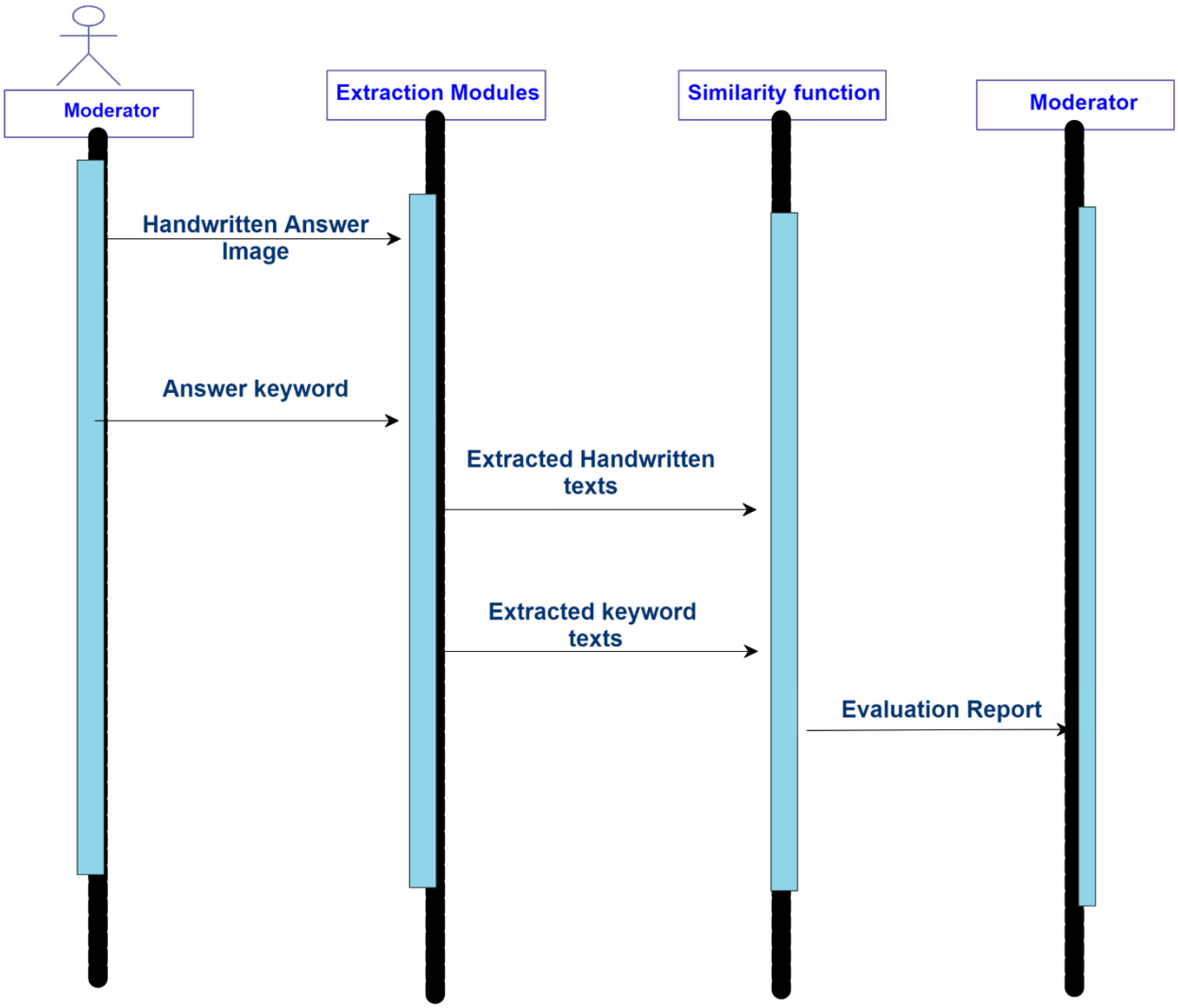


Fig 4.6 Sequence Diagram

4.6.3 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system.

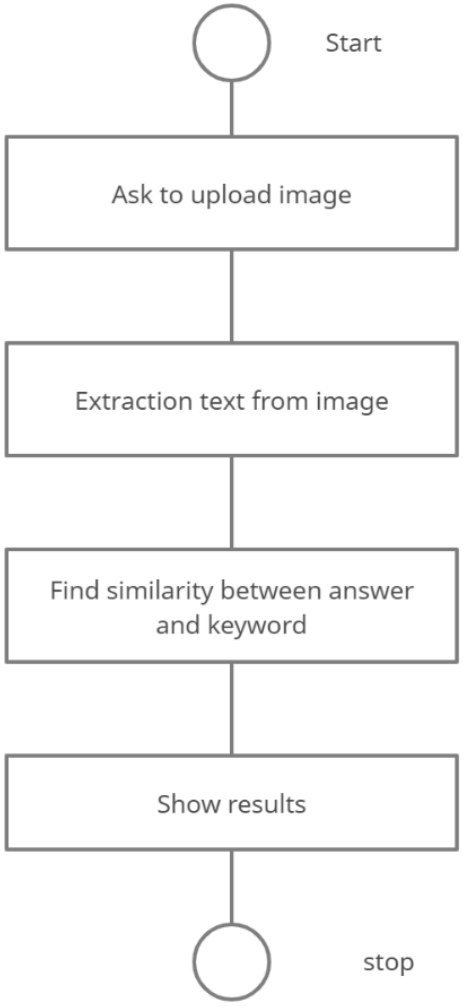


Fig 4.7 Activity Diagram

4.7 SUMMARY

The architecture of the proposed system and each module of the system is briefly explained in this chapter. The UML representations of the system also discussed clearly. So that you can understand the clear explanation of the project.

