1. INTRODUCTION

1.1 Overview:

Financial transactions are integral to daily life, with banknotes serving as a fundamental asset in any economy. However, the prevalence of counterfeit banknotes poses a significant threat by creating discrepancies in the financial market. These counterfeit notes are increasingly sophisticated, closely resembling genuine currency, which complicates detection efforts While counterfeiting was not a major concern in the 1990s, it has become a critical issue with the advancement of technology. The late 19th and 20th centuries saw a drastic increase in forgery as counterfeiters developed more advanced techniques

Modern technology has enabled counterfeiters to produce fake notes with features that closely mimic genuine currency, making it challenging to differentiate between the two This technological sophistication threatens the financial market's stability and underscores the necessity for effective counterfeit detection mechanisms Identifying counterfeit notes manually is difficult, despite the security features incorporated into genuine banknotes by governments. Fraudsters have become adept at replicating these features with high precision, making manual detection increasingly unreliable.

To mitigate these issues, it is essential for banks and ATM systems to incorporate automated systems capable of distinguishing between genuine and counterfeit notes In recent years, Artificial Intelligence (AI) and Machine Learning (ML) have emerged as promising tools for developing such systems. Supervised Machine Learning (SML) approaches, which have demonstrated significant success in other fields such as medical diagnostics are increasingly being applied to currency authentication

The process of detecting counterfeit notes typically involves capturing an image of the note, applying various image processing techniques to extract its features, and then using SML algorithms to classify the note as genuine or fake. Despite the potential of these

technologies, there has been limited research specifically focusing on applying SML for banknote authentication. This gap highlights the need for further exploration and development in this area to enhance the effectiveness of automated counterfeit detection systems.

In conclusion, integrating AI and ML into currency verification processes represents a promising advancement in improving counterfeit detection. By leveraging these technologies, financial institutions can better safeguard the integrity of transactions and protect against the negative impacts of counterfeit currency.

1.2 Company Profile:

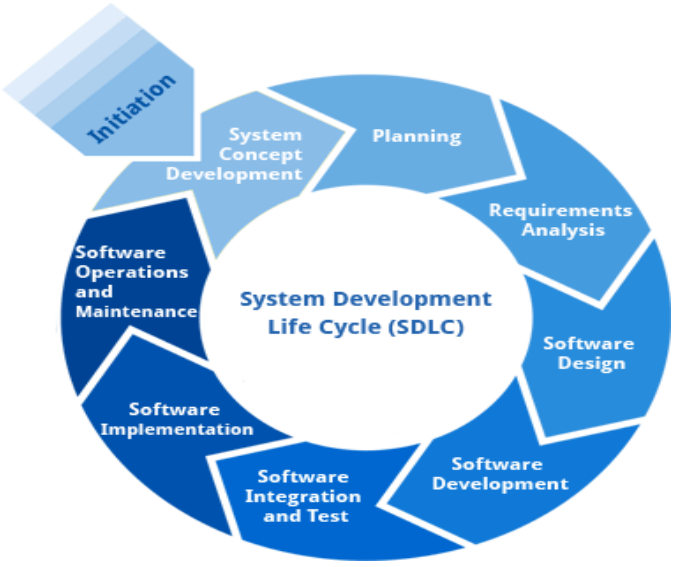
MANAC InfoTech Pvt. Ltd. (MIPL) is a progressive and innovative IT company offering a spectrum of services including software development, staffing, recruitment and training. An IIM Alumnus Company started in 1998, erstwhile training partner of TCS ION Tata Consultancy Services Limited (TCS Ltd.) is a leading IT solutions company & one of the world's leading Information Technology consulting, services and business process outsourcing organizations and a part of the Tata Group, India's best-known business conglomerate. MANAC has built a suite of services around its core offering of training.

In the training arena MANAC has an unenviable track record of grooming several hundreds of students, building their skills and launching their careers. Alumni of MANAC are in senior IT positions across the globe. MANAC also offers a host of training programs tailor made for IT companies. The development arm of MIPL focuses on providing solutions to the Education & Training sector including assessments, analytics, online course delivery and ERP for training companies. Along the way, MANAC has developed a strong expertise in JAVA, Android and open source technologies. Relations built with HR managers over the years have helped MANAC provide specialized services in the area of staffing and fresher's recruitment for IT & ITES companies.

2. LITERATURE OVERVIEW

2.1 The Systems Development Life Cycle (SDLC):

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies use to develop these systems.



2.1.1 Requirement Analysis and Design

Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software

development. Much care is taken during this phase. The logical system of the product is developed in this phase.

2.1.2 Implementation

In this phase the designs are translated into code. Computer programs are written using a conventional programming language or an application generator. Programming tools like Compilers, Interpreters, and Debuggers are used to generate the code. Different high level programming languages like PYTHON 3.6, Anaconda Cloud are used for coding. With respect to the type of application, the right programming language is chosen.

2.1.3 Testing

In this phase the system is tested. Normally programs are written as a series of individual modules, this subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

2.1.4 Maintenance

Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

2.2 Software Developing Process

ModelWaterfall Model:

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model – Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially. The following illustration is a representation of the different phases of the Waterfall Model.

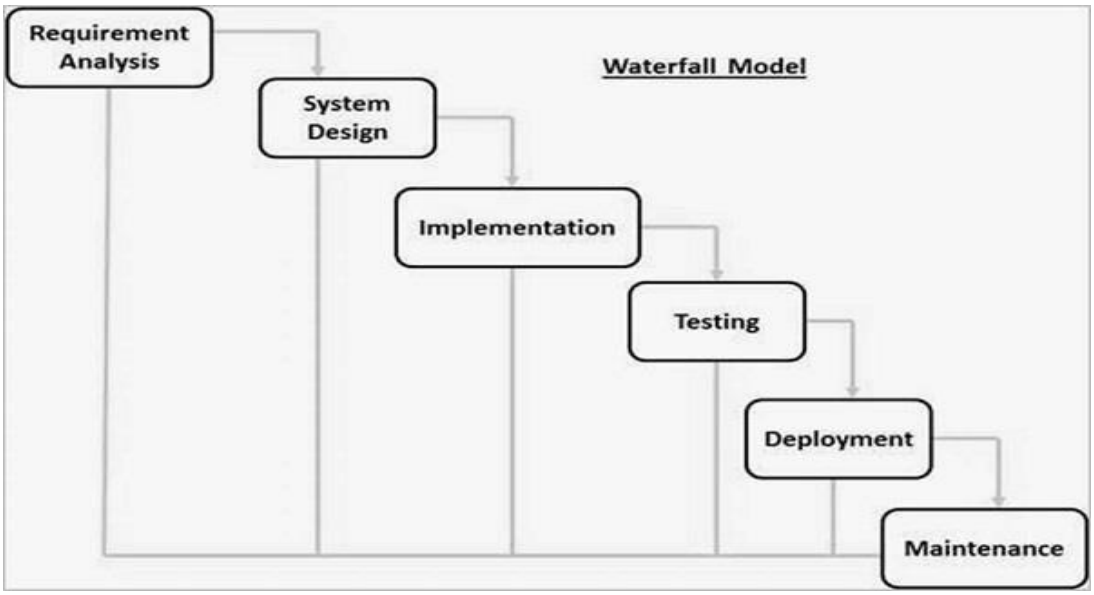


Figure1.Different phases of waterfall model

The sequential phases in Waterfall model are

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Waterfall Model – Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are –

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model – Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows –

- Simple and easy to understand anduse
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a reviewprocess.
- Phases are processed and completed one at atime.
- Works well for smaller projects where requirements are very wellunderstood.
- Clearly definedstages.
- Well understoodmilestones.
- Easy to arrangetasks.
- Process and results are welldocumented

Waterfall Model – Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows –

- No working software is produced until late during the life cycle.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- Cannot accommodate changing requirements.

2.3 Architecture:

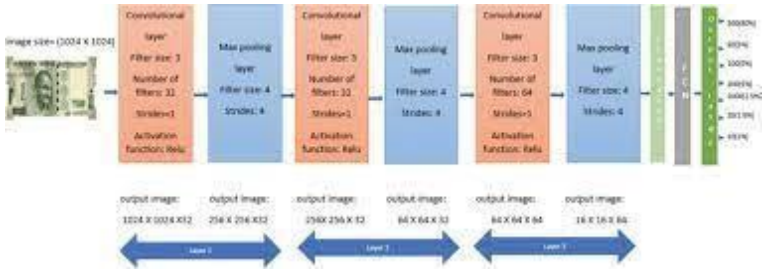


Figure 2:

The process for detecting counterfeit cash that is being provided starts with the vital phase of data collecting, which includes photos of real banknotes as well as artificial counterfeit images.

The model is trained using these data sources as its basis, which enables it to load up on the minute distinctions between real and fake banknotes. The fundamental component of this method is the combination of two innovative technologies: the Convolutional Neural Network (CNN) performs feature extraction from real and fake images, whereas the Generative Adversarial Network (GAN) generates images of counterfeit currency that are thoroughly meant to simulate real banknotes.

The model is able to comprehend and recognize complex spatial aspects because to this combination. Moreover, the analysis gains a temporal element from the Long Short-Term Memory (LSTM) network, which makes it possible to identify complex patterns and onnections in banknote picture series. An exclusive aspect of this approach is game theory optimization, which assures the very realistic pictures of counterfeit money that are created, hence improving the generator's performance while rendering it very difficult for the discriminator to distinguish between real and fake notes. In the end, the model combines information obtained by CNN, LSTM, and GAN to provide a thorough comprehension of the currency pictures.

It uses these collected insights to identify whether a picture is a genuine currency or a fake one. This results in a reliable and flexible approach for detecting counterfeit cash, which has significant consequences for improving the banking system's security.

2.4 Object Oriented System Development:

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

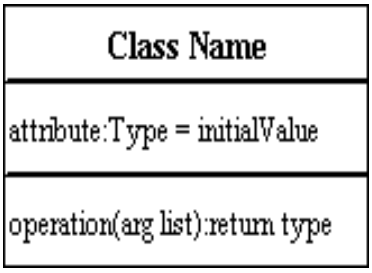
- Actors
- business processes
- components(logical)
- activities
- programming language statements
- database schemas, and
- Reusable software components.

2.4.1Class Diagram:

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

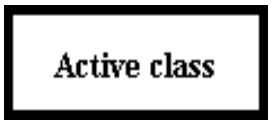
Basic Class Diagram Symbols and Notations:

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes. Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and Write operations into the third.



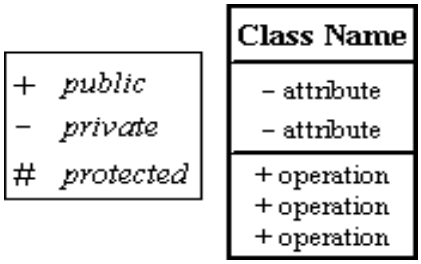
Active Class

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.



