A Project Report On

### FAKE CURRENCY DETECTION USING DEEP LEARNING

Submitted in partial fulfillment of the requirements for the award of the degree of

### BACHELOR OF TECHNOLOGY

in

### INFORMATION TECHNOLOGY

Submitted By

|  |  |
| --- | --- |
| **M.DHANALAKSHMI** | **19P31A1223** |
| **A. KARTHIK** | **19P31A1202** |
| **P.RAGHUNADH** | **19P31A1243** |
| **ANJALI YADAV** | **19P31A1259** |

#### Under the esteemed supervision of

Mrs.N.Suryakala., M.Tech .,

**Assistant Professor**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

### ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

Permanently Affiliated to JNTUK, Kakinada \* Approved by AICTE New Delhi

Accredited by NBA, Accredited by NAAC ( A+ ) with 3.4 CGPA

Aditya Nagar, ADB Road, Surampalem , Kakinada District, Andhra Pradesh.

**2019-2023**

### ADITYA COLLEGE OF ENGINEERING &TECHNOLOGY

Permanently Affiliated to JNTUK, Kakinada \* Approved by AICTE New Delhi

Accredited by NBA, Accredited by NAAC ( A+ ) with 3.4 CGPA

Aditya Nagar, ADB Road, Surampalem , Kakinada District, Andhra Pradesh

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**CERTIFICATE**

This is to certify that the project work entitled “**FAKE CURRENCY DETECTION USING DEEP LEARNING”**, is a bonafide work carried out by **M.Dhanalakshmi (19P31A1223), A.Karthik (19P1A1202), P.Raghunadh(19P31A1243), Anjali yadav (19P31A1259)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology** from Aditya College of Engineering & Technology duringthe academic year 2018-2022.

|  |  |
| --- | --- |
| **PROJECT GUIDE** | **HEAD OF THE DEPARTMENT** |
| **Dr. B.Srinivas.,M.Tech., Ph.D.** | **Mr. R. V. V. N. Bheema Rao.,M.Tech.,(Ph.D)** |
| **Assistant Professor** | **Assistant Professor** |

**EXTERNAL EXAMINER**

# DECLARATION

We hereby declare that this project entitled “ **FAKE CURRENCY DETECTION USING DEEP LEARNING** ”, has been undertaken by us and this work has been submitted to **Aditya College of Engineering & Technology** affiliated to JNTUK, Kakinada, in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology**.

We further declare that this project work has not been submitted in full or part for the award of any degree of this or in any other educational institutions.

**Project Associates**

|  |  |  |
| --- | --- | --- |
| **M.DHANALAKSHMI** | **19P31A1223** |  |
| **A.KARTHIK** | **19P31A1202** |  |
| **P.RAGHUNADH** | **19P31A1243** |  |
| **ANJALI YADAV** | **19P31A1259** |  |

# ACKNOWLEDGEMENT

With immense pleasure that we would like to express our indebted gratitude to our Project Supervisor, **Mr. T. Srinivasulu,** for guidance and encouragement in every step of the project work, his valuable moral support and guidance throughout the project helped us to a great extent.

We wish to express our sincere thanks to **Mr. R.V.V.N Bheema Rao**, Head of the Department, Information Technology, for his valuable guidance throughout the period of the project work.

We feel elated to thank our Dean **Dr. A. Rama Krishna** for his cooperation and help in the completion of our project and throughout our course.

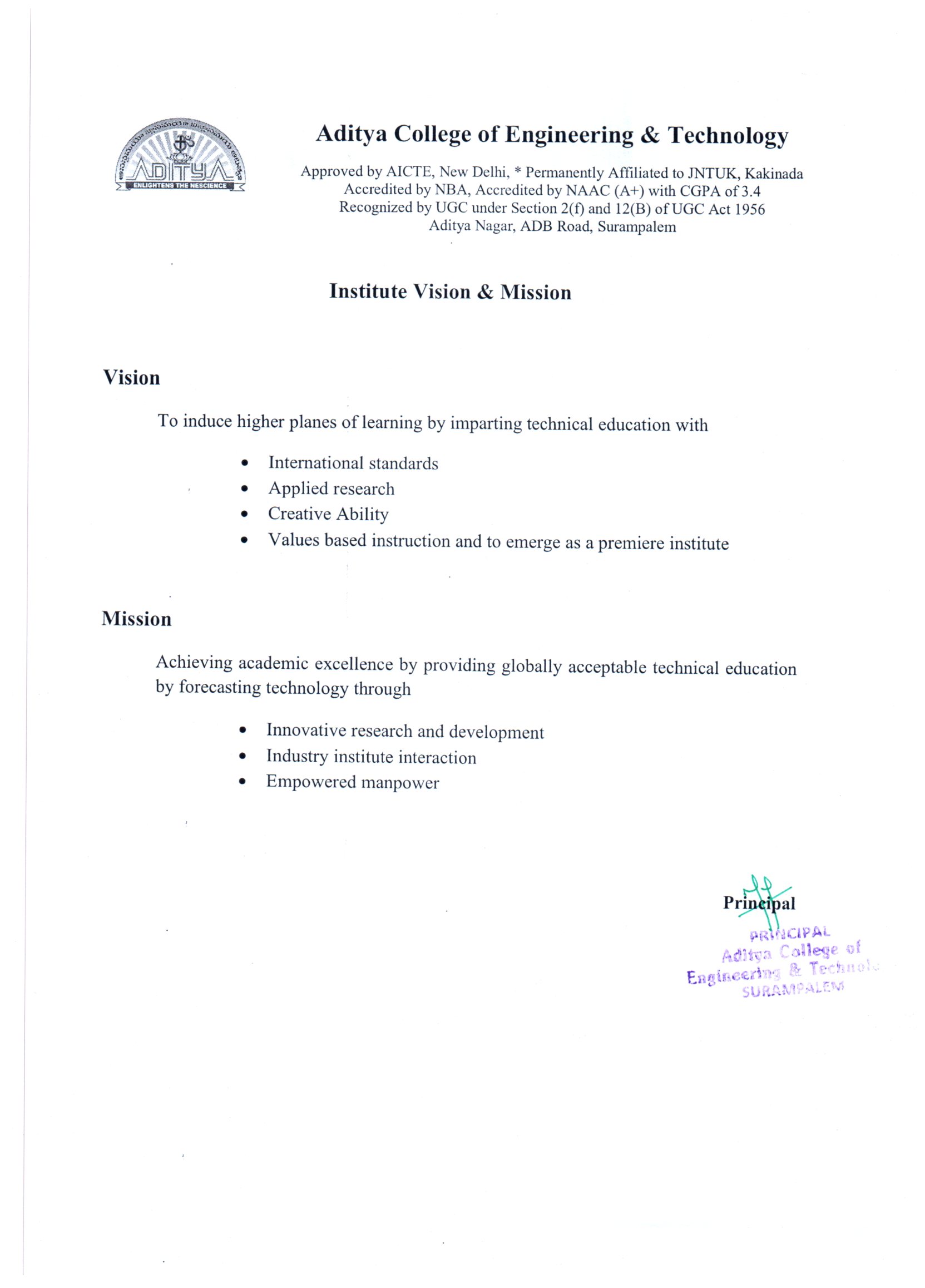
We feel elated to thank our Principal **Dr. Dola Sanjay S** for his cooperation and help in the completion of our project and throughout our course.

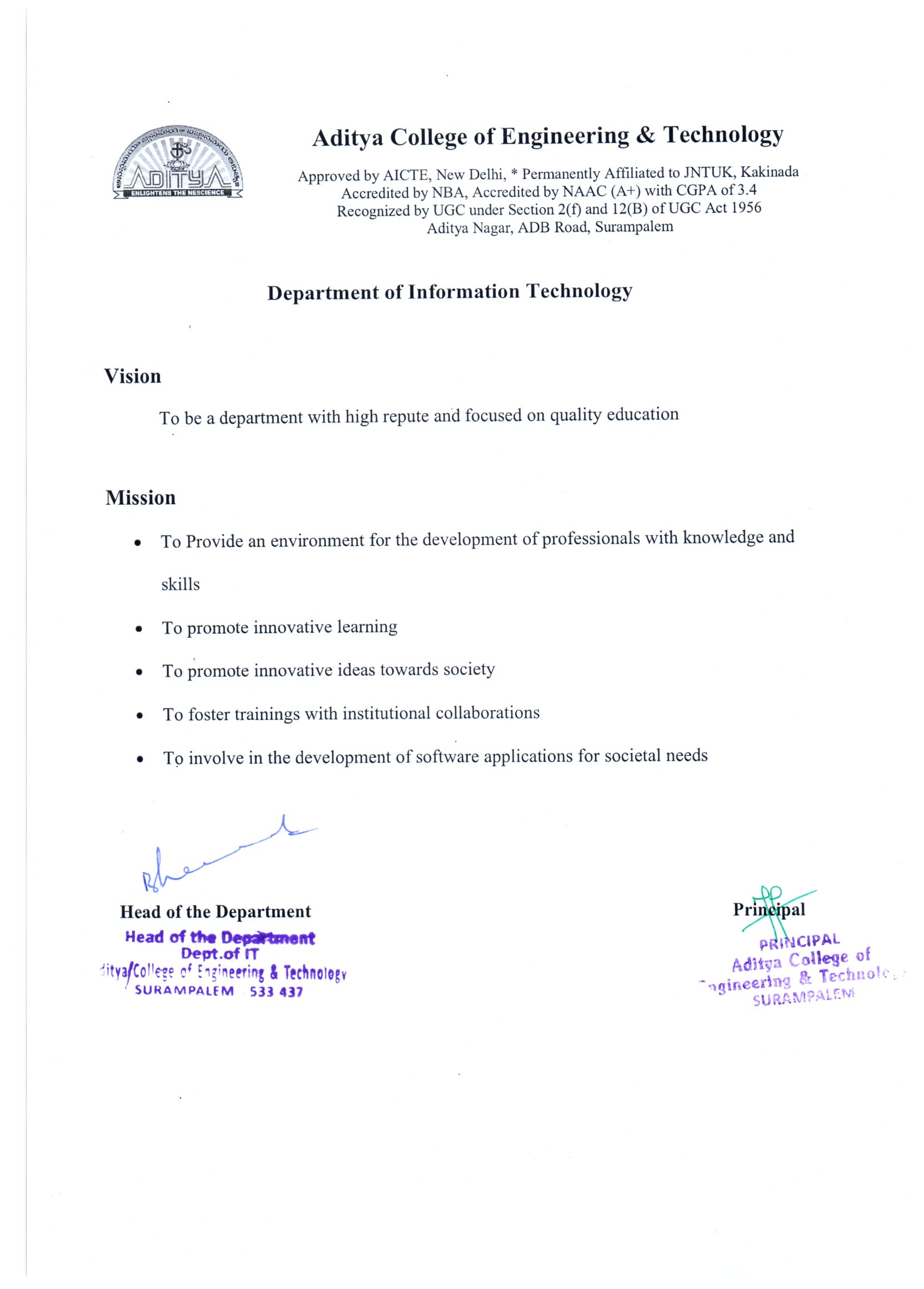
We wish to express our sincere thanks to all **faculty members, lab programmers** for their valuable guidance throughout the period of the project.