CHAPTER 5
RESULTS AND DISCUSSION

5.1 GENERAL

This chapter provides the result that has been obtained after implementing the proposed system. In order to thoroughly test the system, the proposed system has undergone a number of different testing strategies. Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data. It also gives a clear idea regarding different types of testing.

5.2 TESTING

Testing is a valid to the success of the system. This phase follows the coding phase. Testing takes logical assumption in a consideration. If all parts of the system are properly working the goals will be automatically achieved. There are different types of testing namely,

- Unit testing
- Integration testing
- White Box testing
- Black Box testing

5.2.1 UNIT TESTING

In unit testing, the design of the test cases is involved that helps in the validation of the internal program logic. The validation of all three modules and internal code takes place. After the completion of text extraction module and handwritten text extraction is completed, it takes place. Plus, it is taken into the comparison module after the all the three united is completed before integration. The unit test thus performs the basic level test at its component stage and test the particular business process, system configurations etc. The unit test ensures that the particular unique path of the process gets performed precisely to the documented specifications and

contains clearly defined inputs with the results which are expected in both image text and handwritten text extraction.

5.2.2 INTEGRATION TESTING

These tests are designed to test the integrated software items to determine whether if they really execute as a single program or application. The testing is event driven and thus is concerned with the basic outcome of field. The Integration tests demonstrate that the components were individually satisfaction, as already represented by successful unit testing, the components are apt and fine. After completion of extraction of texts we integrate the two extracted text into the comparison module. And it shows the correct comparison between the original extracted text and give results. Hence this test will passed successful. This type of testing is specially aimed to expose the issues that come-up by the component's combination.

5.2.3 WHITE BOX TESTING

The white box testing is the type of testing in which the internal components of the system software is open and can be processed by the tester. It is therefore a complex type of testing process. All the nltk, tesseract ocr, are tested by the tester himself to find out a possible bug or error. It is used in situation in which the black box is incapable of finding out a bug. It is a complex type of testing which takes more time to get applied.

5.2.4 BLACK BOX TESTING

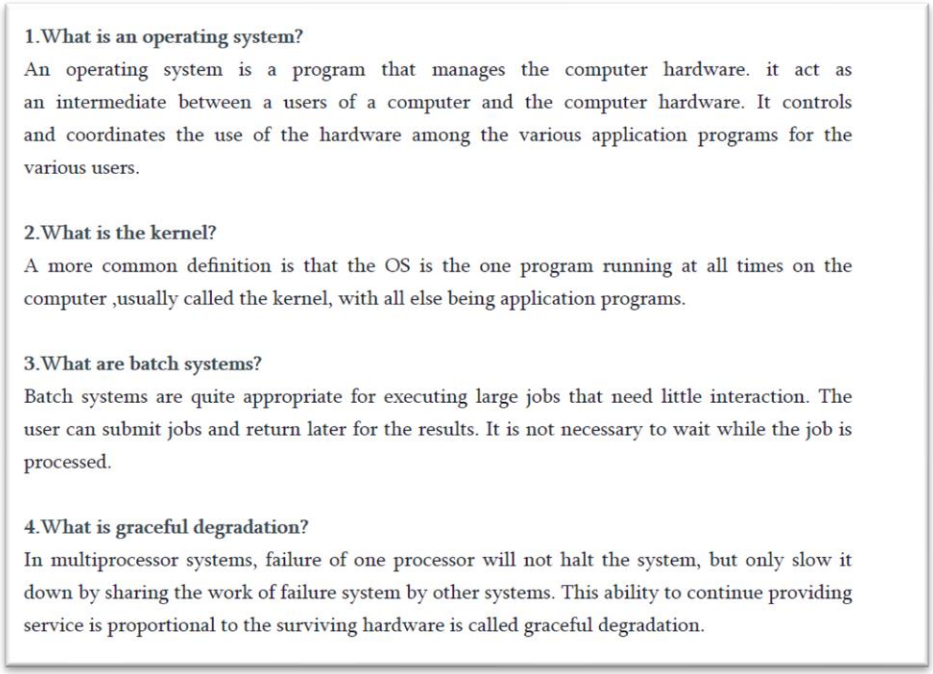
The black box testing is the type of testing in which the internal components of the software is hidden and only the input and output of the system is the key for the tester to find out a bug. It is therefore a simple type of testing. A programmer with basic knowledge can also process this type of testing. It is less time consuming as compared to the white box testing. It is very successful for software which are less complex are straight-forward in nature. It is also less costly than white box testing.

5.3 SCREENSHOTS

INPUT: The input is given to the application interface is

(1) Answer key Text image
(2) Handwritten Answer images

The both should be in jpeg and png format and additionally not exceeding the image size more than 2MB. The uploading images should be clear and good quality in order to get better recognition and extraction hence final result will more accuracy.



1.What is an operating system?
An operating system is a program that manages the computer hardware. it act as an intermediate between a users of a computer and the computer hardware. It controls and coordinates the use of the hardware among the various application programs for the various users.