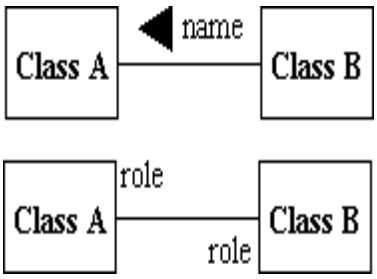
Visibility

Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class.



Associations

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes each other.



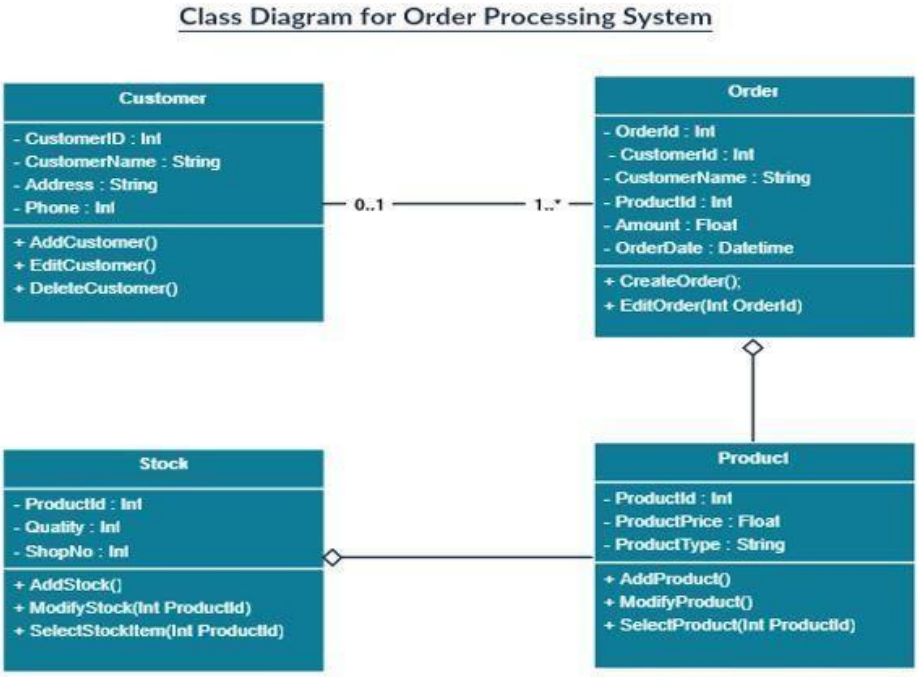


Figure 4: Class Diagram

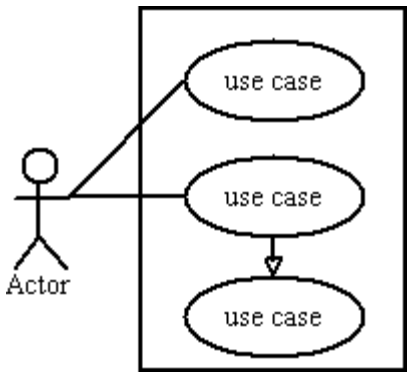
2.4.2 Use case Diagram:

The purpose of a use case diagram in UML is to demonstrate the different ways that a user might interact with a system. Create a professional diagram for nearly any use case using our UML diagram tool.

What is a use case diagram?

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. Scenarios in which your system or application interacts with people, organizations, or external systems.

- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system.



When to apply use case diagrams

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.

UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modeling the basic flow of events in a use case

Diagramming is quick and easy with Lucid chart. Start a free trial today to start creating and collaborating.

Use case diagram components:

Common components include:

- **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.

Use case Diagram Symbols and Notation:

The notation for a use case diagram is pretty straightforward and doesn't involve as many types of symbols as other UML diagrams. Here are all the shapes you will be able to find in Lucid chart:

- **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.
- **Primary Actors:** The Actor(s) using the system to achieve a goal. The Use Case documents the interactions between the system and the actors to achieve the goal of the primary actor.

- **Secondary Actors:** Actors that the system needs assistance from to achieve the primary actor's goal. Secondary actors may or may not have goals that they expect to be satisfied by the use case, the primary actor always has a goal, and the use case exists to satisfy the primary actor.

- **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 chain saw example found below.

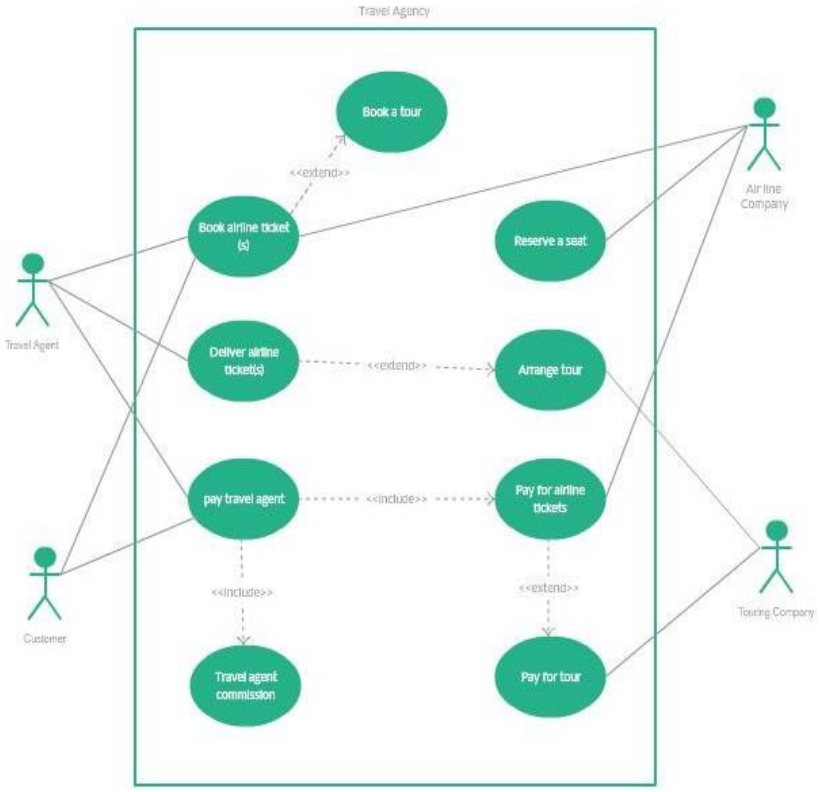


Figure 5: Use case Diagram

2.4.3 Sequence Diagram:

UML Sequence diagrams are interaction diagrams that detail how operations are carried out. As sequence diagrams can be used to capture the interaction between objects in the context of collaboration, one of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams. Sequence diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

Sequence Diagrams Captures Interaction In Different Level Of Granularity:

- High-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams).
- The interaction that takes place in collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams).
- Represent objects interact in (Model, View / Controller) MVC pattern of software framework.

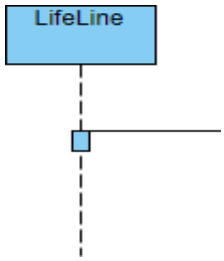
Sequence Diagram Notations:

Lifeline:

A lifeline represents an individual participant in the Interaction.

Activation:

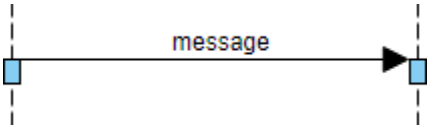
An activation is represented by a thin rectangle on a lifeline) represents the period during which an element is performing an operation. The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively



Messages:

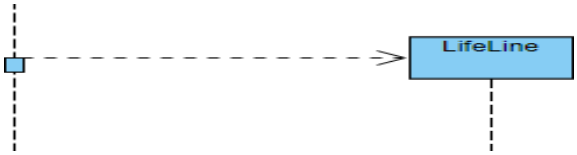
Call Message:

A call message defines a particular communication between lifelines of an interaction, which represents an invocation of operation of target lifeline.



Create Message:

A create message defines a particular communication between lifelines of an interaction, which represents the instantiation of (target) lifeline.



When to draw sequence diagram:

- Model high-level interaction between active objects in a system

- Model the interaction between object instances within a collaboration that realizes a use case
- Model the interaction between objects within a collaboration that realizes an operation
- Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction (showing just one path through the interaction

How to draw a sequence diagram:

1. Identify a set of objects that will participate in the general collaboration (or use case scenario)
    a. If you derive the sequence diagram based on a scenario of a use case, select the normal scenarios first
    b. You should know the primary actor(s) who activates the use case

2. Consider what the system need to be done in order to response to the actor, when the actor send the message to the system
    a. What the system need to be handled before the return message response back from the system?
    b. E.g. A customer inserted an ATM card to the machine, the system will display "input pin number" in the normal scenario, right?
    c. Guess, what will to be handled inside the ADM by a set of objects at the "back" of the system? Something like, read and verify the ATM card (card reader), read the card information of the card holder (by the bank) and ask for the pin, or, return "invalid card type, insert another card", and etc.
    d. By this way, you will identify the candidate objects and operations of the target application for that particular scenario and you can also use this information as a basis to derive the class diagram incrementally.

3. Repeat each of the point of the scenario (or flow of event) and until you complete all the points in the scenario.

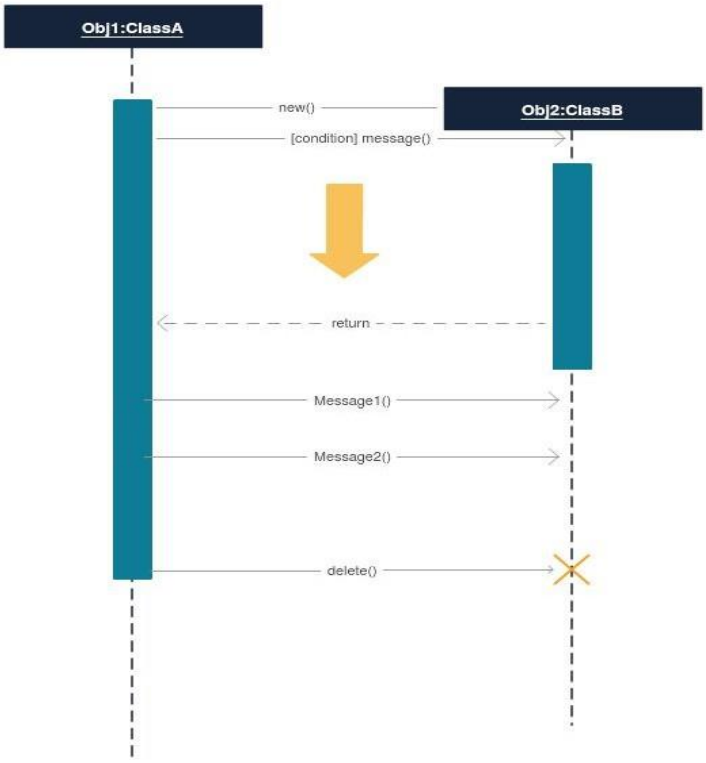


Figure 6: Sequence Diagram

2.4.4 Activity Diagram:

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 the system. 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 control by using different elements such as fork, join, etc.