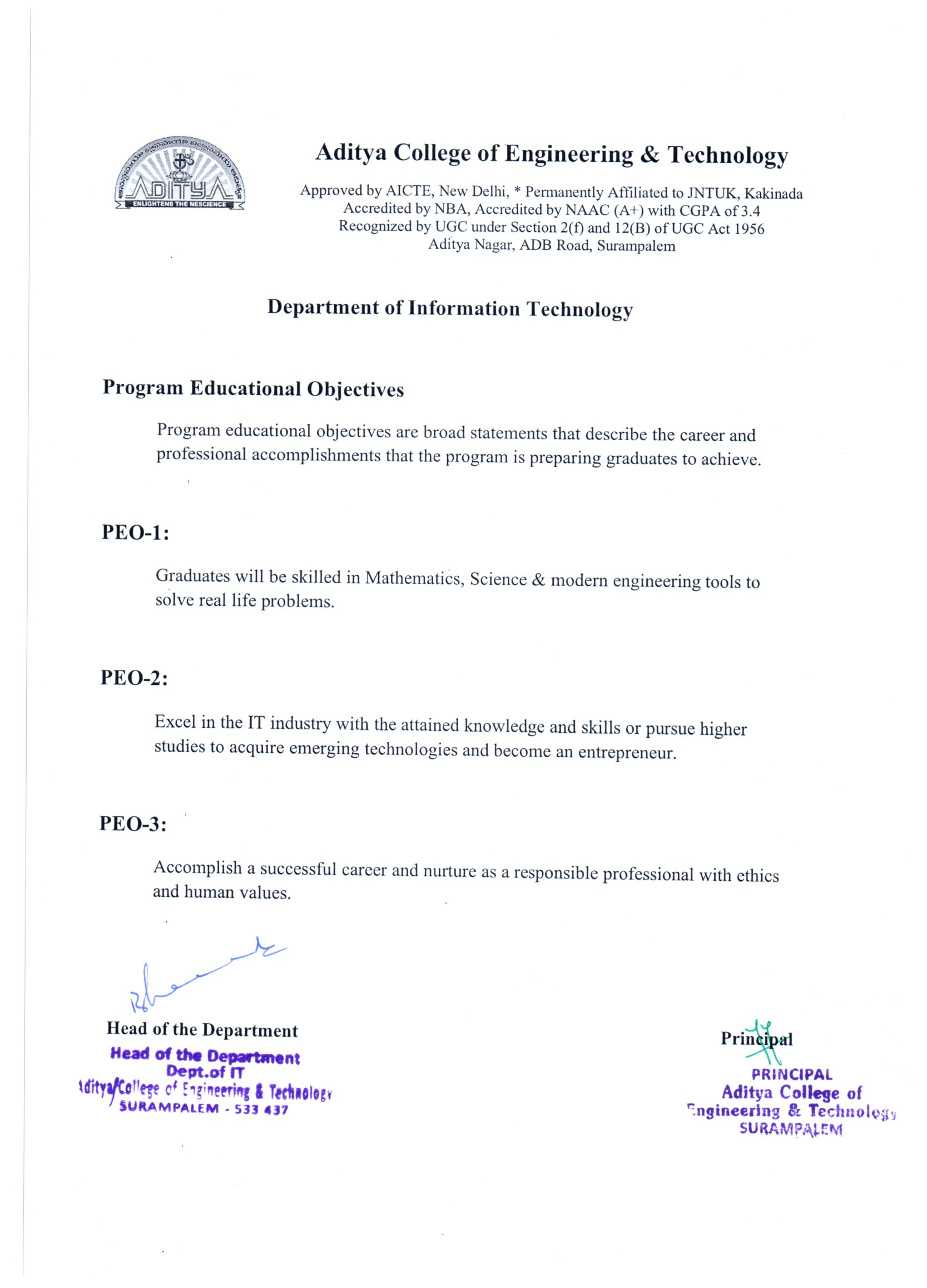
We avail this opportunity to express our deep sense and heart full thanks to the **Management** of **Aditya College Of Engineering & Technology** for providing wonderful infrastructure and facilities.