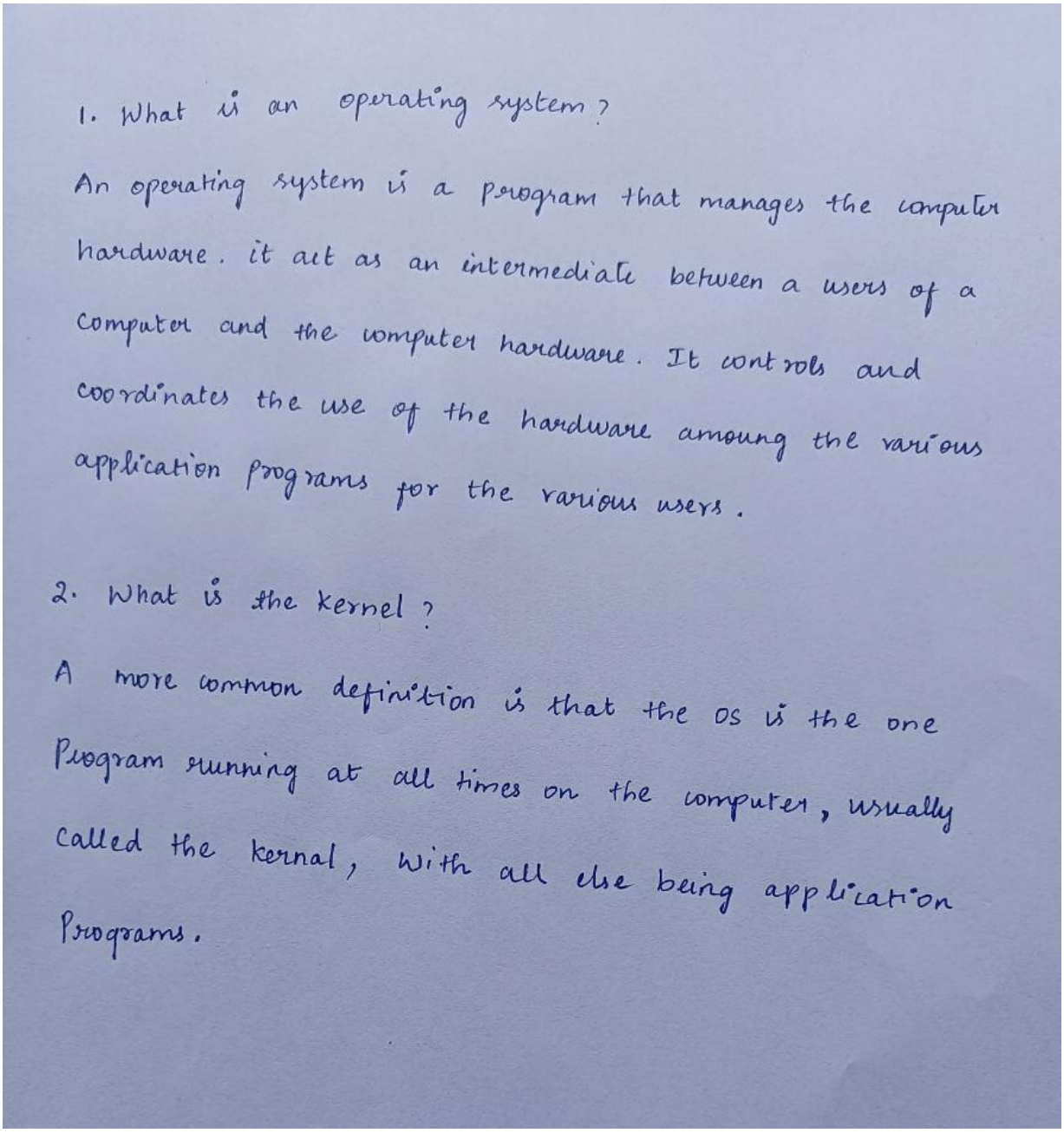
2.What is the kernel?
A more common definition is that the OS is the one program running at all times on the computer ,usually called the kernel, with all else being application programs.

3.What are batch systems?
Batch systems are quite appropriate for executing large jobs that need little interaction. The user can submit jobs and return later for the results. It is not necessary to wait while the job is processed.

4.What is graceful degradation?
In multiprocessor systems, failure of one processor will not halt the system, but only slow it down by sharing the work of failure system by other systems. This ability to continue providing service is proportional to the surviving hardware is called graceful degradation.

Fig 5.1 Keyword Text Image

In this above Fig 5.1 represents the answer key in the form of image, in order to extract the text using tesseract ocr.


1. What is an operating system?

An operating system is a program that manages the computer hardware. it act as an intermediate between a users of a computer and the computer hardware. It controls and coordinates the use of the hardware amoung the various application programs for the various users.

2. What is the kernel?

A more common definition is that the OS is the one Program running at all times on the computer, usually called the kernal, with all else being application Programs.

Fig. 5.2 Student Answer Page01

The above fig 5.2 represents the handwritten answer sheet of student which is in handwriting word.