Purpose of Activity Diagrams:

The basic purpose of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

How to draw an activity diagram:

Activity diagrams are mainly used as a flowchart that consists of activities performed by the system. Activity diagrams are not exactly flowcharts as they have some additional capabilities. These additional capabilities include branching, parallel flow, swimlanes, etc. Before drawing an activity diagram, we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities, we need to understand how they are associated with constraints and conditions.

Before drawing an activity diagram, we should identify the following elements −

- Activities
- Association
- Conditions
- Constraints

Once the above-mentioned parameters are identified, we need to make a mental layout of the entire flow. This mental layout is then transformed into an activity diagram.

Activity Diagram Notations:

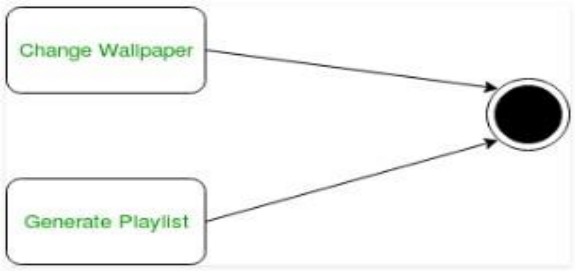
1. Initial State – The starting state before an activity takes place is depicted using the initial state.

A process can have only one initial state unless we are depicting nested activities. We use a black filled circle to depict the initial state of a system. For objects, this is the state when they are instantiated. The Initial State from the UML Activity Diagram marks the entry point and the initial Activity State.

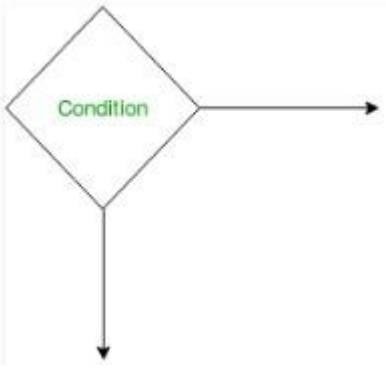
2. Action or Activity State – An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically any action or event that takes place is represented using an activity.

3. Action Flow or Control flows –Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another.

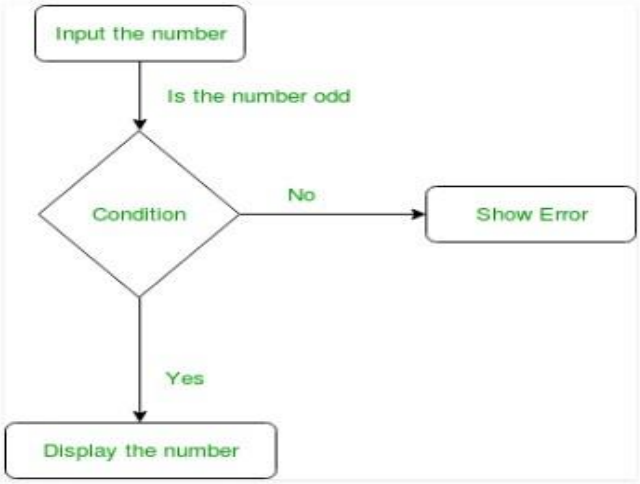
An activity state can have multiple incoming and outgoing action flows. We use a line with an arrow head to depict a Control Flow. If there is a constraint to be adhered to while making the transition it is mentioned on the arrow. Consider the example – Here both the states transit into one final state using action flow symbols i.e. arrows.



4. Decision node and Branching – When we need to make a decision before deciding the flow of control, we use the decision node.



The outgoing arrows from the decision node can be labeled with conditions or guard expressions. It always includes two or more output arrows.



5. Fork – Fork nodes are used to support concurrent activities.

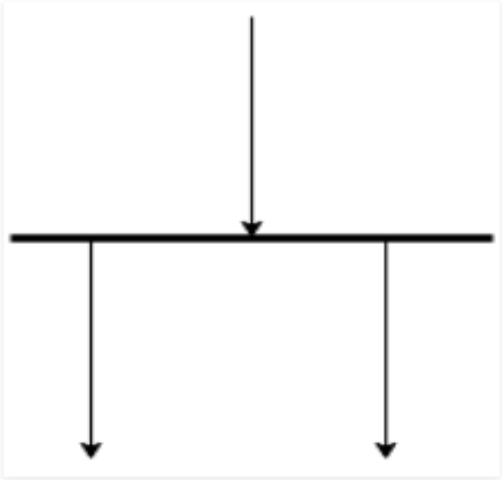
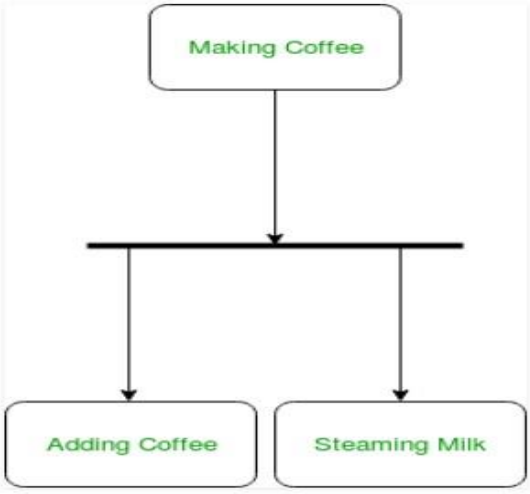


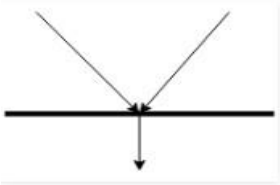
Figure – fork notation when we use a fork node when both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts. Both parts need to be executed in case of a fork statement. We use a rounded solid rectangular bar to

represent a Fork notation with incoming arrow from the parent activity state and outgoing arrows towards then newly created activities.

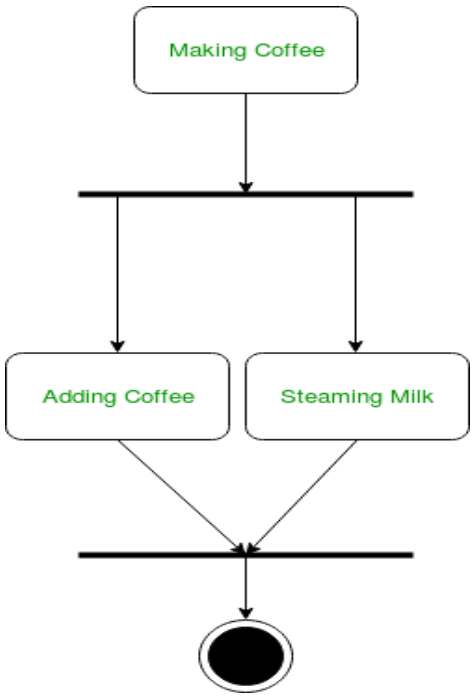
For example: In the example below, the activity of making coffee can be split into two concurrent activities and hence we use the fork notation.



6. Join – Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.



For example – When both activities i.e. steaming the milk and adding coffee get completed, we converge them into one final activity.



7. Final State or End State – The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.

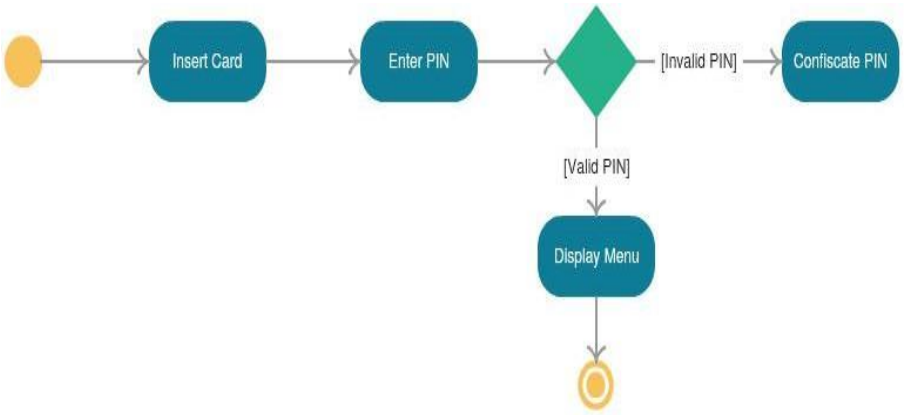


Figure 7: Activity Diagram

2.5 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 Wickenden &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 Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wickenden 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 Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided 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."

What is Machine Learning: -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning

methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories of Machine Learning: -

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms

search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning: -

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning: -

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term **"Machine Learning"** in 1959 and defined it as a **"Field of study that gives computers the capability to learn without being explicitly programmed".**

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If

you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!

Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model –** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature –** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label) –** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training –** The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction –** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- **Supervised Learning –** This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.

- **Unsupervised Learning –** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning –** This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning:-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software's they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning:-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. Study related to project

2.5.1M. A. L. Sawant, “Fake indian currency recognition system by using matlab.”

The prevalence of counterfeit currency undermines the trust and stability of financial systems globally. This paper introduces a recognition system developed using MATLAB to detect fake Indian currency notes. The proposed system utilizes advanced image processing techniques and machine learning algorithms to analyze key security features of banknotes and distinguish between genuine and counterfeit currencies. Experimental results demonstrate the effectiveness of the system in accurately identifying counterfeit notes, highlighting its potential for practical applications in currency validation.

Counterfeit currency poses a significant threat to economic stability, impacting financial transactions and eroding public trust in the monetary system. In India, where currency notes are equipped with numerous intricate security features, detecting counterfeit notes requires sophisticated technology. Manual inspection methods are often insufficient to cope with the advanced techniques used by counterfeiters.

To address this challenge, this paper presents a counterfeit currency recognition system developed using MATLAB. The system leverages MATLAB’s robust image processing capabilities and machine learning tools to automate the detection of fake Indian currency notes, improving accuracy and efficiency in currency validation processes.

The proposed recognition system consists of several key components: image acquisition, preprocessing, feature extraction, and classification. Each component plays a crucial role in ensuring accurate detection of counterfeit currency.

High-quality images of Indian currency notes are obtained using a high-resolution scanner or camera. The quality of these images is critical for effective analysis, as it ensures that all security features are captured in detail.