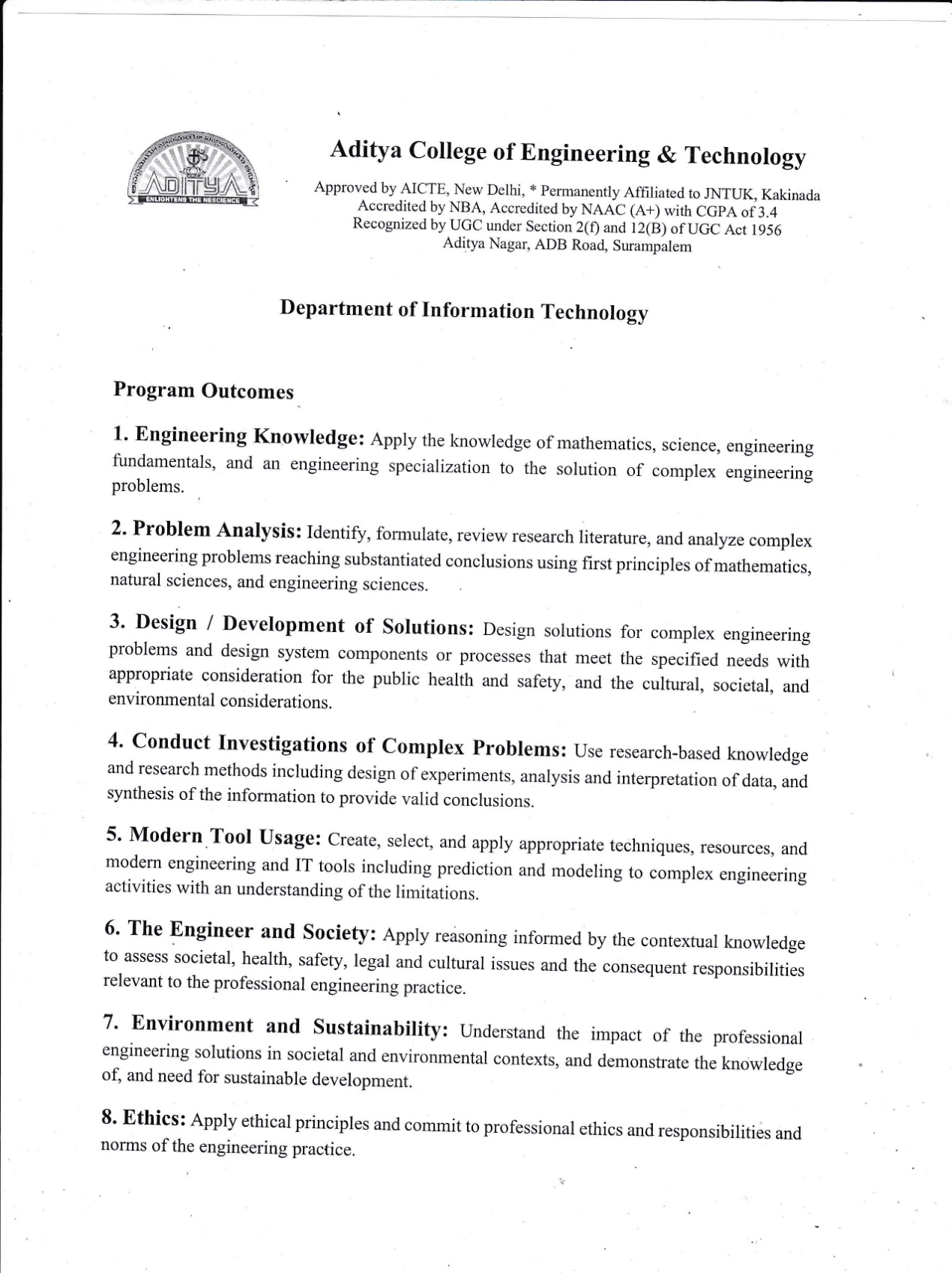
**Project Associates**

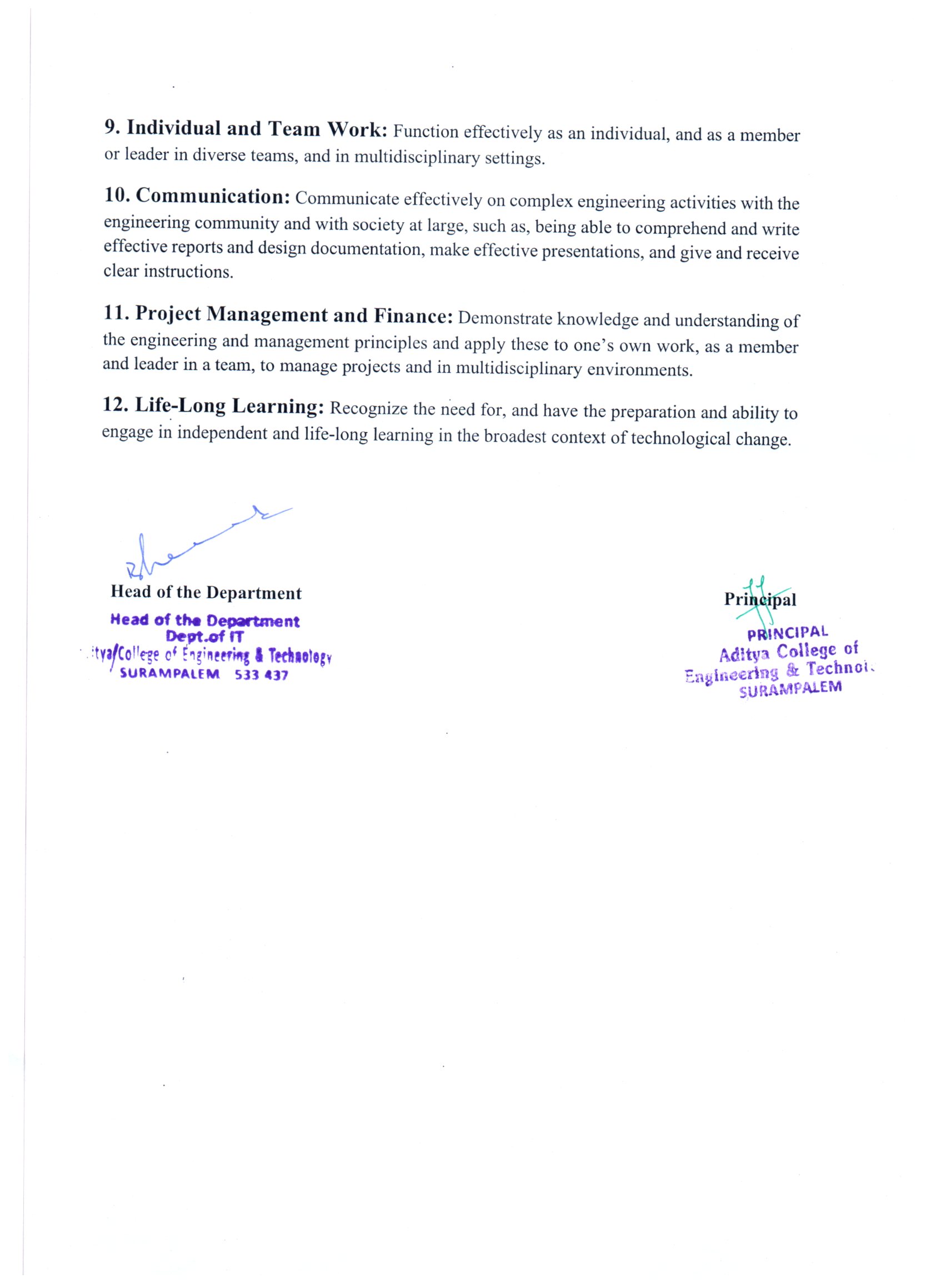
|  |  |
| --- | --- |
| **M.DHANALAKSHMI** | **(19P31A1223)** |
| **A.KARTHIK** | **(19P31A1202)** |
| **P.RAGHUNADH** | **(19P31A1243)** |
| **ANJALI YADAV** | **(19P31A1259)** |

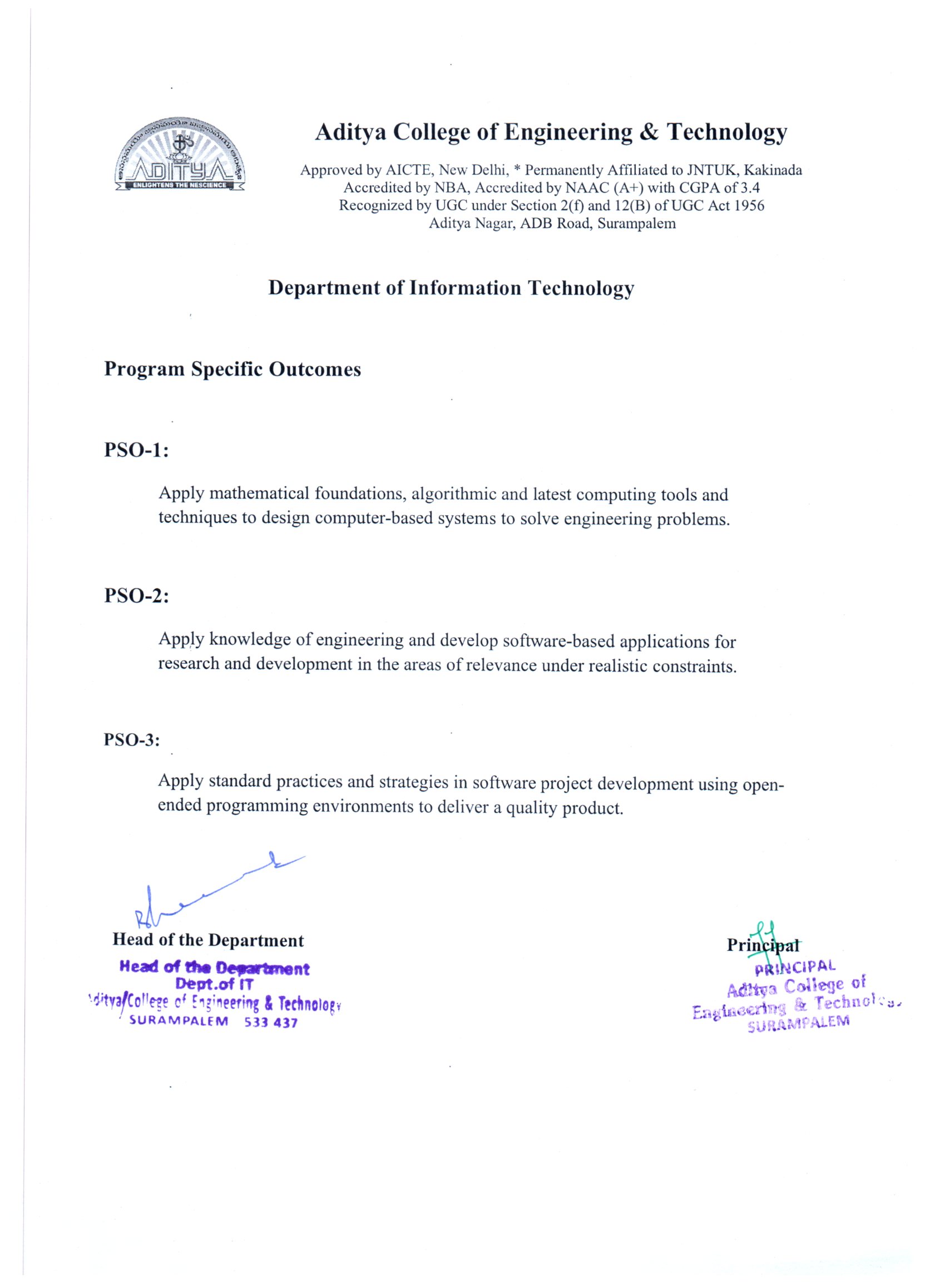












## ABSTRACT

Fake currency notes are increasing day by day, in order to overcome this we proposes a very helpful and efficient system to detect the fake currency. For detecting the fake currency note is done by counting the number of interruptions in the thread line. For predicting the note is real or fake on the basis of number of interruptions. If the number of interruption is zero, if it is real note otherwise it is fake. And also we calculate the entropy of the currency notes for the efficient detection of fake currency note.

Indian fake currency recognition is designed for identification of Indian currency notes and checks whether it is fake or original. The project is very useful in banking system and other field also. In this project we are using CNN(Convolutional neural networks).The project is to provide approaches and strategies,which have proved to be suitable when accessing the image of the desired currency note.

**INDEX**

**CHAPTER**  **PAGE NO**

Abstract i

1.INTRODUCTION

1.1 Artificial Intelligence 1

1.2 Deep Learning 3

2.LITERATURE SURVEY

2.1 Literature survey 6

3.ALGORITHM&DATASET

3.1 Dataset Introduction 11

3.2 Dataset Splitting 11

3.2.1 Training Set 12

3.2.2 Test Set 12

3.2.3 Neural Network 12

3.3 Convolutional Neural Networks 13

3.3.1 Layer Types 14

4.ARCHITECTURE&METHODOLOGY

4.1 System Architecture 24

4.1.1 System Design 24

4.2 UML Diagrams 25

4.2.1 Usecase Diagram 27

4.2.2 Sequence Diagram 27

4.2.3 Activity Diagram 28

4.3 Overview of Development Tools 29

4.3.1 Python 29

4.3.2 Anaconda 29

4.3.3 Anaconda Navigator 30

4.3.4 Jupter Notebook 31

5.PPT PRINTOUTS

6.BASE PAPERS

**LIST OF FIGURES**

**SNO FIGURE NO FIGURE NAME PAGE NO**

1. 1.1 Evolution of Artificial Intelligence 2
2. 1.2 Neural Network Architecture 5
3. 3.1 Convolution Operation 16
4. 3.2 K-Activation Maps 17
5. 3.3 Kernal Types 19
6. 3.4 Pool Types 22
7. 3.5 Fully Connected Layer 22
8. 3.6 Model Architecture 23
9. 4.1 System Architecture 24
10. 4.2 Usecase Diagram 27
11. 4.3 Sequence Diagram 27
12. 4.4 Activity Diagram 28
13. 4.5 Anaconda Prompt 30

# CHAPTER 1

# INTRODUCTION

**1. INTRODUCTION**

Computers and mobile phones have become an unavoidable part of our lives. There are a lot of things which we can do with these technologies. With the rapid development of mobile phones and technologies come several services like application creation - (refers to the process of making application software for handheld and desktop devices such as mobile phones, personal computers and Personal Digital Assistants. Through the usage of apps, the user is provided with various features that will enable him to fulfill all his needs and much more. Apps should be interactive to the users, Camera/webcam services- includes use of camera services for processing various aspects of image. Fake currency Detection is a system that can be used to overcome the limitations most of the people and our institutions of higher learning face with respect to making difference between counterfeit currencies- (is imitation currency produced without the legal sanction of the state or government, usually in a deliberate attempt to imitate that currency and so as to deceive its recipient) and real currencies. The project involves making use of Digital Image Processing Domain - Digital image processing is the use of computer algorithms to perform image processing on digital images.