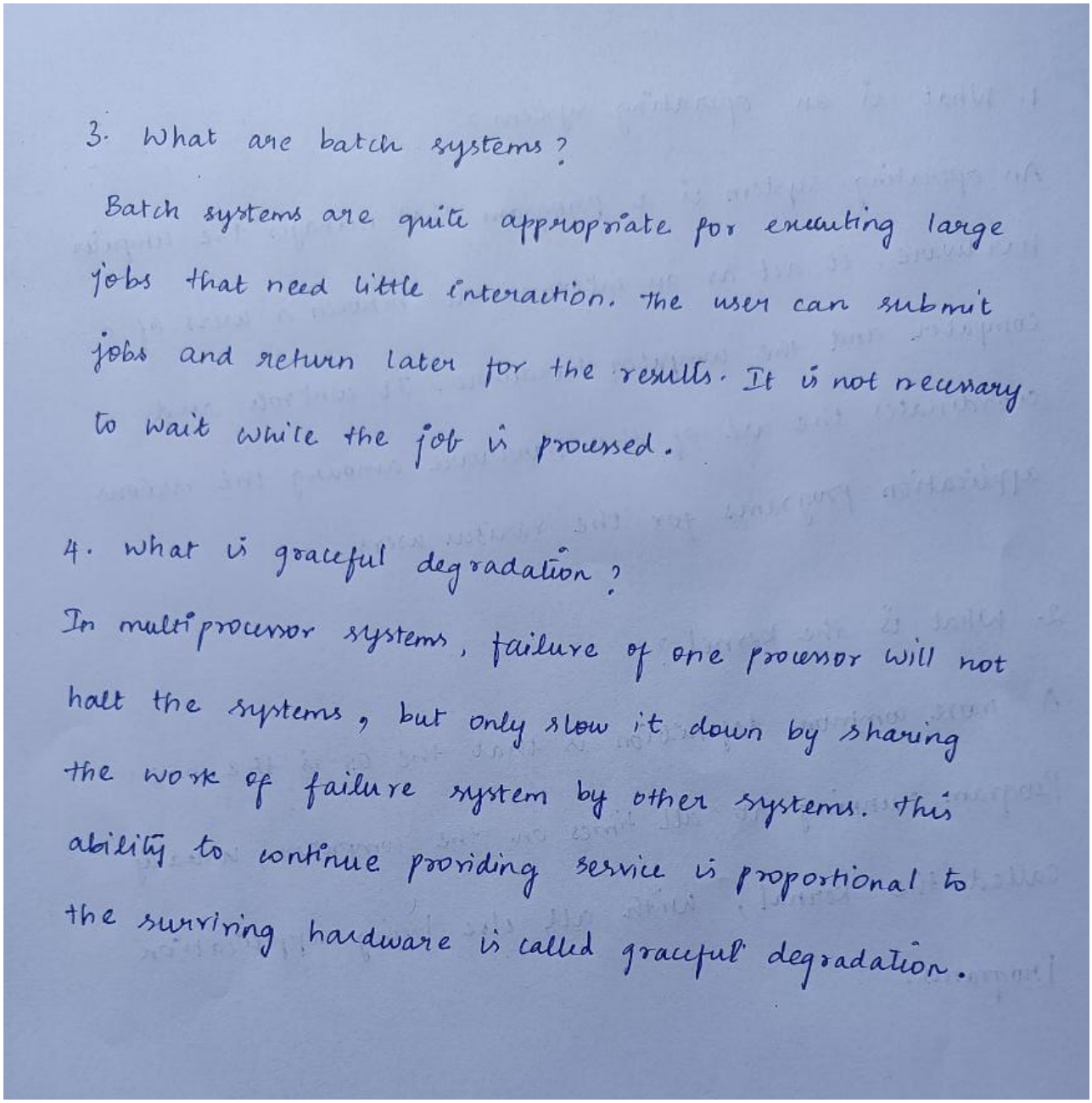


Fig 5.3 Student Answer Page02

In this above Fig 5.2 and Fig 5.3 represents the handwritten answer of students which is related to answer key image.

OUTPUT : Output screen of the each module is shown here, initially our application interface shown a page to ask answer key input using open file button shown in Fig 5.4., the output Screen is build using Tkinter Graphical user Interface.

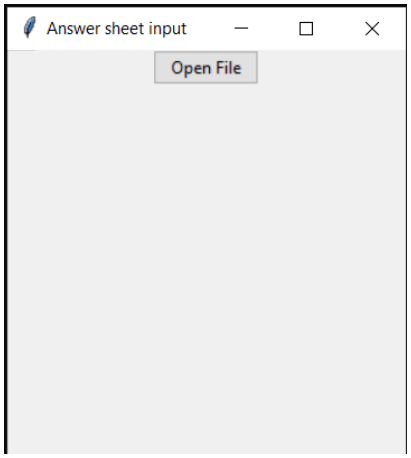


Fig 5.4 Output Home Screen

Once after running the program, the GUI dialog box appears, and it displays the open file button.

When we click the open file button it redirects to select a file page where all the handwritten answer images to choose the image to upload shown in fig 5.5.