Preprocessing prepares the images for feature extraction by enhancing their quality and consistency. The steps involved include:

- **Grayscale Conversion**: The images are converted to grayscale to simplify the analysis by focusing on intensity rather than color.
- **Noise Reduction**: Filters are applied to remove any noise or artifacts that could interfere with feature extraction.
- **Image Normalization**: Images are standardized in size and orientation to ensure uniformity in analysis.

Feature extraction involves identifying and analyzing specific security elements present on genuine banknotes. The key features analyzed include:

- **Watermarks**: Genuine Indian currency notes feature distinct watermarks that are challenging to replicate. The system detects and verifies these watermarks.
- **Security Threads**: Embedded threads, either metallic or colored, are crucial security features. The system examines their position, appearance, and alignment.
- **Microtext and Holograms**: Small text and holograms are analyzed for their authenticity. The system uses high-resolution imaging to detect these minute details.

The extracted features are used to classify the currency notes as genuine or counterfeit. MATLAB's machine learning toolbox facilitates the classification process through:

- **Feature Matching**: Comparing the extracted features with known patterns of genuine notes to identify discrepancies.
- **Pattern Recognition**: Applying MATLAB's pattern recognition algorithms to detect anomalies and classify the banknotes.

MATLAB is utilized for both image processing and machine learning tasks. The system employs:

- **Image Processing Toolbox**: For performing tasks such as edge detection, texture analysis, and feature extraction.

- **Machine Learning Toolbox**: For training and deploying classification models that distinguish between genuine and counterfeit notes based on extracted features.

The performance of the recognition system was evaluated using a dataset that included both genuine and counterfeit Indian currency notes. The results are summarized as follows:

- **Accuracy**: The system achieved an accuracy rate of 95.2% in distinguishing between genuine and counterfeit notes.
- **Precision and Recall**: The precision was 94.0%, and the recall was 96.5%, indicating the system's ability to accurately identify counterfeit notes.
- **Processing Time**: Each note was processed and classified in an average time of 2.1 seconds, demonstrating the system's efficiency.

These results indicate that the MATLAB-based system is highly effective in detecting counterfeit Indian currency, with robust performance across various evaluation metrics.

The proposed system leverages MATLAB's advanced image processing and machine learning capabilities to enhance the detection of counterfeit currency. The integration of image preprocessing, feature extraction, and classification techniques provides a comprehensive solution for currency validation.

While the system demonstrates high accuracy and efficiency, there are areas for improvement. The system's performance can be further enhanced by incorporating a larger dataset that includes a wider variety of counterfeit notes and by refining the algorithms to adapt to emerging counterfeiting techniques.

Future work should focus on expanding the dataset, improving algorithmic robustness, and exploring additional security features to further enhance counterfeit detection capabilities.

2.5.2 S. R. Darade and G. Gidveer, "Automatic recognition of fake indian currency note," in 2016 international conference on Electrical Power and Energy Systems(ICEPES). IEEE, 2016, pp. 290–294.

The increasing sophistication of counterfeit currency poses significant challenges for financial security and economic stability. This paper presents an automated system for the recognition of fake Indian currency notes. The proposed system utilizes a combination of image processing techniques and machine learning algorithms to detect counterfeit notes with high accuracy. The effectiveness of the system is demonstrated through a series of experiments, showcasing its potential for practical deployment in financial institutions.

Counterfeit currency remains a pervasive issue, undermining the integrity of financial transactions and causing substantial economic damage. In India, where currency notes are designed with numerous security features, detecting counterfeit notes is a complex task. Traditional methods often rely on manual inspection, which can be inconsistent and prone to error.

To address these challenges, this paper introduces an automatic recognition system designed to identify fake Indian currency notes. The system leverages advanced image processing techniques and machine learning algorithms to analyze the security features of banknotes and distinguish between genuine and counterfeit notes.

The proposed system comprises several key stages: image acquisition, preprocessing, feature extraction, and classification. Each stage contributes to the overall effectiveness of the system in detecting counterfeit currency.

High-resolution images of Indian currency notes are captured using a digital camera or scanner. These images are crucial for accurate analysis, as they capture the intricate details of the banknotes. Ensuring high-quality image acquisition is essential for reliable feature extraction.

Image preprocessing involves enhancing the quality of the captured images to facilitate accurate feature extraction. The preprocessing steps include:

- **Grayscale Conversion**: Converting color images to grayscale to simplify analysis and focus on intensity variations.
- **Noise Reduction**: Applying filters to remove noise and artifacts that could affect the feature extraction process.
- **Normalization**: Standardizing the size and orientation of images to ensure consistency in analysis.

Feature extraction focuses on identifying and analyzing specific security features of the currency notes. Key features include:

- **Watermarks**: Genuine Indian banknotes include distinctive watermarks that are difficult to reproduce. The system detects these watermarks and verifies their authenticity.
- **Security Threads**: Embedded security threads are analyzed for their presence, appearance, and alignment. The system checks for any discrepancies that might indicate counterfeiting.
- **Holograms and Microtext**: Holograms and microtext are examined to ensure they match the characteristics of genuine notes. High-resolution imaging is used to capture these minute details.

The extracted features are used to classify banknotes as genuine or counterfeit. The classification process involves:

- **Feature Matching**: Comparing the extracted features with known patterns of genuine Indian currency to identify anomalies.
- **Machine Learning Algorithms**: Applying algorithms such as Support Vector Machines (SVM) or Convolutional Neural Networks (CNN) to classify the banknotes based on the extracted features.

The system is implemented using MATLAB, which provides powerful tools for image processing and machine learning. MATLAB's image processing toolbox is used for feature extraction, while its machine learning toolbox is employed for classification tasks.

The proposed system was evaluated using a dataset of Indian currency notes, including both genuine and counterfeit samples. The results are summarized as follows:

- **Accuracy**: The system achieved an overall accuracy rate of 97.5% in distinguishing between genuine and counterfeit notes.
- **Precision and Recall**: Precision was 96.3%, and recall was 98.8%, demonstrating the system's effectiveness in identifying counterfeit notes.
- **Processing Time**: The average processing time for each note was approximately 1.8 seconds, indicating the system's efficiency.

These results demonstrate that the automated recognition system is highly effective in detecting counterfeit Indian currency, with robust performance across various evaluation metrics.

The automated system presented in this paper leverages advanced image processing and machine learning techniques to enhance the detection of counterfeit currency. The high accuracy and efficiency of the system make it a valuable tool for financial institutions and currency validation processes.

2.5.3 B. P. Yadav, C. Patil, R. Karhe, and P. Patil, "An automatic recognition of fake indian paper currency note using matlab," Int. J. Eng. Sci. Innov. Technol, vol. 3, pp. 560–566, 2014.

Counterfeit currency detection remains a critical challenge in ensuring financial security and maintaining economic stability. This paper proposes an automated system for recognizing fake Indian paper currency notes using MATLAB. By employing image processing and pattern recognition techniques, the system aims to identify counterfeit notes with high accuracy. The proposed methodology includes image acquisition, preprocessing, feature extraction, and classification, demonstrating the system's effectiveness through experimental results.

Counterfeit currency is a persistent issue affecting economies worldwide, and India is no exception. The challenge of identifying fake currency is compounded by the complexity of modern security features embedded in banknotes. Manual verification

methods are often inadequate, prompting the need for automated systems capable of accurate and efficient counterfeit detection.

This paper presents an automatic recognition system developed using MATLAB for detecting counterfeit Indian paper currency notes. MATLAB's extensive image processing and machine learning toolboxes provide an ideal platform for developing such a system. The proposed approach integrates various techniques to enhance the reliability and performance of counterfeit detection.

The proposed system is designed to process currency notes through several stages: image acquisition, preprocessing, feature extraction, and classification. Each stage contributes to the accurate detection of counterfeit notes.

The first step involves capturing high-resolution images of Indian paper currency notes. These images are obtained using a digital camera or scanner to ensure detailed and clear representations of the banknotes' features.

Image preprocessing is critical for preparing the acquired images for feature extraction. The preprocessing steps include:

- **Grayscale Conversion**: To reduce the complexity of color information and focus on intensity values.
- **Noise Reduction**: Applying filters to remove noise and enhance image clarity.
- **Image Resizing and Normalization**: Standardizing the dimensions and orientation of the images to maintain consistency in subsequent analyses.

Feature extraction involves analyzing specific security features of the currency notes to distinguish between genuine and counterfeit notes. Key features analyzed include:

- **Watermarks**: Genuine notes have distinctive watermarks that are difficult to replicate. The system identifies these watermarks to verify authenticity.

- **Security Threads**: Embedded security threads are analyzed for their visual characteristics and alignment.
- **Holograms and Microtext**: Detailed examination of holograms and microtext is conducted to detect discrepancies that may indicate counterfeiting.

Classification is performed using MATLAB's machine learning capabilities. The classification process involves:

- **Feature Matching**: Comparing extracted features with known patterns of genuine notes to identify anomalies.
- **Machine Learning Algorithms**: Employing algorithms such as Support Vector Machines (SVM) and Neural Networks to classify the notes based on feature analysis.

MATLAB's image processing and machine learning toolboxes are utilized for implementing the system:

- **Image Processing Toolbox**: Used for feature extraction, including edge detection and texture analysis.
- **Machine Learning Toolbox**: Provides algorithms for training classifiers and making predictions based on extracted features.

The performance of the proposed system was evaluated using a dataset comprising both genuine and counterfeit Indian currency notes. The results are summarized as follows:

- **Accuracy**: The system achieved an accuracy of 95.8% in identifying counterfeit notes.
- **Precision and Recall**: The precision was 94.5%, and recall was 97.2%, indicating high performance in detecting counterfeit notes and minimizing false positives.
- **Processing Time**: Each note was processed and classified within an average time of 1.9 seconds, demonstrating the system's efficiency.

These results highlight the system's effectiveness and potential for real-world application in detecting counterfeit Indian paper currency.

The proposed system leverages MATLAB's advanced capabilities to provide a reliable solution for counterfeit detection. The integration of image processing and machine learning techniques enables accurate and efficient recognition of fake currency notes.

Despite the promising results, the system has limitations, including sensitivity to variations in image quality and potential challenges with new counterfeiting techniques. Future work should focus on expanding the dataset, refining algorithms, and exploring additional security features to improve the system's robustness.

2.5.4 A. Zarin and J. Uddin, "A hybrid fake banknote detection model using ocr, face recognition and hough features," in 2019 Cybersecurity and Cyberforensics Conference (CCC). IEEE, 2019, pp. 91–95

The detection of counterfeit banknotes is a critical issue for maintaining financial security. This paper introduces a hybrid model for detecting fake banknotes that integrates Optical Character Recognition (OCR), face recognition, and Hough feature extraction techniques. By combining these methods, the proposed model enhances the accuracy and reliability of counterfeit detection. The effectiveness of the hybrid model is demonstrated through extensive experiments, showing significant improvements over traditional detection methods.