**1.2 Purpose of the System**

To develop a system using deep learning algorithm that is conventional neural network to predict the whether the indian currency note is real or fake.

* 1. **Scope of the System**

Study existing image detection schemes and concern on recognition base types.

Study the usability features of the existing fake currency detection methods from the general features.

Mapping between the recognition-based image detection system methods and the usability features and extract a collection of usability features to be built in the new system prototype.

**1.4 Existing System**

* In Existing System takes an image acquisition that means taking an input as the image only through the scanner and in this there is no use of any digital camera to capture the image in the real time system.

In this existing system, only front part of the note is take into consideration and not the rear part.

**1.5 Proposed System**

* The proposed system contains the advantages of the existing system and eliminates the disadvantages of it.
* The project centers on the design and implementation of Fake Currency Detection Application.
* The scope of the project is to provide approaches and strategies, which have proved to be suitable when accessing the image of the desired currency note.

# CHAPTER 2

# REQUIREMENTS ANALYSIS

## REQUIREMENT ANALYSIS

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. Requirements analysis is an important aspect of project management. Requirements can be architectural, structural, behavioral, functional and non-functional.

### Functional Requirements

* + - Ease of Use: To promote efficiency and productivity in the development process.
    - High Performance: To support real-time, large scale simulation.
    - Maintainability: To minimize life-cycle cost.
    - Scalability: To support new requirements and ever-increasing scope.
    - Cross-Platform Support: To support multiple operating systems to enable maximum selection of available hardware.

### Non-Functional Requirements

A non-functional requirement is a specification that describes the system’s operation capabilities and constraints that enhance its functionality. We’ve covered different types of [software](https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/) [requirements](https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/), based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system.

* User Interface and Human Factor
* Software Requirements
* Hardware Requirements
* Usability
* Reliability
* Performance
* Physical Environment
* Resource Requirements

#### User Interface and Human Factor

User interface design focuses on the following key areas:

* + - * The design of interfaces between different software components.
      * The design of interfaces between the software and other nonhuman producers and consumers of information.
      * The design of the interface between a human and the computer.

#### Software requirements

The Software requirements of the system are mentioned below:

**Operating System:** Windows 8 or newer

**Software:** IDLE

**Language Used:** Python

**Libraries/Packages:** Pandas,numpy,seaborn,Sklearn..

#### Hardware Requirements

The Hardware requirements of the system are mentioned below:

**Processor:** Intel i3 & above

**Ram:** Minimum 4GB & above

**Hard Disk:** 2GB Onwards

**System Type:** 64 bit

**Operating System:** Windows 10

#### Usability

This system is used for predicting the disease that been trained by the model that takes input as the image of the leaf to be classified.

#### Reliability

This system is used to make accuracy between different models, thus it has high Reliability with value lying between 0 and 1.

#### Performance

As it makes use of different models that help us to make the system high having performance output.

#### Physical Environment

It is compatible with any system like a laptop & system.

#### Resource Requirements

Required software is to be installed like python idle and respected libraries pandas, NumPy, sklearn etc.are to be installed in our system to   execute our project successfully.

# CHAPTER 3

# SYSTEM ANALYSIS

## SYSTEM ANALYSIS

### Introduction

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements on the system. System analysis is a problem-solving activity that requires intensive communication between the system users and system developers.

System analysis is an important phase of any system development process. The system is studied to the minutest detail and analysed. The system analyst plays the role of an interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the inputs to the system are identified. The outputs from the organization are traced through the various processing that the inputs phase through in the organization.

A detailed study of these processes is made by various techniques like Interviews,reviews, etc. The data collected by these sources is scrutinized to arrive to a conclusion.The conclusion is an understanding of how the system functions.

### Use cases

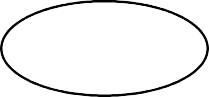
#### Actors

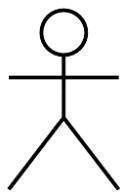
* + - * Developer
      * User

#### Use case Diagrams

Use case diagram is created to visualize the relationships between actors and usecases. A use case is a pattern of behaviour the system exhibits. Each use case is a sequence of related transactions performed by an actor and the system. Diagrammatically actor and use case are represented by stick figure and oval respectively.

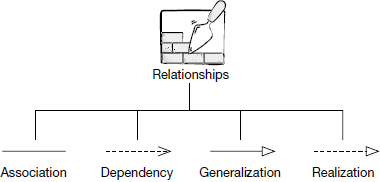
#### Symbols



New Use Case

Actor

#### Relationships



A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

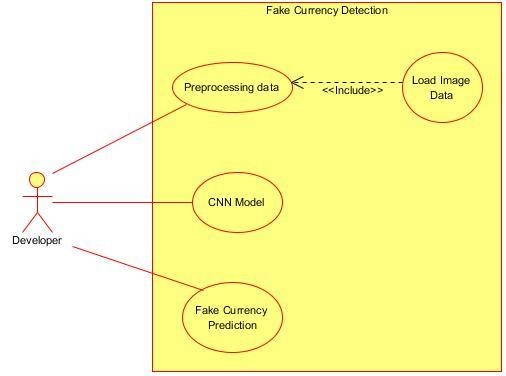
* Use cases: Horizontally shaped ovals that represent the different uses that a user might have.
* Actors: Stick figures that represent the people actually employing the use cases.
* Associations: A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
* System boundary boxes: A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.
* Packages: A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.
* Actors: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
* System: A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
* Goals: The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

### 4.2.1 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

#### List of Use cases

* Load Image Data
* Pre-processing data
* CNN model
* Fake Currency Predict



**Fig:3.2.2.1 Usecase Diagram**

# CHAPTER 4

# SYSTEM DESIGN

## SYSTEM DESIGN

### Introduction

An effective system development life cycle (SDLC) should result in a high qualitysystem that meets customer expectations, reaches completion within time and cost evaluations, and works effectively and efficiently in the current and planned informationtechnology infrastructure.

System Development Life Cycle (SDLC) is a conceptual model which includes policies and procedures for developing or altering systems throughout their life cycles.

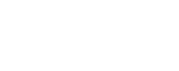
System design is the process of defining the architecture, models, interfaces, anddata for a system to satisfy specified requirements. System design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of system analysis, system architecture and system engineering.

The design phase describes how the system will fulfil the user requirements. To achieve this, we must create both logical and physical design. In this phase the system design functions and operations are described.

## 4.1 System Architecture

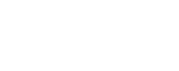
System architecture is a conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structure and behavior of the system.

A representation of a system, including a mapping of functionally onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components.



Input

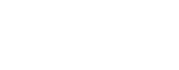
Currency



Gr

ay

scale



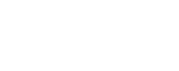
Preprocessssing



Train

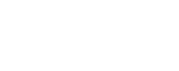


Test



CNN

Model



Output prediction

(

Fake or REAL

)

Fig:4.2.1 System Architecture

### System Object Model

#### Introduction

System object model (SOM) is an object- oriented library packaging technology developed by IBM that allows various programming languages to share class libraries, regardless of the language in which they were originally written. System object model was intended to be used as a solution to many of the interoperability and reuse problems that occur while sharing class libraries between object-oriented and non-object-orientedlanguages.

SOM was designed to be used across IBM’ s mainframe computers and desktops. It serves as an object-oriented model that can be distinguished from other models contained in object-oriented programming languages. SOM basically includes an interface definition a runtime environment with procedure calls and a set of enabling frame

#### Subsystems

Modules

* + Juypter Notebook: Used to create and run the system application.
  + User: To predict the fake and real notes.

### Object Description

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of a system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.

#### Objects

Object is any entity that can be manipulated by the commands of programming languages such as value, variable, function, or data structure.

#### Class Diagrams

A class diagram showing the systems classes, their attributes, operations, and collaborations and the relationships among objects. The class diagrams is the main building lock of object oriented modelling. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application and the classes to be programmed. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system.

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object- oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as −

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.

Class diagrams are the most popular UML diagrams used for construction of software applications. It is very important to learn the drawing procedure of class diagram.Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represents the whole system.

The following points should be remembered while drawing a class diagram −

* The name of the class diagram should be meaningful to describe the aspect of the system.
* Each element and their relationships should be identified in advance.
* Responsibility (attributes and methods) of each class should be clearly identified
* For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
* Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
* Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

#### Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. A sequence diagram shows object interactions arranged in time sequence. The interaction that takes place in collaboration that either realizes a use case or an operation. Sequence diagrams provide high level interactions between user of the system and the system, between the system and the other systems, or between subsystems.

Sequence diagrams can be useful references for businesses and other organizations.