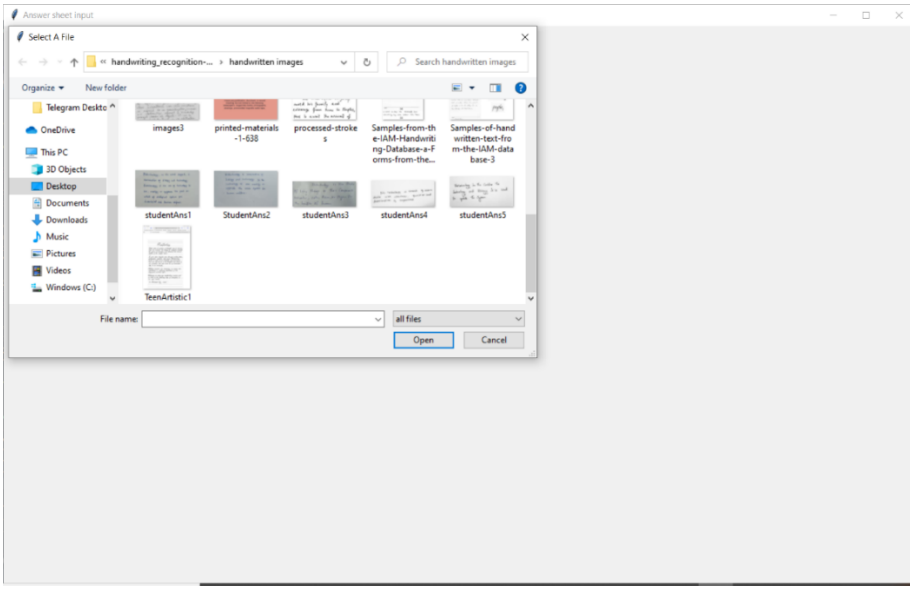


Fig 5.5 Upload the input image

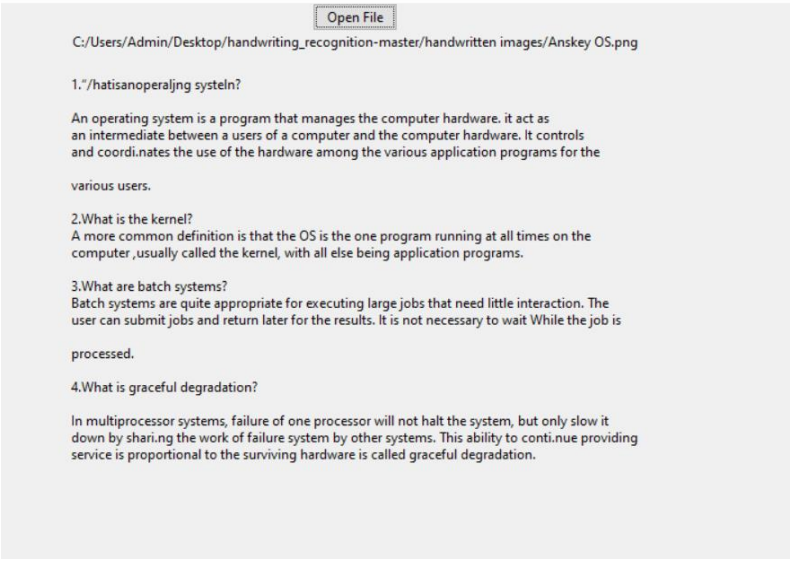


Fig 5.6 Keyword Text Extraction

After upload the image it shows the extracted text in GUI screen. In this part answerkey text image has converted into digital text shown in fig 5.6.

The next process is to get a students handwritten answer sheet and extract the handwritten text into digital text. In this module the screen heading will be Answer Sheet Input .

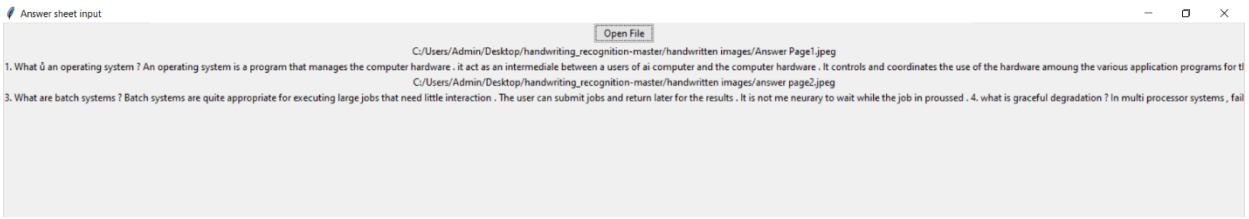


Fig 5.7 Handwritten Text Extraction

After we upload the image then show the extracted the handwritten text form an image file shown in Fig 5.6.

Finally using the both extracted text applying NLP and Cosine Similarity function we display the result, in AnswerSheet Evaluation page contain two button one is Results and another one is Exit as shown in Fig 5.7.

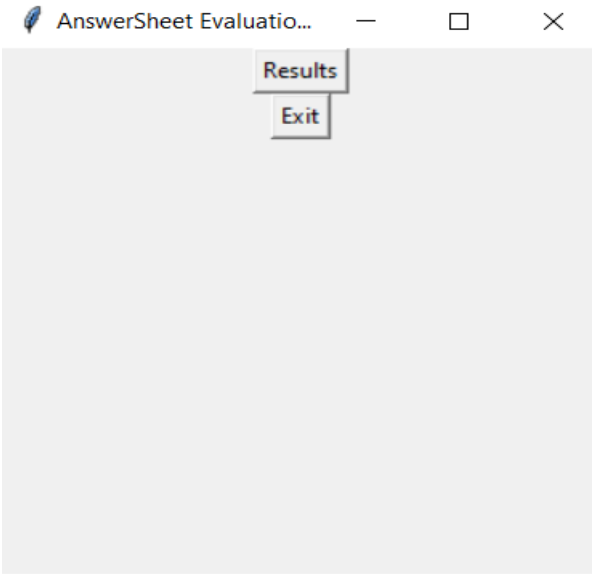


Fig 5.8 AnswerSheet Evaluation Screen

If we press the Results button it show the mark the student get and show the student result status whether they pass or fail shown in Fig 5.9.

Fig 5.9 Results Screen

Additionally the info button will show the Total number of words written, and total number of words matched with keyword and student answers and also show any grammatical mistakes shown in Fig 5.10.