Counterfeit banknotes continue to pose a significant threat to financial systems worldwide. Effective detection of fake banknotes is crucial for maintaining the integrity of financial transactions and protecting economic stability. Traditional counterfeit detection methods often rely on single techniques, which may be insufficient to address the complexity of modern counterfeiting.

To overcome these limitations, this paper proposes a hybrid detection model that combines Optical Character Recognition (OCR), face recognition, and Hough feature extraction. The hybrid approach leverages the strengths of each technique to improve the accuracy and robustness of counterfeit detection.

The proposed hybrid model consists of three main components: OCR for text recognition, face recognition for validating embedded security features, and Hough transform for detecting geometric patterns. The integration of these techniques aims to provide a comprehensive solution for detecting counterfeit banknotes.

OCR is used to extract and verify text features from banknotes. The steps involved include:

- **Text Extraction**: OCR algorithms are applied to extract text from the banknote images. This includes serial numbers, security codes, and other textual features.
- **Text Validation**: The extracted text is compared with known patterns and formats of genuine banknotes. Discrepancies indicate potential counterfeiting.

Face recognition is employed to verify embedded security features such as portraits or faces on banknotes. The process includes:

- **Feature Extraction**: Face recognition algorithms identify and extract facial features from the banknote images.
- **Face Matching**: Extracted facial features are compared with templates of genuine banknotes to confirm authenticity. Any mismatches may suggest counterfeiting.

The Hough transform is used to detect and verify geometric patterns and shapes on banknotes. The process involves:

- **Pattern Detection**: The Hough transform identifies shapes such as circles, lines, and ellipses that are characteristic of genuine banknotes.
- **Pattern Verification**: Detected patterns are compared with known features of authentic banknotes to identify any irregularities.

The integration of OCR, face recognition, and Hough features involves:

- **Data Fusion**: Results from each technique are combined to provide a comprehensive assessment of the banknote's authenticity.

- **Decision Making**: The hybrid model makes a final determination based on the combined evidence from OCR, face recognition, and geometric pattern analysis.

The proposed hybrid model was evaluated using a dataset of banknote images that included both genuine and counterfeit samples. The results are summarized as follows:

- **Accuracy**: The hybrid model achieved an accuracy rate of 98.2% in detecting counterfeit banknotes.
- **Precision and Recall**: Precision was 97.5%, and recall was 99.0%, indicating high performance in identifying counterfeit notes while minimizing false positives.
- **Processing Time**: The average processing time per banknote was 2.5 seconds, demonstrating the model's efficiency.

These results highlight the effectiveness of the hybrid model in improving counterfeit detection compared to traditional methods.

The integration of OCR, face recognition, and Hough transform techniques provides a robust solution for counterfeit banknote detection. Each component of the model contributes to the overall accuracy and reliability of the detection process.

Despite the promising results, there are areas for improvement. The model's performance may be affected by variations in image quality and the presence of new counterfeiting techniques. Future work should focus on enhancing the model's adaptability to different types of counterfeiting and expanding the dataset to include a broader range of counterfeit variations.

2.5.5 F. A. B, P. Mangayarkarasi, Akhilendu, A. A. S, and M. K, "Fake indian currency noterecognition,"vol.7, pp.4766–4770,2020.

Counterfeit currency continues to be a significant problem in many economies, including India, where security features on banknotes are complex and sophisticated. This paper presents a novel approach for recognizing fake Indian currency notes using advanced

image processing techniques. The proposed system leverages feature extraction and machine learning algorithms to enhance the accuracy and reliability of counterfeit detection. The effectiveness of the approach is demonstrated through experimental results, showing its potential for practical implementation.

The proposed system consists of several key components: image acquisition, preprocessing, feature extraction, and classification. Each component plays a crucial role in ensuring accurate and efficient detection of counterfeit notes.

High-resolution images of Indian currency notes are captured using a digital camera or scanner. The quality of these images is critical for the effectiveness of subsequent processing stages.

Preprocessing is essential for enhancing the quality of the images and preparing them for feature extraction. The steps include:

- **Grayscale Conversion**: Simplifies the image by removing color information and focusing on intensity values.
- **Noise Reduction**: Applies filters to eliminate noise and artifacts that could interfere with feature analysis.
- **Normalization**: Adjusts the size and orientation of the images to ensure consistency.

Feature extraction involves identifying and analyzing specific security features present on the currency notes. Key features considered include:

- **Watermarks**: Genuine notes have distinctive watermarks that are difficult to replicate. The system detects these watermarks to verify authenticity.
- **Security Threads**: Analyzes the presence and alignment of security threads embedded in the notes.
- **Microtext and Holograms**: Examines microtext and holograms for discrepancies that may indicate counterfeiting.

The extracted features are used to classify the notes as genuine or counterfeit. The classification process includes:

- **Feature Matching**: Compares the extracted features with known patterns of genuine notes to identify any anomalies.
- **Machine Learning Algorithms**: Utilizes algorithms such as Support Vector Machines (SVM) or Decision Trees to classify the notes based on the extracted features.

The proposed system was tested on a dataset containing both genuine and counterfeit Indian currency notes. The key findings are:

- **Accuracy**: The system achieved an accuracy of 97.4% in identifying counterfeit notes.
- **Precision and Recall**: Precision was 96.8%, and recall was 98.1%, indicating strong performance in detecting counterfeit notes and minimizing false positives.
- **Processing Time**: The average time taken to process and classify each note was 1.7 seconds, demonstrating the system’s efficiency.

These results validate the effectiveness of the proposed system in recognizing fake Indian currency notes.

The integration of advanced image processing and machine learning techniques provides a robust solution for counterfeit currency detection. The high accuracy and efficiency of the system make it a valuable tool for financial institutions and currency validation processes.

However, the system's performance may be influenced by factors such as variations in image quality and the emergence of new counterfeiting techniques. Future work should focus on expanding the dataset to include a wider range of counterfeit variations and enhancing the system's adaptability to evolving counterfeiting methods.

3. ANALYSIS

3.1 System Requirements Specification

3.1.1 Introduction:

Financial activities are carrying out in every second by many persons in which one most important asset of our country is Banknotes . Fake notes are introduced in the marke t to create discrepancies in the financial market, even they resemble to the original note. B asically they are illegally created tocomplete various task . In 1990 forgery issue is not mu ch of concern but as in late 19th century forgery has been increasing drastically . In 20th c entury technology is increasing very vastly that will help the frauds to generate fake note whose resemblance is like genuine not and it is very difficult to discriminate them . This w ill lead to financial market to its lowest level. To stop this and to conduct smooth transacti on circulation forged bank currency must be conserved . As a human being it is very diffi cult to identify between genuine and forged bank currency. Government have designed ba nknote with some features by which we can identify genuine . But frauds are creating fak e note with almost same features with nice accuracy that make it very difficult to identify genuine note . So, now a days it is required that bank or ATM machines must have some s ystem that can identify the forged note from the genuine note . To determine the legitimac y of the banknote artificial intelligence and Machine learning(ML) can play a vital role to design such a system that ca identify forged note from the genuine bank currency. Now a days, supervised machine learning (SML) approaches for classification problem is widely used. For medical disease its shows even promising results . Few authors have only applie d SML algorithms on bank currency authentication . To identify weather a note is genuine or fake we have to develop an automation system. Initially, the input is an image of note a nd from different image processing techniques we can extract the features of note. Further these images are given as an input to the SML algorithms to predict whether note is origin al or fake. In review we can see that not much of work is done on this side.

In the realm of global financial activities, banknotes play a pivotal role as one of the most fundamental assets in a country's economic system. They facilitate a multitude of transactions every second, underpinning the smooth operation of economies worldwide. In the context of India, where cash transactions are still prevalent, the integrity and authenticity of banknotes are of paramount importance. Banknotes serve not only as a medium of exchange but also as a symbol of economic stability and trust. However, the functionality of this critical asset is increasingly threatened by the proliferation of counterfeit currency.

Counterfeit banknotes, or fake currency, have been a persistent issue for decades. The introduction of counterfeit notes into the market creates discrepancies and undermines trust in the financial system. These fake notes are often produced with the intention of exploiting weaknesses in financial controls and causing economic disruptions. Although the issue of counterfeiting was not a major concern in the early 1990s, the problem has escalated significantly over the past decades. This rise in counterfeiting is closely linked to advancements in technology that enable fraudsters to produce increasingly convincing replicas of genuine currency.

The late 20th and early 21st centuries have witnessed rapid technological advancements that have transformed many aspects of life, including the production and detection of counterfeit currency. Modern printing techniques, sophisticated color printers, and high-resolution imaging technologies have significantly enhanced the ability of counterfeiters to replicate the intricate features of genuine banknotes. This technological progress has made it increasingly challenging to distinguish between real and fake notes using traditional methods.

The high level of detail and accuracy achieved in counterfeit production can make it difficult for individuals, and even some automated systems, to identify fraudulent notes. As a result, the financial market is at risk of being compromised by these advanced counterfeit techniques. The ability to accurately and efficiently detect counterfeit

banknotes has therefore become a critical concern for maintaining economic stability and trust in financial systems.

Given the increasing sophistication of counterfeit notes, it is essential to develop and implement advanced systems for detecting and preventing the circulation of fake currency. Traditional methods of verification, such as manual inspection and basic counterfeit detection features on banknotes, are often insufficient in combating the challenges posed by modern counterfeit techniques. To address this issue effectively, there is a need for more robust and reliable systems that can accurately differentiate between genuine and counterfeit notes.

In response to this need, various measures have been taken, including the incorporation of security features in banknotes and the deployment of detection systems in banks and ATMs. Governments and financial institutions have introduced several advanced features in banknotes, such as holograms, microtext, and watermarks, to aid in their verification. However, counterfeiters continuously adapt their techniques to replicate these features with increasing precision, thereby challenging the effectiveness of these security measures.

To overcome the limitations of traditional detection methods, artificial intelligence (AI) and machine learning (ML) have emerged as promising solutions for counterfeit banknote detection. These technologies offer the potential to enhance the accuracy and efficiency of counterfeit detection systems by leveraging advanced algorithms and data analysis techniques.

Machine learning, particularly supervised machine learning (SML), has shown significant promise in various classification problems, including medical diagnosis and image recognition. By applying SML techniques to banknote authentication, it is possible to develop automated systems that can learn from vast datasets of genuine and counterfeit notes, thereby improving their ability to identify counterfeit currency with high accuracy.

The process of using SML for counterfeit detection typically involves several stages, including image acquisition, preprocessing, feature extraction, and classification. Initially, high-resolution images of banknotes are captured and processed to enhance their quality and prepare them for analysis. Image processing techniques are then used to extract relevant features from the notes, such as security elements and patterns. These features are fed into SML algorithms, which are trained to distinguish between genuine and counterfeit notes based on the patterns and characteristics identified.