Try drawing a sequence diagram to:

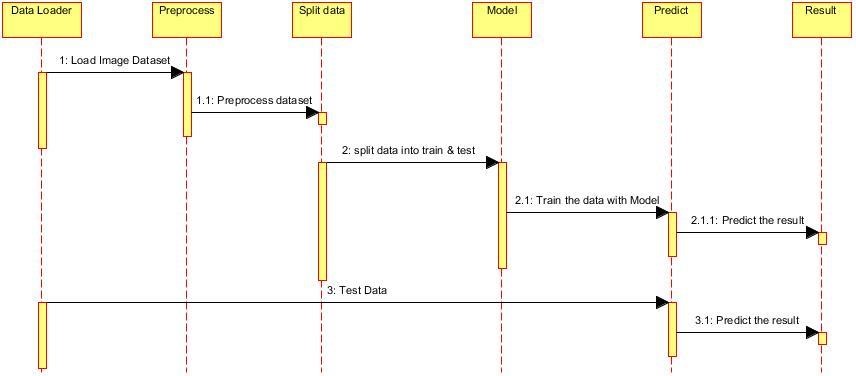
* + - * Represent the details of a UML use case.
      * Model the logic of a sophisticated procedure, function, or operation.
      * See how objects and components interact with each other to complete a process.
      * Plan and understand the detailed functionality of an existing or future scenario.

The following scenarios are ideal for using a sequence diagram:

* + - * Usage scenario: A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system.
      * Method logic: Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to explore the logic of any function, procedure, or complex process.
      * Service logic: If you consider a service to be a high-level method used by different clients, a sequence diagram is an ideal way to map that out.
      * Sequence diagram Visio - Any sequence diagram that you create with Visio can also be uploaded into Lucidchart. Lucidchart supports .vsd and .vdx file import and is a great Microsoft Visio alternative. Almost all of the images you see in the UML section of this site were generated using Lucidchart.

#### 

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. A sequence diagram shows a set of objects and the messages sent and received by those objects. The objects are typically named or anonymous instances of classes, but may also represent instances of other things, such as collaborations, components, and node.



**Fig 4.3** Sequence Diagram

### Dynamic Model

#### State Chart Diagram

An activity diagram is another important diagram in UML to describe the dynamicaspects of the system. An activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of thesystem. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow controlby using different elements such as fork, join, etc. The basic purposes of activity diagrams are similar to the other four diagrams. It captures the dynamic behaviour of thesystem. Other four diagrams are used to show the message flow from one object to another but the activity diagram is used to show message flow from one activity to another.

State Chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State Chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

State Chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State Chart diagram is to model lifetime of an object from creation to termination.

State Chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using State Chart diagrams −

* + - * To model the dynamic aspect of a system.
      * To model the life time of a reactive system.
      * To describe different states of an object during its life time.
      * Define a state machine to model the states of an object.

State Chart diagram defines the states of a component and these state changes are dynamic in nature. Its specific purpose is to define the state changes triggered by events. Events are internal or external factors influencing the system.State Chart diagrams are used to model the states and also the events operating on the system. When implementing a system, it is very important to clarify different states of an object during its life time and State Chart diagrams are used for this purpose. When these states and events are identified, they are used to model it and these models are used during the implementation of the system.

If we look into the practical implementation of State Chart diagram, then it is mainly used to analyze the object states influenced by events. This analysis is helpful to understand the system behavior during its execution.

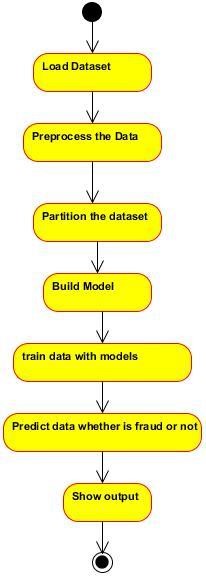
The main usage can be described as −

* + - * To model the object states of a system.
      * To model the reactive system. Reactive system consists of reactive objects.
      * To identify the events responsible for state changes.
      * Forward and reverse engineering

State Chart diagram is used to describe the states of different objects in its life cycle. Emphasis is placed on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately. State Chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

Before drawing a State Chart diagram, we should clarify the following points −

* + - * Identify the important objects to be analyzed.
      * Identify the states.
      * Identify the events.



**Fig 4.4** State Chart Diagram

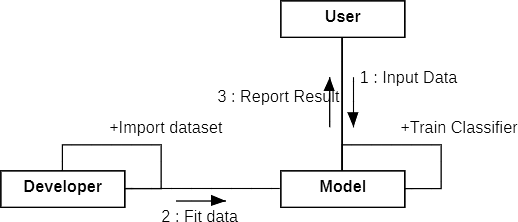
#### Communication Diagram

The communication diagram and the sequence diagram are similar. They're semantically equivalent, that is, they present the same information, and you can turn a communication to a sequence diagram and vice versa. The main distinction between them is that the communication diagram arranges elements according to space, the sequence diagram is according to time.

Communication diagrams offer benefits similar to sequence diagrams, but they will offer a better understanding of how components communicate and interact with each other rather than solely emphasizing the sequence of events. They can be a useful reference for businesses, organizations, and engineers who need to visualize and understand the physical communications within a program. Try drawing a sequence diagram to:

* Model the logic of a sophisticated procedure, function, or operation.
* Identify how commands are sent and received between objects or components of a process.
* Visualize the consequences of specific interactions between various components in a process.
* Plan and understand the detailed functionality of an existing or future scenario.

A communication diagram offers the same information as a sequence diagram, but while a sequence diagram emphasizes the time and order of events, a communication diagram emphasizes the messages exchanged between objects in an application. Sequence diagrams can fall short of offering the "big picture”.This is where communication diagrams come in and offer that broader perspective within a process.



**Fig 4.5.3.1 Communication diagram**

### Static Model

#### Component Diagram

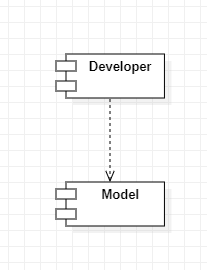
UML Component diagrams are used in modelling the physical aspects of object oriented systems that are used for visualizing, specifying, and documenting component- based systems and also for constructing executable systems through forward and reverse engineering.Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as −

* Visualize the components of a system.
* Construct executables by using forward and reverse engineering.
* Describe the organization and relationships of the components.



**Fig. 4.6.1.1 Component Diagram**

#### Deployment Diagram

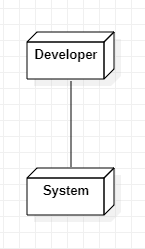
Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Deployment diagrams are important for visualizing, specifying, and documenting embedded, client/server, and distributed systems and also for managing executable systems through forward and reverse engineering.

A deployment diagram is just a special kind of class diagram, which focuses on a system's nodes. Graphically, a deployment diagram is a collection of vertices and arcs. Deployment diagrams commonly contain:

A **UML** deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

Deployment diagrams are important for visualizing, specifying, and documenting embedded, client/server, and distributed systems and also for managing executable systems through forward and reverse engineering.A deployment diagram is just a special kind of class diagram, which focuses on a system's nodes. Graphically, a deployment diagram is a collection of vertices and arcs.



**Fig. 4.6.2.1 Deployment Diagram**

# CHAPTER 5

# IMPLEMENTATION

## IMPLEMENTATION

### Software Used

### 4.3.4 Jupyter Notebook

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents—which are a type of computational notebook. The "notebook" term can colloquially refer to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on the context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells that can contain code, text (using Markdown), mathematics, plots, and rich media, usually ending with the ".ipynb" extension.

Jupyter Notebook can connect to many kernels to allow programming in many languages. By defaultJupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia, and Haskell.

The Notebook interface was added to IPython in the 0.12 release(December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s.[13] According to The Atlantic, Jupyter interest overtook the popularity of the Mathematical notebook interface in early 2018.

**Features of Jupyter**

Jupyter provides several features that make it a popular tool for data science and scientific computing:

* Interactive Environment: Jupyter notebooks provide an interactive environment where you can write and execute code, view the results of your computations, and explore data in real-time.
* Multiple Programming Languages: Jupyter supports multiple programming languages, including Python, R, Julia, and many others.
* Live Code Execution: Jupyter notebooks enable live code execution, which means that you can see the results of your code as you type it. This feature makes it easy to debug and experiment with your code.
* Rich Media Support: Jupyter notebooks support rich media formats such as images, videos, and audio files, enabling you to create interactive and engaging presentations.
* Collaboration: Jupyter notebooks support collaboration, allowing multiple users to work on the same notebook simultaneously. This feature makes it easy to share and collaborate on research and data analysis projects.
* Data Visualization: Jupyter notebooks provide powerful data visualization capabilities through popular Python libraries such as Matplotlib, Seaborn, and Plotly.
* Documentation: Jupyter notebooks provide a convenient way to document your code and data analysis processes, making it easier to reproduce and understand your results.

Overall, Jupyter notebooks provide an intuitive and flexible environment for data science and scientific computing that enables researchers and developers to explore, analyze, and communicate their data in a collaborative and interactive way.

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

There are two major Python versions: Python2 and Python3

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

The biggest strength of Python is huge collection of standard library which can be

used for the following :

* Machine Learning
* GUI Applications (like Kivy, Tkinter, PyQt etc. )
* Web frameworks like Django (used by YouTube, Instagram, Dropbox)
* Image processing (like OpenCV, Pillow)
* Web scraping (like Scrapy, BeautifulSoup, Selenium)
* Test frameworks
* Multimedia

There are more advantages of using Python Programming Language. Some of them are:

* **Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit- testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don‘t have to write the complete code for that manually.

* **Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

* **Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

* **Improved Productivity**

The language‘s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

* **IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

* **Simple And Easy**

When working with Java, you may have to create a class to print „Hello World‟. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

* **Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

* **Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

* **Free And Open Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

* **Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

* **Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn‘t the same with Python. Here, you need to code.

only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

**Advantages of Python Over Other Languages**

* **Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don‘t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

* **Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

* **Python is for everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

**Disadvantages of Python**

So far, we‘ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let‘s now see the downsides of choosing Python over another language.