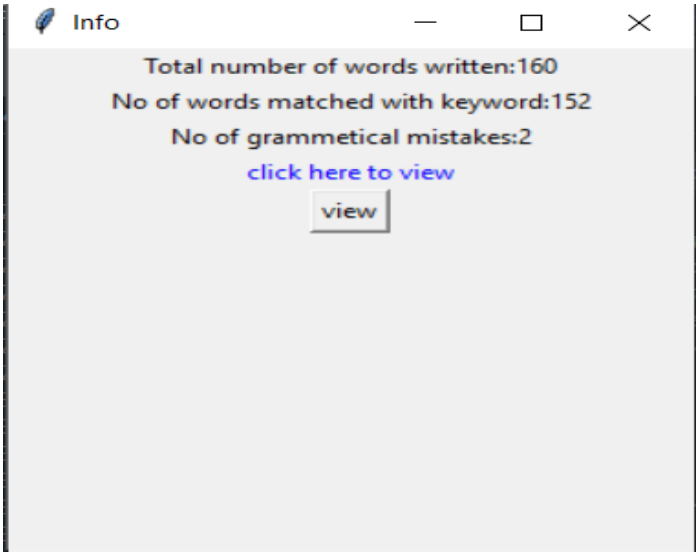


Fig 5.10 Info Screen

5.4 RESULTS

Our Proposed System is about 98 percent accurate with comparison to manual system. It tested in different students handwritten fonts and different kind of answers. We get a result as nearly compared to human evaluators. And it eliminates the human effort and time to evaluate an answer.

In image text extraction using pytesseract OCR we get an accuracy level at 100%. But in handwritten image OCR not provide an accurate results. So, in image handwritten text extraction using google Vision API we get an finite accuracy at various type of handwritten texts and cursive handwriting.

Sample Input Images

The different handwritten answers images collected from Students and also each answer contain different content of Answers.

From fig 5.8 to fig 5.12 are five different answer from the five different handwritten sheets from different handwriting and different weighted answers as we collected and tested to show the comparison between our proposed system with manual checking.

Table 5.1 Compare between our system evaluation and manual evaluation

Sample input	**No of words**	**No of Keyword matching**	**Marks provide by our application**	**Marks allocate By Human evaluator**
Sample 1	160	152	100	100
Sample 2	283	263	98	96
Sample 3	156	123	85	80
Sample 4	233	150	70	72
Sample 5	213	40	32	35

Thus the table 5.1 illustrates that the parameters like no of words, No of keyword matching and mark provided by our application and human evaluator.

We also tested our comparsion module with the various number of questions its shows the better result as similar to human evaluator

Table 5.2 Comparison between Manual checking and Proposed System Checking based on different number of questions

Sample No	**Total number of questions**	**Number of questions student answer**	**Mark provided by our application**	**Mark provided by Human evaluator**
Sample 1	5	5	100	100
Sample 2	5	4	80	80
Sample 3	5	3.5	70	70
Sample 4	5	2	40	40
Sample 5	5	0	0	0

According to Table 5.2, the results are compared. The scores are calculated for 5 different sample. In this table, fourth and fifth column has evaluated scores based on our application system and manual checking. The difference between the manual evaluation and system evaluation are exactly same.

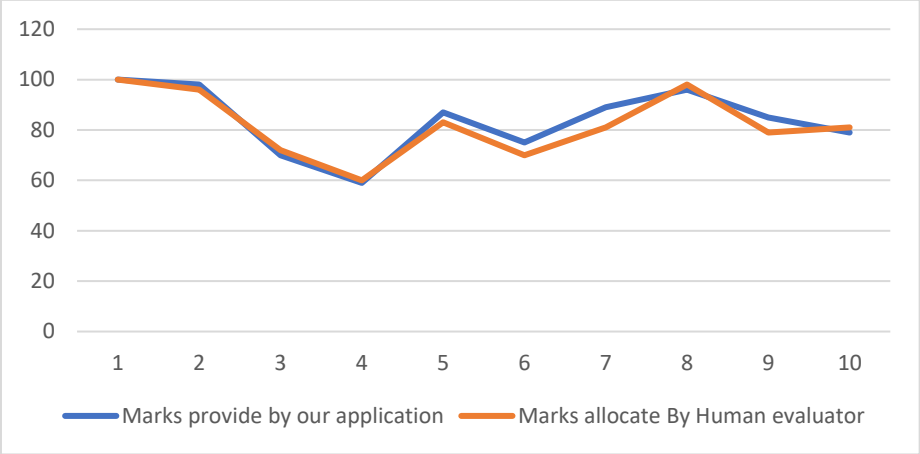


Fig 5.11 Chart for Comparison

From the Fig. 5.11. Mean absolute error is calculated between the marks calculated by our proposed algorithm and the evaluated samples that were evaluated by the moderator. It is calculated by taking difference between marks calculated by our proposed system and the marks calculated by moderator evaluated answer samples. The absolute error for the system is 1.9% compared to manual evaluation. Hence, it can be concluded that accuracy of 98% is achieved in comparison to manual evaluation calculated for the given sample of 5 questions and answers.

Finally in comparison module we compare our scoring with manual evaluator, our system evaluation yields better results than human evaluator.

5.5 SUMMARY

Thus, the working process of the project was illustrated using the screen shots. Screen shots gave a clear understanding regarding the working of the proposed system Input and output details of the project execution.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 GENERAL

In the above chapters we have discussed about the results and output. Here we are going to conclude and give the future enhancements to develop the project.

6.2 CONCLUSION

Thus the application provides the better result of automated evaluation process as compared to human evaluator. Therefore by our application reduce the manpower and human error in evaluation process. Text extraction from image files is an useful technique for document digitalization. There are several well-developed OCR engines for printed text extraction, such as Tesseract and EasyOCR. However, for handwritten text extraction, it's more Tesseract and EasyOCR can't achieve satisfying results unless the texts are hand-printed. In this post, I will describe how to use Tesseract to extract printed texts, and use Google Cloud Vision API to extract handwritten texts.

We will use the Cosine Similarity from scikit-learn, as the metric to compute the similarity between two vectors. Cosine similarity is a metric used to measure how similar two items are. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space.