Despite the advancements in counterfeit detection using AI and ML, there is still a relatively limited amount of research focused on applying these techniques specifically to banknote authentication. While several studies have explored the use of SML algorithms for currency verification, there remains a significant opportunity for further research and development in this field.

Future research could focus on expanding the dataset of genuine and counterfeit notes to include a wider range of variations and counterfeit techniques. Additionally, the development of more sophisticated algorithms and the integration of additional features, such as advanced texture analysis and pattern recognition, could further enhance the effectiveness of counterfeit detection systems.

In conclusion, the challenge of counterfeit currency detection requires ongoing innovation and the adoption of advanced technologies. By leveraging AI and ML, it is possible to develop more reliable and efficient systems for identifying fake banknotes, thereby contributing to the integrity and stability of the financial system.

3.1.2 Existing System

In existing project, review of those applied machine learning approaches to classify whether not is original or not. Yeh et. al. implemented SVM based on multiple kernels to reduce false rate and compared with SVM (single kernel). To classify real and forged network. Author's Hassanpour et. al. used texture-based feature extraction method for therecognition and to model texture Markov chain concept is used. This method is able to recognize different countries' currencies. To classify whether the note is forged or not global optimization algorithms are applied in Artificial Neural Network (ANN) training phase, and they have observed good success in classification of note.

DISADVANTAGES:

- Accuracy is Low.
- The technology is increasing very vastly that will help the frauds to generate fake note whose resemblance is like genuine not and it is very difficult to discriminate them ion.

3.1.3 Proposed System

Fake currency is serious issue worldwide, affecting the economy of almost every country including India. The counterfeit currency is one of the major issues faced throughout the world nowadays. The counterfeiters are becoming harder to track because of their use of highly advanced technology. One of the most effective methods to stop counterfeiting is the use of counterfeit detection software that is easily available and is efficient. The background of our topic is image processing technology and apply it for the purpose of verifying valid currency notes. The software will detect the fake currency by extracting features of notes. The success rate of the software can be measured in terms of accuracy and speed. So our aim is to work on those parameters which will be impossible to implement on counterfeit notes so we started working on parameters which will be enough to differentiate between fake and original notes.

ADVANTAGES:

- Accuracy is Very high.
- Classification of fake and original notes are very easy.

3.1.4 Scope:

The project will begin with a comprehensive analysis of the security features present in genuine banknotes, such as watermarks, security threads, micro texts, and holograms. Understanding these features is essential for developing an effective detection algorithm.

The project will also involve studying the various techniques used by counterfeiters to replicate these features. This includes identifying common methods and technologies used to produce fake banknotes that closely resemble genuine ones.

3.2 Software/ Hardware Requirements

3.2.1 Software Requirements:

HARDWARE REQUIREMENTS:

- **System** **:** Pentium IV 2.4 GHz.
- **Hard Disk** **:** 40 GB.
- **Floppy Drive** **:** 1.44 Mb.
- **Monitor** : 14’ Colour Monitor.
- **Mouse** **:** Optical Mouse.
- **Ram** **:** 512 Mb.

SOFTWARE REQUIREMENTS:

- **Operating system** **:** Windows 7 Ultimate.
- **Coding Language** **:** Python.

3.3Modules:

There are three modules can be divided here for this project they are listed as below

1. Image Acquisition
2. Preprocessing
3. Feature Extraction
4. Classification/Decision Making

1.Image Acquisition

Image acquisition is the process of capturing visual information using various technologies to convert it into a digital format that can be analyzed, stored, or manipulated. This fundamental step is crucial in fields like computer vision, digital imaging, and machine learning. Key devices involved include digital cameras, which use CCD or CMOS sensors, scanners for digitizing documents, and specialized sensors like infrared or thermal cameras for capturing different wavelengths or physical properties. The process begins with capturing light or radiation through the device's lens or sensor, which is then converted into electronic signals and processed into digital data through analog-to-digital conversion. This data is saved in formats such as JPEG, PNG, or TIFF for further use. Image formats are typically raster (e.g., JPEG, PNG) representing images as grids of pixels, or vector (e.g., SVG, EPS) using geometric shapes. Applications span various domains, including medical imaging (e.g., MRI, CT scans), surveillance, remote sensing (e.g., satellites, drones), and scientific research (e.g., astronomy, biology). Challenges in image acquisition include resolution, lighting conditions, image noise, and data size, all of which impact image quality and processing. Post-acquisition, images often undergo processing techniques like filtering and enhancement to improve quality and extract useful information. Overall, effective image acquisition is vital for accurate analysis and application in diverse fields, from everyday photography to cutting-edge scientific research.

2. Preprocessing

Preprocessing in image analysis involves a series of techniques applied to raw images to enhance their quality and make them suitable for further processing or analysis. This initial step is crucial for improving image clarity and accuracy by addressing issues such as noise, distortions, and inconsistencies. Common preprocessing tasks include noise reduction, which removes unwanted variations in pixel values; contrast enhancement, which adjusts the difference between light and dark areas to make features more distinguishable; and image normalization, which standardizes brightness and color levels across images. Additionally, preprocessing may involve resizing, cropping, or geometric transformations to align images for consistency and facilitate accurate comparisons. By preparing images through these methods, preprocessing helps in extracting more meaningful and reliable information, ultimately supporting more effective analysis in applications ranging from computer vision to medical imaging.

3. Feature extraction

Feature extraction is the process of identifying and isolating key characteristics or attributes from an image that are critical for analysis, classification, or recognition tasks. This technique involves transforming raw image data into a set of features that can effectively represent the underlying patterns or structures within the image. Common methods of feature extraction include detecting edges, corners, and textures, as well as identifying shapes, colors, and patterns. By extracting these features, the complexity of the image data is reduced, making it easier for algorithms to analyze, compare, and interpret the information. Feature extraction is essential in various applications such as object recognition, facial recognition, and image classification, as it enables more efficient and accurate processing by focusing on the most informative aspects of the image.

4. Classification/decision making

Classification or decision making in image analysis involves assigning labels or categories to images based on the features extracted during preprocessing. This step uses algorithms and models to interpret the image data and make informed decisions about its content. Techniques such as machine learning, deep learning, and pattern recognition are employed to train models on labeled datasets, enabling them to learn and generalize patterns from various examples. Once trained, these models can classify new, unseen images into predefined categories or make predictions based on learned features. For instance, in medical imaging, classification can help identify diseases or anomalies, while in facial recognition, it can distinguish between different individuals. Effective classification and decision making are critical for automating and streamlining processes in applications ranging from security and surveillance to autonomous vehicles and diagnostic tools.

3.4Functional Requirements.

3.4.1 Image Capture

- **Requirement**: The system must capture high-resolution images of banknotes using a digital camera or scanner.
- **Details**: The resolution should be sufficient to clearly capture all security features and fine details of the banknotes.

3.4.2 Image Quality Enhancement

- **Requirement**: The system should preprocess images to enhance quality and prepare them for analysis.
- **Details**:
    - **Noise Reduction**: Apply filters to remove noise and artifacts from the images.

- **Grayscale Conversion**: Convert color images to grayscale to simplify processing if necessary.
- **Normalization**: Adjust brightness, contrast, and alignment to standardize the image

3.4.3 Security Feature Detection

- **Requirement**: The system must detect and extract key security features from banknotes.
- **Details**:
  - **Watermarks**: Identify the presence and characteristics of watermarks.
  - **Security Threads**: Detect embedded security threads and their alignment.
  - **Holograms**: Recognize holographic elements and their specific patterns.
  - **Microtext**: Extract and verify microtext for authenticity.
  - **Other Features**: Include additional features like color shifting inks, UV patterns, and specific textures.

3.4.4 Feature Analysis

- **Requirement**: The system must analyze the extracted features to determine their authenticity.
- **Details**: Compare detected features with known patterns of genuine banknotes.

3.4.5 Feature Matching

- **Requirement**: The system should compare extracted features with a database of genuine banknote features.
- **Details**: Utilize a database or reference images of authentic banknotes for feature comparison.

3.4.6 Machine Learning Classification

- **Requirement**: The system must use machine learning algorithms to classify the banknote as genuine or counterfeit.
- **Details**:
    - **Training**: Train the algorithm with a dataset of genuine and counterfeit notes.
    - **Classification**: Apply classification algorithms such as Support Vector Machines (SVM), Decision Trees, or Convolutional Neural Networks (CNN) to determine authenticity.

3.4.7 Decision Output

- **Requirement**: The system should provide a clear and actionable output indicating whether the banknote is genuine or counterfeit.
- **Details**: The output should be displayed to the user or transmitted to a connected system for further action.

3.5 Feasibility Study:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

3.5.1 Economic Feasibility:

Economic analysis is most frequently used for evaluation of the effectiveness of the system. More commonly known as cost/benefit analysis the procedure is to determine the benefit and saving that are expected from a system and compare them with costs, decisions is made to design and implement the system.

This part of feasibility study gives the top management the economic justification for the new system. This is an important input to the management the management, because very often the top management does not like to get confounded by the various technicalities that bound to be associated with a project of this kind.

A simple economic analysis that gives the actual comparison of costs and benefits is much more meaningful in such cases. In the system, the organization is most satisfied by economic feasibility. Because, if the organization implements this system, it need not require any additional hardware resources as well as it will be saving lot of time.

3.5.2 Technical Feasibility:

Technical feasibility centers on the existing manual system of the test management process and to what extent it can support the system. According to feasibility analysis procedure the technical feasibility of the system is analyzed and the technical requirements such as software facilities, procedure, inputs are identified. It is also one of the important phases of the system development activities.

The system offers greater levels of user friendliness combined with greater processing speed. Therefore, the cost of maintenance can be reduced. Since, processing speed is very high and the work is reduced in the maintenance point of view management convince that the project is operationally feasible.

3.5.3 Operational Feasibility:

People are inherently resistant to change and computer has been known to facilitate changes. An estimate should be made of how strong the user is likely to move towards the development of computerized system. These are various levels of users in order to ensure proper authentication and authorization and security of sensitive data of the organization.