* **Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn‘t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

* **Weak in mobile computing and browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

* **Design Restrictions**

As you know, Python is dynamically-typed. This means that you don‘t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what‘s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run- time errors.

* **Underdeveloped database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python‘s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

This was all about the Advantages and Disadvantages of Python Programming Language.

**History of Python**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it." Later on in the same Interview, Guido van Rossumcontinued: "I remembered all my experience and some of my frustration withABC**.** Idecided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

#### 1.3 Deep Learning

Deep learning is an [artificial intelligence (AI)](https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of [machine learning](https://www.investopedia.com/terms/m/machine-learning.asp) in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network. Deep learning, also known as deep neural networks or neural learning, is a form of artificial intelligence (AI) that seeks to replicate the workings of a human brain. It is a form of machine learning, with functions that operate in a nonlinear decision-making process. Deep learning occurs when decisions are made on unstructured data without supervision. Object recognition, speech recognition, and language translation are some of the tasks performed through deep learning.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as [big data,](https://www.investopedia.com/terms/b/big-data.asp) is drawn from sources like social media, internet search engines, [e-commerce](https://www.investopedia.com/terms/e/ecommerce.asp) platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through [fintech](https://www.investopedia.com/tech/worlds-top-10-fintech-companies-baba/) applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

* Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.
* Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.
* Deep learning, a form of machine learning, can be used to help detect fraud or money laundering, among other functions.
* Using the fraud detection system mentioned above with machine learning, one can create a deep learning example. If the machine learning system created a model with parameters built around the number of dollars a user sends or receives, the deep-learning method can start building on the results offered by machine learning.

Each layer of its neural network builds on its previous layer with added data like a retailer, sender, user, social media event, credit score, IP address, and a host of other features that may take years to connect together if processed by a human being. Deep learning algorithms are trained to not just create patterns from all transactions, but also know when a pattern is signaling the need for a fraudulent investigation. The final layer relays a signal to an analyst who may freeze the user’s account until all pending investigations are finalized.

Deep learning is used across all industries for a number of different tasks. Commercial apps that use image recognition, [open-source](https://www.investopedia.com/terms/o/open-source.asp) platforms with consumer recommendation apps, and medical research tools that explore the possibility of reusing drugs for new ailments are a few of the examples of deep learning incorporation.

**1.3.2 Typical Neural Network Architecture**

The typical neural network architecture consists of several layers. We call the first layer as the input layer.

The input layer receives the input x, data from which the neural network learns. In our previous example of classification of handwritten numbers, these input x would represent the images of these numbers ( x is basically an entire vector where each entry is a pixel). The input layer has the same number of neurons as there are entries in the vector x. Meaning: each input neuron represents one element in the vector x.

The last layer is called the output layer, which outputs a vector y representing the result that the neural network came up with. The entries in this vector represent the values of the neurons in the output layer. In our case of classification, each neuron in the last layer would represent a different class. In this case, the value of an output neuron gives the probability that the handwritten digit given by the features x belongs to one of the possible classes (one of the digits 0–9). As you can imagine the number of output neurons must be the same as there are classes.

In order to obtain a prediction vector y, the network must perform certain mathematical operations. These operations are performed in the layers between the input and output layers. We call these layers the hiddenlayers.Now lets us discuss how the connections between the layers looklike.

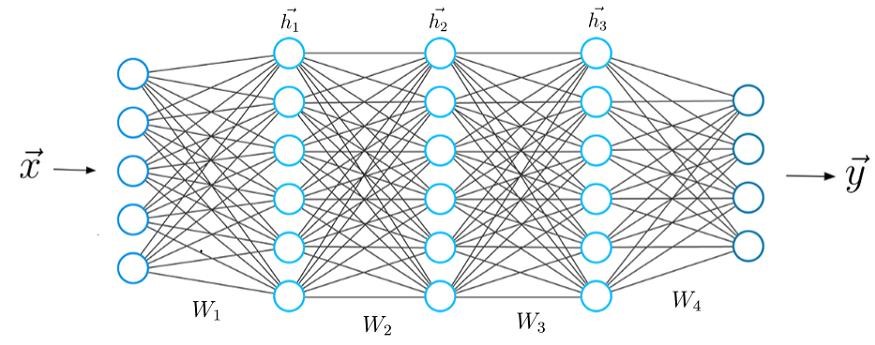


Fig 1.2 Neural Network Architecture

## 3.3 Convolutional Neural networks

In traditional feed-forward neural networks, each neuron in the input layer is connected to every output neuron in the next layer – we call this a fully connected (FC) layer. However, on CNN, we don’t use FC layers until the very last layers in the network. We can thus define a CNN as a neural network that swaps in a specialized “convolutional” layer in place of a “fully-connected” layer for at least one of the layers in the network.

A nonlinear activation function, such as RELU, is then applied to the output of these convolutions and the process of convolution => activation continues along with a mixture of other layer types to help reduce the width and height of the input volume and help reduce the width and height of the input volume and help reduce overfitting until we finally reach the end of the network and apply one or two FC layers where we can obtain our final output classifications.

Each layer in a CNN applies a different set of filters, typically hundreds or thousands of them, and combines the results, feeding the output into the next layer in the network. During training, a CNN automatically learns the values for these filters. In the context of image classification, our CNN may learn to:

* Detect edges from raw pixel data in the first layer.
* Use these edges to detect shapes in the second layer.
* Use these shapes to detect higher-level features in the highest layers of the network.
* The last layer on CNN uses these higher-level features to make predictions regarding the contents of the image. In practice, CNNs give us two key benefits: local invariance and compositionality. The concept of local invariance allows us to classify an image as containing a particular object regardless of where in the image the object appears. We obtain this local invariance through the usage of “pooling layers” which identifies regions of our input volume with a high response to a particular filter.

The second benefit is compositionality. Each filter composes a local patch of lower-level features into a higher-level representation, similar to how we can compose a set of mathematical functions that build on the output of previous functions. This composition allows our network to learn more rich features deeper in the network. For example, our network may build edges from pixels, shapes from edges, and then complex objects from shapes – all in an automated fashion that happens naturally during the training process. The concept of building higher-level features from lower-level ones is exactly why CNN’s are so powerful in computer vision. To understand the contents of the image, we must apply image classification, which is the task of using computer vision and machine learning algorithms to extract meaning from an image. This action could be as simple as assigning a label to what the image contains, or as advanced as interpreting the contents of an image and returning a human-readable sentence.

### 3.3.1 Layer types

There are many types of layers used to build convolutional neural networks, but the ones we are most likely to encounter include:

* Convolutional Layer
* Pooling Layer
* Fully-connected Layer

As considering the requirement of the project we are using Convolutional (CONV), Pooling (POOL), Fully Connected (FC) as our layers in the model.

Stacking a series of these layers in a specific manner yields a CNN. We often use simple text diagrams to describe a CNN: INPUT => CONV => RELU=> POOL => FC=> SOFTMAX.

Here we define a simple CNN that accepts input, applies a convolution layer, then an activation layer, then a fully-connected layer, and finally, a SOFTMAX classifier to obtain the output classification

probabilities. The SOFTMAX activation layer is often omitted from the network diagram as it is assumed it directly follows the final Fully-Connected layer.

Of these layer types, CONV and FC are the only layers that obtain parameters that are learned during the training process. Activation and dropout layers are not considered true “layers” themselves but are often included in network diagrams to make the architecture explicitly clear. Pooling layers(POOL), of equal importance as CONV and FC, are also included in network diagrams as they have a substantial impact on the spatial dimensions of an image as it moves through a CNN. CONV, POOL, and FC are the most important when defining your actualnetwork architecture.

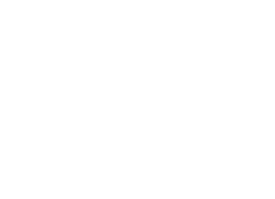
That’s not to say that the other layers are not critical, but take a backseat to this critical set of four as they define the actual architecture itself.

## • Convolutional Layer

The CONV layer is the core building block of a Convolutional Neural Network. The CONV layer parameters consist of a set of K learnable filters (i.e., “kernels”), where each filter has a width and a height, and are nearly always square. These filters are small (in terms of their spatial dimensions) but extend throughout the full depth of the volume.

For inputs to the CNN, the depth is the number of channels in the image (i.e., a depth of three when working with RGB images, one for each channel). For volumes in the network, the depth will be the number of filters applied in the previous layer.

To make this concept clearer, let’s consider the forward-pass of a CNN, where we convolve each of the K filters across the width and height of the input volume. More simply, we can think of each of our K kernels sliding across the input region, computing an element-wise multiplication, summing, and then storing the output value in a 2- dimensional activation map, such as



Image



K

**Fig 3.1** convolution operation

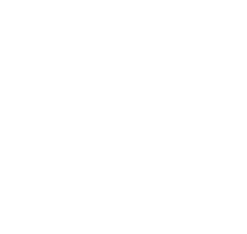
**Step 1:** K kernels waiting to be applied to the image.

**Step 2**: Each kernel is convolved with the input volume.

**Step 3:** The output of each Convolution operation produces a 2d output.

After applying all k filters to the input volume, we now have K, 2-dimensional activation maps. We then stack our K activation maps along the depth dimension of our array to form the final output volume.

**++Fig 3.2:** After obtaining the K activation maps, they are stacked together to form the input volume to the next layer in the network.



where features specific to the kernel have been detected in the input. Every entry in the output volume is thus an output of a neuron that looks at only a small region of the input. In this manner, the network learns filters that activate when they see a specific type of a feature at a given spatial location in the input volume. In lower layers of the network, filters may activate when they see edge-like or corner-like regions.

Then, in the deeper layers of the network, filters may activate in the presence of high-level features, such as parts of the face, the hood of a car, etc. This activation concept goes back to our neural network analogy these neurons are becoming excited and activating when they see a particular pattern in an

input image.