6.3 FUTURE ENHANCEMENT

Further more in future, there is a possibility of producing better results in automated handwritten answer sheet evaluation system by focusing and adding many segmentation modules and semantic word analysis using genism library in machine learning.

APPENDICES

SOURCE CODE

1.1 Source code for text extraction module

textread.py

```
import pytesseract
import tkinter
import tkinter.messagebox
from tkinter import *
from tkinter.ttk import *
from tkinter.filedialog import askopenfile
from PIL import ImageTk,Image
from PIL import Image


root = Tk()
root.title('Answer key input')
v = Scrollbar(root, orient='vertical')
v.config(command=root.yview)


def open1():
    global my_image
root.filename =
filedialog.askopenfilename(initialdir="C:/Users/Admin/Desktop/handwriting_recog
nition-master/handwritten images", title="Select A File",filetypes=(("PNG
files","*.PNG"),("all files","*.*")))
my_label = Label(root,text=root.filename).pack()
my_image = ImageTk.PhotoImage(Image.open(root.filename))
my_image_label = Label(image=my_image).pack()
```

```
img = Image.open(root.filename)


pytesseract.pytesseract.tesseract_cmd ='C:/Program Files (x86)/Tesseract-OCR/tesseract.exe'
    result = pytesseract.image_to_string(img)
    print(result)
    Label(root,text=result).pack()
# write text in a text file and save it to source path
    with open('abc.txt',mode='w') as file:


file.write(result)
                print(result)
my_btn = Button(root,text="Open File", command=open1)
my_btn.pack()
root.mainloop()
```

1.2 Source code for handwritten text extraction module

handwritingread.py

```
import tkinter
import tkinter.messagebox
from tkinter import *
from tkinter.ttk import *

# importing askopenfile function
# from class filedialog
from tkinter.filedialog import askopenfilename
from PIL import ImageTk,Image
```

```
from google.cloud import vision
import io
import os
root = Tk()
root.title("Answer sheet input")
root.geometry('1300x1300')
root.resizable(width = True, height = True)
path=r"C:\Users\Admin\Desktop\handwriting_recognition-master\handwritten images\images1.jpg"
def open1():
    global my_image
root.filename = askopenfilename(initialdir="C:/Users/Admin/Desktop/handwriting_recognition-master/handwritten images", title="Select A File",filetypes=(("PNG files","*.PNG"),("all files","*.*")))
my_label = Label(root,text=root.filename).pack()
my_image = ImageTk.PhotoImage(Image.open(root.filename))
my_image_label = Label(image=my_image).pack()
credential_path =r"C:\Users\Admin\.spyder-py3\VisionAPIDEMO\handwritten-project-82cff1ea3561.json"
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = credential_path
    client = vision.ImageAnnotatorClient()
    with io.open(root.filename, 'rb') as image_file:
        content = image_file.read()
    image = vision.Image(content=content)
    response = client.document_text_detection(image=image)
    text=[]
    for page in response.full_text_annotation.pages:
        for block in page.blocks:
```

```
            # print('\nBlock confidence: {}\n'.format(block.confidence))
             for paragraph in block.paragraphs:
                # print('Paragraph confidence: {}'.format(
                  #  paragraph.confidence))
                 for word in paragraph.words:
word_text = ''.join([
symbol.text for symbol in word.symbols
                    ])
                    #print('Word text: {} (confidence: {})'.format(
                      #  word_text, word.confidence))
                    #print(word_text,end=" ")
                    Label(root,text=word_text).pack()
text.append(word_text)
                    #for symbol in word.symbols:
                       # print('\tSymbol: {} (confidence: {})'.format(
                           # symbol.text, symbol.confidence))
listToStr = ' '.join([str(elem) for elem in text])
   with open('abc1.txt',mode='w') as file:
file.write(listToStr)
            print(listToStr)
   if response.error.message:
        raise Exception(
            '{}\nFor more info on error messages, check: '
            'https://cloud.google.com/apis/design/errors'.format(
response.error.message))
"""Detects document features in an image."""
my_btn = Button(root,text="Open File", command=open1)
```

```
my_btn.pack()
root.mainloop()
```

1.3 Source code for comparsion module

comparison.py

```
import numpy as np
#import cosine_similarity
import tkinter
import tkinter.messagebox
from  tkinter import *
#from tkinter.ttk import *
from decimal import Decimal
myfile = open("abc.txt")
txt = myfile.read()
myfile1 = open("ans3.txt")
txt1 = myfile1.read()

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
stopwords = set(stopwords.words('english'))
word_tokens = word_tokenize(txt)
filtered_Sentence=[]
for w in word_tokens:
    if w not in stopwords:
filtered_Sentence.append(w)
#print(word_tokens)
#print(filtered_Sentence)
```

```
word_tokens1 = word_tokenize(txt1)
filtered_Sentence1 =[]
for w in word_tokens1:
    if w not in stopwords:
        filtered_Sentence1.append(w)

#print(word_tokens1)
#print(filtered_Sentence1)
listToStr = ' '.join([str(elem) for elem in filtered_Sentence])
listToStr1 = ' '.join([str(elem) for elem in filtered_Sentence1])
documents = [listToStr, listToStr1]
#print(documents)
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(documents)
vectorizer.fit(documents)
vectors = vectorizer.transform(documents).toarray()
from sklearn.metrics.pairwise import cosine_similarity
print (cosine_similarity(vectors))
l = cosine_similarity(vectors)
print(l[0][1])
def show_marks():
att = Toplevel()
att.title('Results')
att.geometry('200x100')
    Marks = Decimal(l[0][1]*100).quantize(Decimal("1.0"))
    m = str(Marks)
    print (m)
```

```
    if (Marks>=70):
Label(att, text= "2m\t pass", fg = "green").pack()
elif (35<=Marks<70):
Label(att, text= "1m\t pass", fg = "green").pack()
    else:
Label(att, text= "0m\t fail", fg = "red").pack()
    button = Button(att, text = "exit", command = att.destroy)
button.pack()
def main():
    root = Tk()
root.title("AnswerSheet Evaluation App")
root.minsize(width=200, height=200)
root.maxsize(width=200, height=200)
    button = Button(root, text="Results", command = show_marks)
button.pack()
    button = Button(root, text="Exit", command = root.destroy)
button.pack()
root.mainloop()
if __name__ == "__main__":
main()
```