The concept of convolving small filters with a large(r) input volume has special meaning in convolutional neural networks-specifically, the local connectivity, and the receptive field of a neuron.

When working with images, it’s often impractical to connect neurons in the current volume to all neurons in the previous volume – there are simply too many connections and too many weights, making it impossible to train deep networks on images with large spatial dimensions. Instead, when utilizing CNNs, we choose to connect each neuron to only a local region of the input volume –we call the size of this local region the receptive field of the neuron.

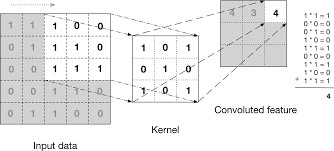
To make the point clear, consider a dataset where the input volume as an input size of 32x32x3. Each image that has a width of 32 pixels, a height of 32 pixels, and depth of 3.if our reception field is of size 3x3, then each neuron in the CONV layer will connect to a 3x3 local region of the image for a total of 3x3x3=27 weights (remember, the depth of the filters is three because they extend through the full depth of the input image, in this case, three channels).

Now, let’s assume that the spatial dimensions of our input volume have been reduced to a smaller size, but our depth is now larger due to utilizing more filters deeper in the network, such that volume size is now 16x16x94. Again, if we assume a receptive field of size 3x3, then every neuron in the CONV layer will have a total of 3x3x94=846 connections to the input volume, simply put, the receptive field F is the size of the filter, yielding an FXF kernel that is convolved with the input volume.

At this point we have explained the connectivity of neurons in the input volume, but not the arrangement or size of the output volume. Three parameters control the size of an output volume: the kernel, stride, and zero-padding size, each of which we’ll review below.

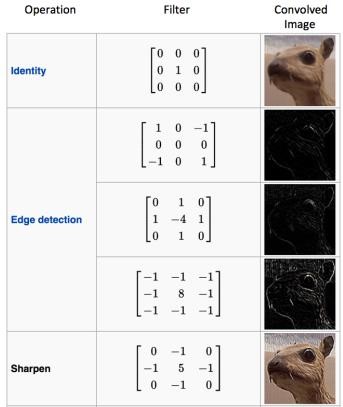
### KERNEL

A filter or kernel is an integral component of the layered architecture. It is a smaller-sized matrix in comparison to the dimensions of the image, that contains real-valued entries. The kernels are then convolved with the input volume to obtain so-called activation maps. Activation maps indicate activated regions, i.e. regions



**Fig 3.2** Kernel operation

Kernel for the different things like image identity, edge detection, sharpening the images with



**Fig 3.3** Kernel types

### STRIDE

We described a convolution operation as “sliding” a small matrix across a large matrix, stopping at each coordinate, computing an element-wise multiplication and sum, then storing the output. This description is similar to a sliding window that slides from left-to-right and top-to-bottom across an image.

### ZERO-PADDING

It needs to “pad” the borders of an image to retain the original image size when applying a convolution – the same is true for filters inside of a CNN. Using zero-padding, we can “pad” our input along the borders such that our output volume size matches our input volume size. The amount of padding .- apply is controlled by the parameter “padding”.

This technique is especially critical when we start looking at deep CNN architectures that apply multiple CONV filters on top of each other.

Without zero padding, the spatial dimensions of the input volume would decrease too quickly, and

we wouldn't be able to train deep networks (as the input volumes would be too tiny to learn any useful patterns from).

Putting all these parameters together, we can compute the size of an output volume as a function of the input volume size (W, assuming the input images are square, which they nearly always are), the receptive field size F, the stride S, and the amount of zero-padding P, To construct a valid CONV layer, we need to ensure the following equation is an integer:

## ((W-F+2P)-S)+1

If it is not an integer, then the strides are set incorrectly, and the neurons cannot be tiled such that they symmetrically fit across the input volume.

### • Pooling Layers

There are two methods to reduce the size of an input volume-CONV lavers with a stride > 1 (which we've already seen) and POOL layers. It is common to insert POOL layers in-between consecutive CONV layers in CNN architectures:

#### INPUT -> CONV => RELU=> POOL=> CONV => RELU => POOL>FC

The primary function of the POOL layer is to progressively reduce the spatial size (ie., width and height) of the input volume. Doing this allows us to reduce the number of parameters and computation in the network- pooling also helps us control overfitting.

POOL Layers operate on each of the depth slices of an input independently using either the max or average function. Max pooling is typically done in the middle of the CNN architecture to reduce the spatial size, whereas average pooling is normally used as the final layer of the network where we wish to avoid using FC layers entirely. The most common type of POOL layer is max pooling, although this trend is changing with the introduction of more exotic micro-architectures.

Typically we'll use a pool size of 2x2, although deeper CNNs that use larger input images (>200 pixels) may use a 3x3 pool size early in the network architecture. We also commonly set the stride to either SI or S- 2. Figure 11,10 (heavily inspired by Karpathy et al. 121) follows an example of applying max pooling with a 2x2 pool size and a stride of S = 1. Notice for every 2x2 block, we keep only the largest value, take a single step (like a sliding window), and apply the operation again thus producing an output volume size of 3x3.

We can further decrease the size of our output volume by increasing the stride -here we apply S= 2 to the same input. For every 2x2 block in the input, we keep only the largest value. Then take a step of two pixels, and apply the operation again. This pooling allows us to reduce the width and height by a factor of two, effectively discarding 75% of activations from the previous layer.

In summary, POOL layers Accept an input volume of size Winput x H input x D input. They then require two parameters:

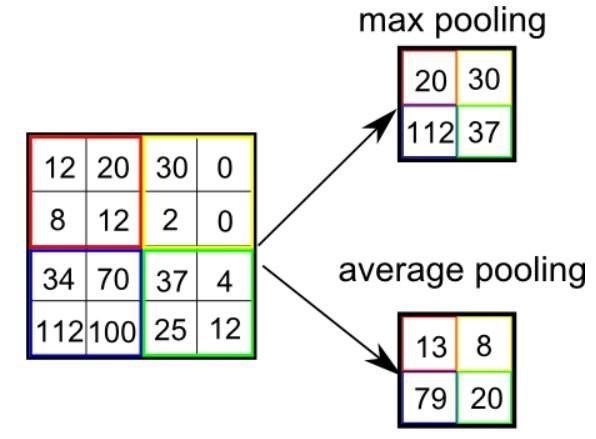
The receptive field size F (also called the "pool size"), Stride S.

Applying the POOL operation yields an output volume of size Woutput x H output x D output, where:

Woutput =((Winput-F)=S)+ 1

Houtput=((hinput-F)=S)+1

Doutput=Dinput



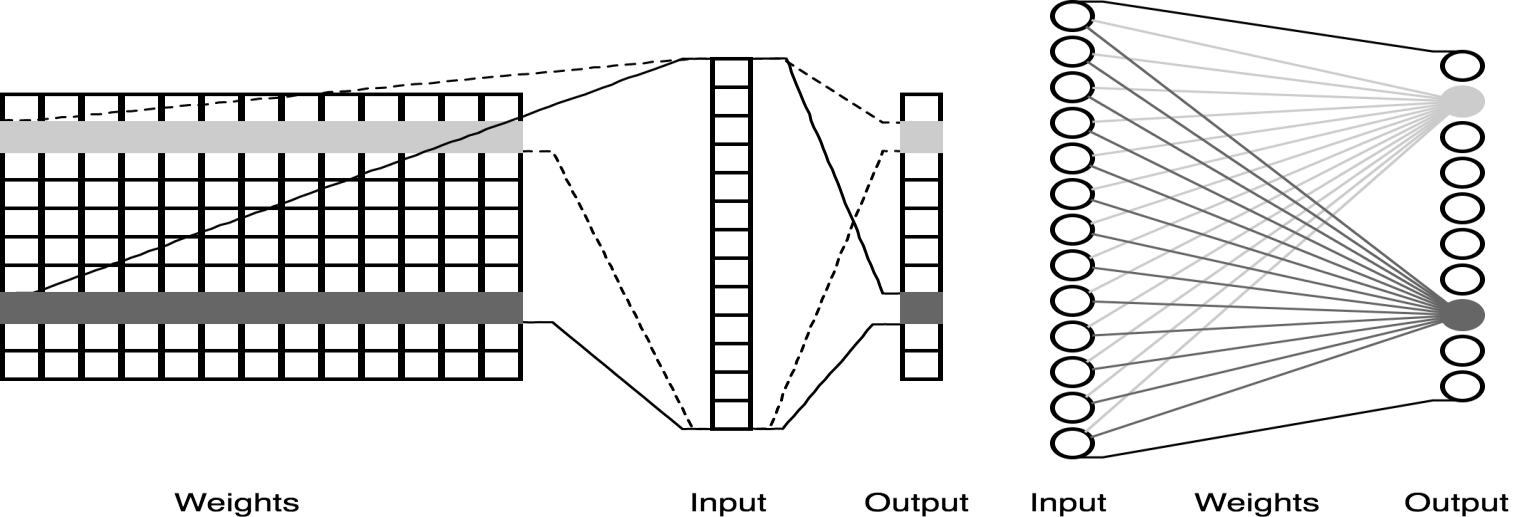
**Fig 3.4** Pool types

### • Fully-connected Layers

Neurons in FC layers are fully-connected to all activations in the previous layer, as is the standard for feedforward neural networks. FC layers are always placed at the end of the network (i.e, we don't apply a CONV layer, then an FC layer, I followed by another CONV) layer.

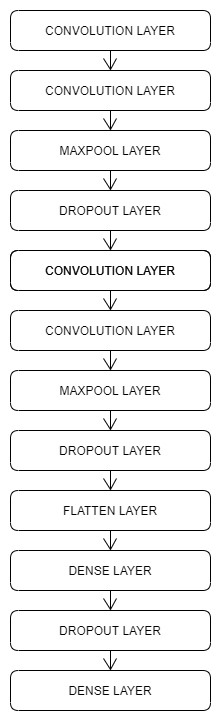
It's common to use one or two FC layers prior to applying the softmax classifier, as the following (simplified) architecture demonstrates: INPUT-> CONV-> RELU => POOL=> CONV-> RELU=> POOL-> FC->FC.

Here we apply two fully-connected layers before our (implied) softmax classifier which will compute our final output probabilities for eachclass.



**Fig 3.5** Fully connected layer

**CNN MODEL ARCHITECTURE:**



**Fig 3.6** Model architecture

**SourceCode:**

**from** tensorflow.keras.applications.vgg16 **import** VGG16,preprocess\_input

**from** keras.preprocessing.image **import** ImageDataGenerator

**from** tensorflow.keras.layers **import** Dense,Activation,Flatten,Dropout

**from** tensorflow.keras.models **import** Sequential,Model,load\_model

**from** tensorflow.keras **import** optimizers

**from** tensorflow.keras.callbacks **import** ModelCheckpoint,EarlyStopping

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

In [ ]:

*#define height and width of the image*

height**=**300

width**=**300

*#create a ResNet50 model instance without the top layer as we will add our own top layer*

base\_model**=**VGG16(weights**=**'imagenet',include\_top**=False**,input\_shape**=**(height,width,3))

In [ ]:

*#define directory containing training and validation data*

train\_dir**=**"Dataset/Training"

validation\_dir**=**"Dataset/Validation"

*#number of batches the data has to be divided into*

batch\_size**=**8

*#create datagen and generator to load the data from training directory*

train\_datagen**=**ImageDataGenerator(preprocessing\_function**=**preprocess\_input,rotation\_range**=**90,horizontal\_flip**=True**,vertical\_flip**=True**)

train\_generator**=**train\_datagen**.**flow\_from\_directory(train\_dir,target\_size**=**(height,width),batch\_size**=**batch\_size)

*#create datagen and generator to load the data from validation directory*

validation\_datagen**=**ImageDataGenerator(preprocessing\_function**=**preprocess\_input,rotation\_range**=**90,horizontal\_flip**=True**,vertical\_flip**=True**)

validation\_generator**=**validation\_datagen**.**flow\_from\_directory(validation\_dir,target\_size**=**(height,width),batch\_size**=**batch\_size)

*#our own model which will be added onto the ResNet50 model*

**def** build\_finetune\_model(base\_model,dropout,fc\_layers,num\_classes):

**for** layer **in** base\_model**.**layers:

layer**.**trainable**=False**

x**=**base\_model**.**output

x**=**Flatten()(x)

**for** fc **in** fc\_layers:

x**=**Dense(fc,activation**=**'relu')(x)

x**=**Dropout(dropout)(x)

predictions**=**Dense(num\_classes,activation**=**'softmax')(x)

finetune\_model**=**Model(inputs**=**base\_model**.**input,outputs**=**predictions)

**return** finetune\_model

class\_list**=**['Real','Fake'] *#the labels of our data*

FC\_Layers**=**[1024,1024]

dropout**=**0.5

finetune\_model**=**build\_finetune\_model(base\_model,dropout**=**dropout,fc\_layers**=**FC\_Layers,num\_classes**=**len(class\_list))

*#define number of epochs(the number of times the model will be trained) and number of training images*

num\_epochs**=**30

num\_train\_images**=**35

*#checkpoint in case anything goes wrong*

checkpoint**=**ModelCheckpoint("Final\_model.h5",monitor**=**'val\_acc',verbose**=**1,save\_best\_only**=True**,save\_weights\_only**=False**,mode**=**'auto',period**=**1)

early**=**EarlyStopping(monitor**=**'val\_acc',min\_delta**=**0,patience**=**40,verbose**=**1,mode**=**"auto")

*#compile the model before using*

finetune\_model**.**compile(loss**=**"categorical\_crossentropy",optimizer**=**optimizers**.**SGD(lr**=**0.000001,momentum**=**0.9),metrics**=**['accuracy', ''])

*#train the model*

history **=** finetune\_model**.**fit\_generator(generator**=**train\_generator,steps\_per\_epoch**=**num\_train\_images**//**batch\_size,epochs**=**num\_epochs,validation\_data**=**validation\_generator,validation\_steps**=**1,callbacks**=**[checkpoint,early])

*#save the model*

finetune\_model**.**save\_weights("Final\_model.h5")

*#testing the model*

img**=**image**.**load\_img("Dataset/Testing/Fake.jpeg",target\_size**=**(300,300)) *#The path of the testing image,the pic taken from the phone should come here*

img**=**np**.**asarray(img)

plt**.**imshow(img)

img**=**np**.**expand\_dims(img,axis**=**0)

finetune\_model**.**load\_weights("Final\_model.h5")

output**=**finetune\_model**.**predict(img) *#predicting the image using model created*

**if**(output[0][0]**>**output[0][1]): *#comparison*

print("Real")

**else**:

print("Fake")

*#testing the model*

img**=**image**.**load\_img("Dataset/Testing/Real.jpg",target\_size**=**(300,300)) *#The path of the testing image,the pic taken from the phone should come here*

img**=**np**.**asarray(img)

plt**.**imshow(img)

img**=**np**.**expand\_dims(img,axis**=**0)

finetune\_model**.**load\_weights("Final\_model.h5")

output**=**finetune\_model**.**predict(img) *#predicting the image using model created*

**if**(output[0][0]**>**output[0][1]): *#comparison*

print("Real")

**else**:

print("Fake")

**CHAPTER 6**

**TESTING**

**6.TESTING**

**6.1 Introduction**

A test case is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. Insome setting, an oracle could be a requirement or use case. It may take a test case to determine that a software program or system is functioning correctly. Test cases are often referred to as test scripts, particularly when written. Written test cases are usuallycollected into test suites.

**6.1.1 Levels of Testing**

In order to uncover the errors present in different phases, we have the concept of levelsof testing. The basic levels of testing are

Client needs Acceptance Testing

System Testing

Design Integration

Testing

Code Unit Testing

**6.1.2 Software Testing Strategies**

* Unit Testing
* Integration Testing
* Validation Testing
* System Testing

**i. Unit Testing**

Unit Testing is a level of software testing where individual units/ componentsof the software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usuallyhas one or a few inputs and usually a single output. In procedural programming, a unitmay be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

**ii. Integration Testing**

Integration Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in integration Testing. Integration Testing is the second level of testing performed after Unit Testing and before System Testing.

**iii. Validation testing**

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed in an appropriate environment.

**iv. System Testing**

System testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements. System testing is performed on the entire system in the context of either functional requirement specifications (FRS) or system requirement specification (SRS), or both. System testing tests not only the design but also the behaviour and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software or hardware requirements specification.

**CHAPTER 8**

**CONCLUSION**

**8.1 CONCLUSION**

We commenced with a brief introduction to our system and discussed the scope and objectives of our project. During the literature survey we got an opportunity to look closely into the problem that people are facing in the current environment, we reviewed multiple research papers we selected five papers as our base research papers. We analyzed all existing architectures of our base papers and by understanding their working we have discovered some flaws in the currently existing system. We have kept all the prime features of existing systems as a primary focus with some of the additional features for our proposed system.

# CHAPTER 9

# FUTURE SCOPE

## FUTURE SCOPE

In Future To Extension of these we can use more no of variables for comparing and the calculating the Accuracy Between the models. And we can also use different models for performing the parametric techniques. Where different models and usage of different variables can bring us to understand more about the outcome.

And there may be using different dataset as input for different notes , we have to update according to the Deep Learning upcoming parametric models. And we come across several information known to us. As in future we only depends upon the models differently used by the developer.

# CHAPTER 10

# BIBLIOGRAPHY

## 10. BIBLIOGRAPHY

1. Prof Chetan More, Monu Kumar, Rupesh Chandra, Raushan Singh, “Fake currency Detection using Basic Python Programming and Web Framework” IRJET International Research Journal of Engineering and Technology, Volume: 07 Issue: 04 | Apr 2020 ISSN: 2395- 0056
2. Vivek Sharan, Amandeep Kaur,” Detection of Counterfeit Indian Currency Note Using Image

Processing” International Journal of Engineering and Advanced Technology (IJEAT), Volume.09, Issue:01, ISSN: 2249-8958 (October 2019)

1. Aakash S Patel, “Indian Paper currency detection” International Journal for Scientific Research & Development (IJSRD), Vol. 7, Issue 06, ISSN: 2321-0613 (June 2019)
2. Archana M Kalpitha C P, Prajwal S K, Pratiksha N,” Identification of fake notes and denomination recognition” International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume. 6, Issue V, ISSN: 2321-9653, (May 2018)
3. S. Atchaya, K. Harini, G. Kaviarasi, B. Swathi, “Fake currency detection using Image processing”, International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333 (2017)
4. Prof Chetan More, Monu Kumar, Rupesh Chandra, Raushan Singh, “Fake currency Detection using Basic Python Programming and Web Framework” IRJET International Research Journal of Engineering and Technology, Volume: 07 Issue: 04 | Apr 2020 ISSN: 2395- 0056
5. Vivek Sharan, Amandeep Kaur,” Detection of Counterfeit Indian Currency Note Using Image

Processing” International Journal of Engineering and Advanced Technology (IJEAT), Volume.09, Issue:01, ISSN: 2249-8958 (October 2019)

1. Aakash S Patel, “Indian Paper currency detection” International Journal for Scientific Research & Development (IJSRD), Vol. 7, Issue 06, ISSN: 2321-0613 (June 2019)
2. Archana M Kalpitha C P, Prajwal S K, Pratiksha N,” Identification of fake notes and denomination recognition” International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume. 6, Issue V, ISSN: 2321-9653, (May 2018)
3. S. Atchaya, K. Harini, G. Kaviarasi, B. Swathi, “Fake currency detection using Image processing”, International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-92.