REFERENCES

[1] A. R.Lahitani, A. E. Permanasari and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," 2016 4th International Conference on Cyber and IT Service Management, Bandung, Indonesia, 2016, pp. 1-6, doi: 10.1109/CITSM.2016.7577578.

[2] Buddhiprabha Erabadda, Surangika Ranathunga and Gihan Dias, "Computer Aided Evaluation of Multi-Step Answers to Algebra Questions," IEEE Int. Conf. on Adv. Learning Technologies, pp. 45–65, vol. 28, 2016.

[3] C. Kaundilya, D. Chawla and Y. Chopra, "Automated Text Extraction from Images using OCR System," 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2019.

[4] Charusheela Nehete, Vasant Powar, Shivam Upadhyay and Jitesh Wadhwani, "Checkpoint –An Online Descriptive Answers Grading Tool," IJARCS, April 2017.

[5] D. Mulfari, A. Celesti, M. Fazio, M. Villari and A. Puliafito, "Using Google Cloud Vision in assistive technology scenarios," 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, 2016, pp. 214-219, doi: 10.1109/ISCC.2016.7543742

[6] D. Vaithiyanathan and M. Muniraj, "Cloud based Text extraction using Google Cloud Vison for Visually Impaired applications," 2019 11th International Conference on Advanced Computing (ICoAC), Chennai, India, 2019, pp. 90-96, doi: 10.1109/ICoAC48765.2019.246822.

[7] J. O. Contreras, S. Hilles and Z. B. Abubakar, "Automated Essay Scoring with Ontology based on Text Mining and NLTK tools," 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, Malaysia, 2018, pp. 1-6, doi: 10.1109/ICSCEE.2018.8538399.

[8] J. Talwar, S. Ranjani, Anwaya Aras, and M. Bedekar, Intelligent Classroom System for Qualitative Analysis of Students ‟ Conceptual Understanding", IEEE, 2013.

[9] Jannat Talwar, Shree Ranjani and Anwaya, "Intelligent Classroom System for Qualitative Analysis of Students‟ Conceptual Understanding," IEEE Int. Conf. on Emerging Trends in Engg. And Technology, pp. 105–125,vol. 28, 2013.

[10] Jamsheedh C. V, Aby Abahai T and Surekha Mariam Varghese, "A Fair Assessment System For Evaluation And Grading Of Text In Degitized Descriptive Answer Scripts," unpublished.

[11] K. P. N. V Satya and J. V. R. Murthy, "Clustering based on cosine similarity measure", no. 3, pp. 508-512, 2012.

[12] Magdi Z. Rashad, Ahmed E. Hassan, Mahmoud A. Zaher and Mahmoud S. Kandil, " An Arabic, Web-based Exam Management System ", IJECS – IJENS International Journal of Computer Sciences and Electricals ,February 2010.

[13] Mohammad Salim Ahmed, Fahad Bin Muhaya, Lathifur khan and Sourabh Jain, " Predicted Probability Enhancement for Multilabel Text Classification using Class-Label Pair Associations," IEEE Conference on Evolving and Adaptive Intelligent Systems, April 2013.

[14] Panchami K.S, Surekha Mariam Vargheseb and Aby Abhahai T, " Grading of Diagrams in Answer Scripts Using Support Vector Machine," International Journal of Control Theory and Applications, Vol.10,No.29,pp. 345-350, 2017.

[15] Philip E. Robinson and Johnson Carroll, "An Online Learning Platform for Teaching, Learning,and Assessment of Programming,'" IEEE Global Engineering Education Conference (EDUCON), vol. 3, no. 6, pp. 547-556, April 2017.

[16] Pantulkar Sravanthi and Dr. B. Srinivasu, "Semantic Similarity between Sentences," International Research Journal of Engineering and Technology (IR-JET) vol. 2, pp. 156-161, 2017.

[17] S. Soundararajan, J. D. Arthur and O. Balci, "A Methodology for Assessing Agile Software Development Methods," 2012 Agile Conference, Dallas, TX, USA, 2012, pp. 51-54, doi: 10.1109/Agile.2012.24.

[18] S. Srihari, J. Collins, R. Srihari, H. Srinivasan, S. Shetty,and J. Brutt-Griffler, “Automatic scoring of short handwrittenessays in reading comprehension tests,” 2008.

[19] Semire DIKLI and Tallahassee, "Automated Essay Scoring," Turkish Online Journal of Distance Education-TOJDE, Vol. 7, No. 1, Article: 5, January 2006.

[20] V. Rowtula, S. R. Oota and J. C.V, "Towards Automated Evaluation of Handwritten Assessments," 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, NSW, Australia, 2019, pp. 426-433, doi: 10.1109/ICDAR.2019.00075.