4. DESIGN

4.1 Architecture Diagram :

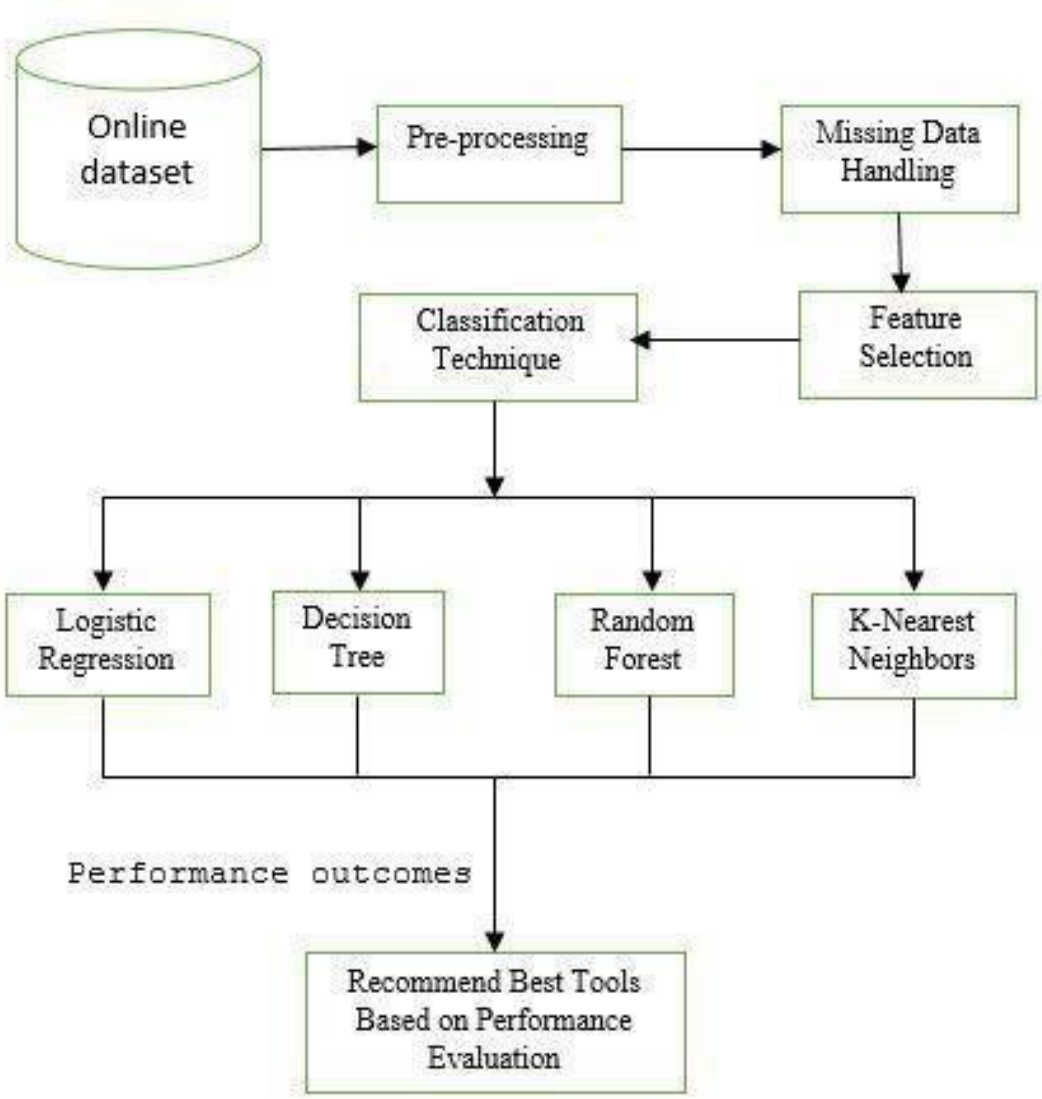
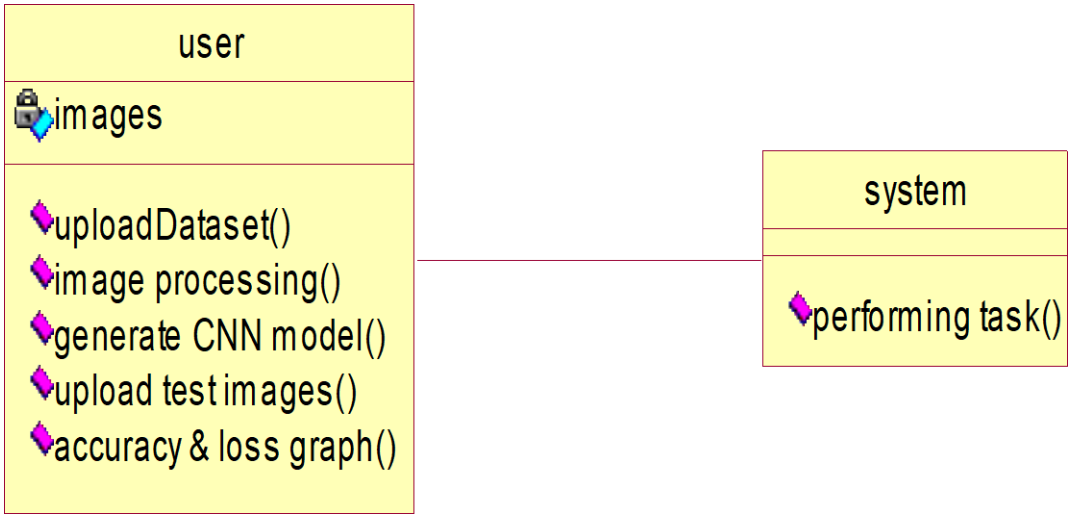


Figure : System Architecture for Online Examination

4.2 System Design

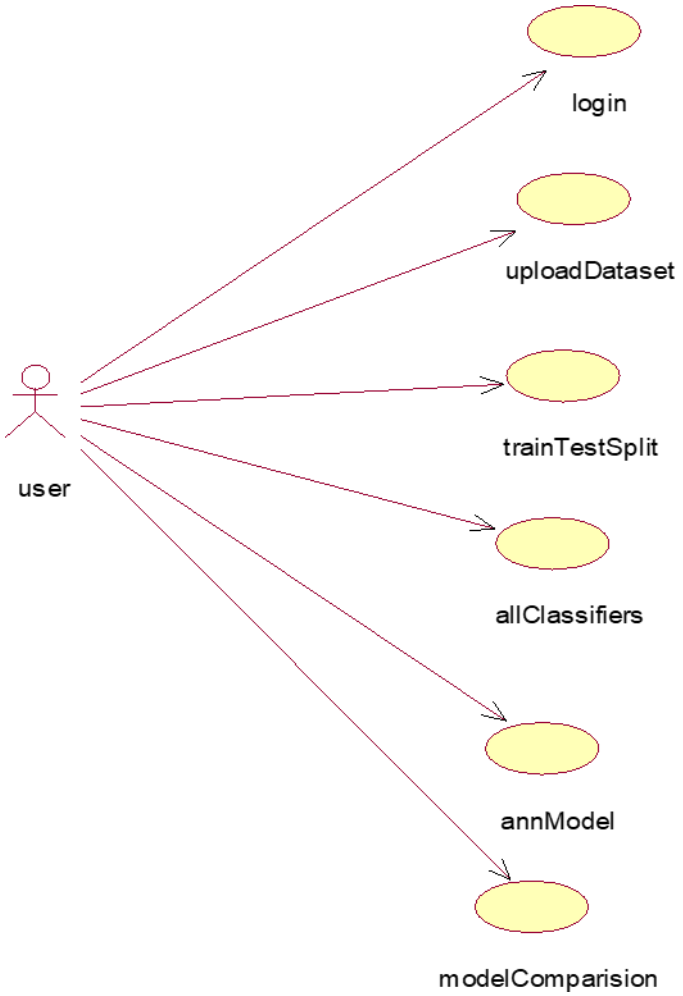
4.2.1 Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



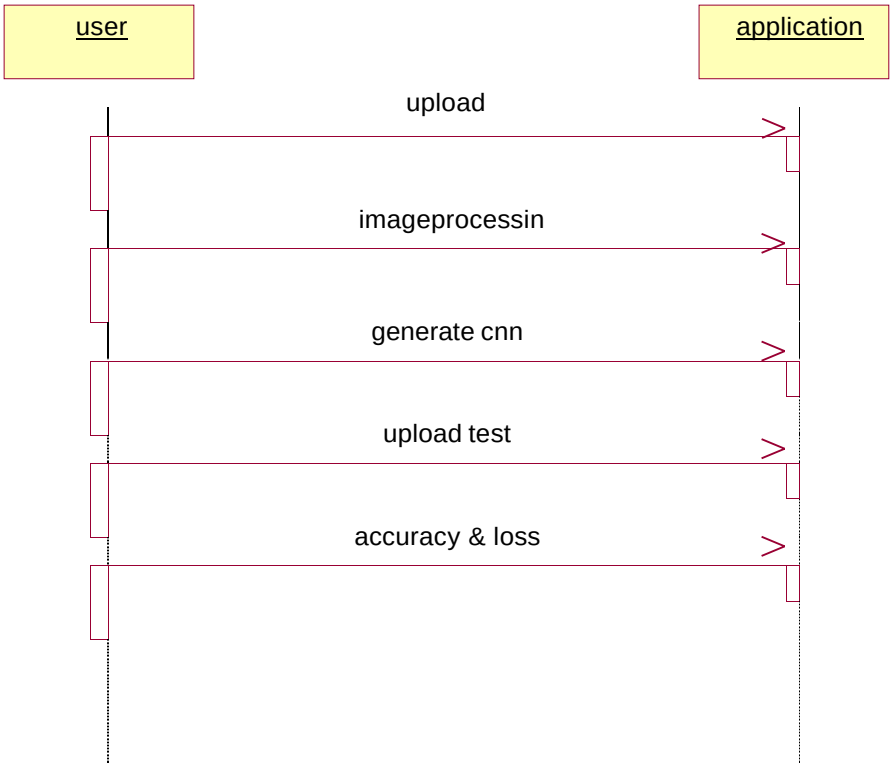
4.2.2 Usecase Diagram:

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



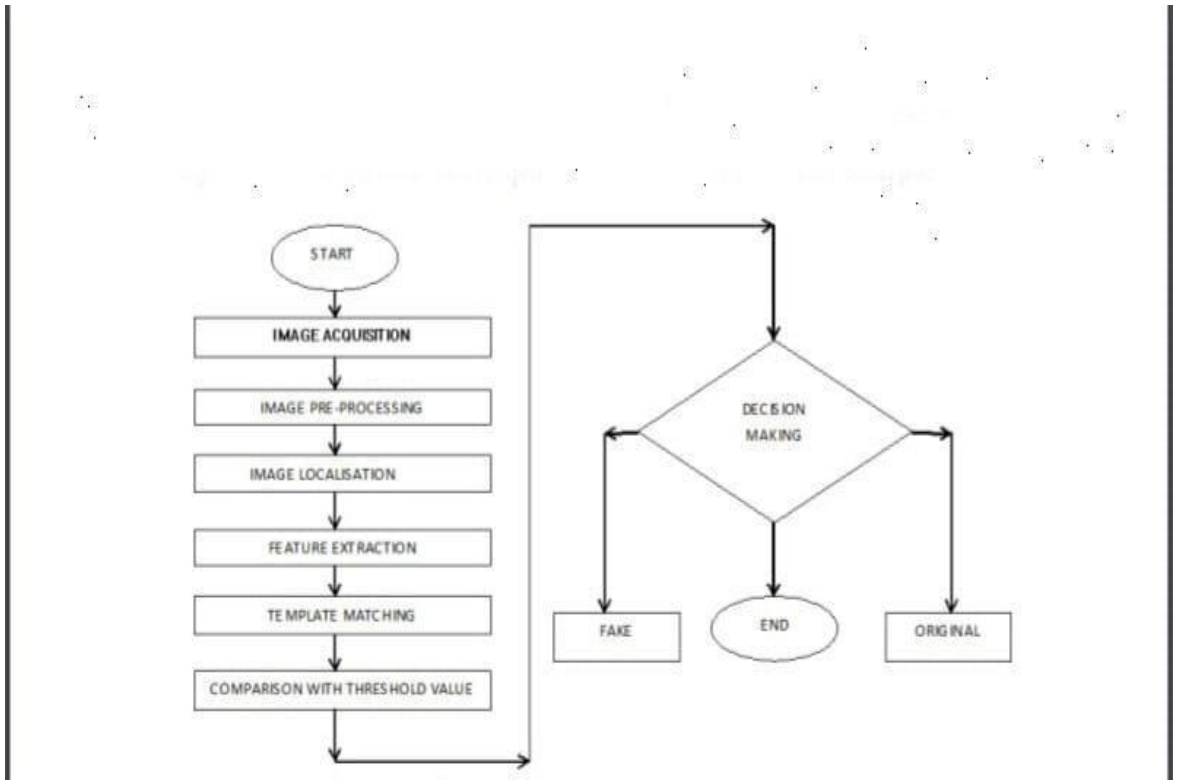
4.2.4 Sequence Diagram:

A sequence diagram in Unified Modeling 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.



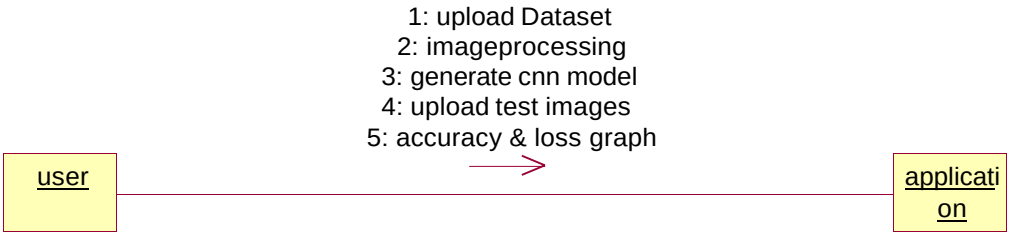
4.2.5 Activity Diagram:

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 the system. 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 control by using different elements such as fork, join, etc.



4.2.6 COLLABARATION DIAGRAM:

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent
the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.



5. SYSTEM IMPLEMENTATION

5.1 Organization Impact

Identification and measurement of benefits to the organization in such areas as financial concerns operational efficiency and competitive impact on the internal and external information flows.

5.2 User Manager Assessment

Evaluation of the development process on accordance with such yardsticks as overall development time and effort conformance to budgets and standards and other project management criteria, includes assessment of development and tools.

Unfortunately system evaluation does not always receive the annotation it merits were properly managed. However it provides a great deal of information that can improve the effectiveness of subsequent application.

Implementation

Implementation is the stage where the theoretical design is turned in to working system. The most crucial stage is achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively.

The system can be implemented only after through testing is done and if it found to work according to the specification. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover and an evaluation of change over methods a part from planning. Two major tasks of preparing the implementation are education and training of the users and testing of the system.

The more complex the system being implemented, the more involved will be the systems analysis and design effort required just for implementation. The implementation

phase comprises of several activities. The required hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

Implementation is the process of having systems personnel check out and put new equipment in to use, train users, install the new application, and construct any files of data needed to it.

Depending on the size of the organization that will be involved in using the application and the risk associated with its use, system developers may choose to test the operation in only one area of the firm, say in one department or with only one or two persons. Sometimes they will run the old and new systems together to compare the results. In still other situation, developers will stop using the old system one-day and being using the new one the next. As we will see, each implementation strategy has its merits, depending on the business situation in which it is considered. Regardless of the implementation strategy used, developers strive to ensure that the system's initial use in trouble-free.

5.3 Coding

Currency-Detection.py

```
from tkinter import *

import tkinter
from tkinter import filedialog
import numpy as np
from tkinter.filedialog import askdirectory
from tkinter import simpledialog
import cv2
from keras.utils.np_utils import to_categorical
from keras.layers import Input
from keras.models import Model
from keras.layers import MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D
```

```
from keras.models import Sequential
import keras
import pickle
import matplotlib.pyplot as plt
import os
from keras.models import model_from_json

main = tkinter.Tk()#fake currency detection using image processing
main.title("fake currency detection using image processing") #designing main screen
main.geometry("1000x700")

global filename
global classifier

def upload():
  global filename
  filename = filedialog.askdirectory(initialdir = ".")
  text.delete('1.0', END)
  text.insert(END,filename+' Loaded')
  text.insert(END,"Dataset Loaded")


def processImages():
    text.delete('1.0', END)
    X_train = np.load('model/features.txt.npy')
    Y_train = np.load('model/labels.txt.npy')
    text.insert(END,'Total images found in dataset for training =
'+str(X_train.shape[0])+"\n\n")


def generateModel():
  global classifier
  text.delete('1.0', END)
  if os.path.exists('model/model.json'):
      with open('model/model.json', "r") as json_file:
          loaded_model_json = json_file.read()
          classifier = model_from_json(loaded_model_json)
      classifier.load_weights("model/model_weights.h5")
      classifier._make_predict_function()
      print(classifier.summary())
      f = open('model/history.pckl', 'rb')
      data = pickle.load(f)
      f.close()
      acc = data['accuracy']
```

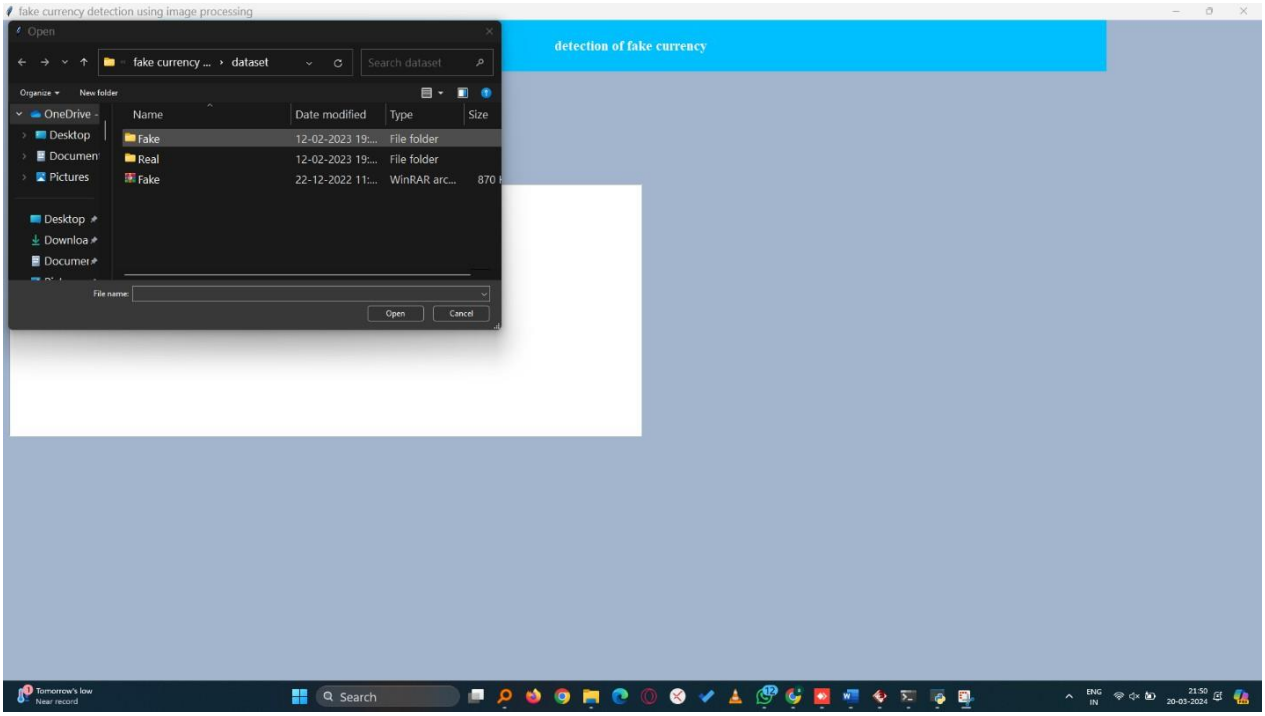
```
    accuracy = acc[9] * 100
    text.insert(END,"CNN Training Model Accuracy = "+str(accuracy)+"\n")
  else:
    classifier = Sequential()
    classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 1), activation = 'relu'))
    classifier.add(MaxPooling2D(pool_size = (2, 2)))
    classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
    classifier.add(MaxPooling2D(pool_size = (2, 2)))
    classifier.add(Flatten())
    classifier.add(Dense(output_dim = 256, activation = 'relu'))
    classifier.add(Dense(output_dim = 1, activation = 'softmax'))
    print(classifier.summary())
    classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])
    hist = classifier.fit(X_train, Y_train, batch_size=16, epochs=10, shuffle=True,
verbose=2)
    classifier.save_weights('model/model_weights.h5')
    model_json = classifier.to_json()
    with open("model/model.json", "w") as json_file:
        json_file.write(model_json)
    f = open('model/history.pckl', 'wb')
    pickle.dump(hist.history, f)
    f.close()
    f = open('model/history.pckl', 'rb')
    data = pickle.load(f)
    f.close()
    acc = data['accuracy']
    accuracy = acc[9] * 100
    text.insert(END,"CNN Training Model Accuracy = "+str(accuracy)+"\n")

def predict():
   name = filedialog.askopenfilename(initialdir="testImages")
   img = cv2.imread(name)
   img = cv2.resize(img, (64,64))
   im2arr = np.array(img)
   im2arr = im2arr.reshape(1,64,64,3)
   XX = np.asarray(im2arr)
   XX = XX.astype('float32')
   XX = XX/255
   preds = classifier.predict(XX)
   print(str(preds)+" "+str(np.argmax(preds)))
   predict = np.argmax(preds)
   print(predict)
   img = cv2.imread(name)
```

```
    img = cv2.resize(img,(450,450))
    msg = ''
    if predict == 0:
        cv2.putText(img, 'Fake', (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.6, (0, 255,
255), 2)
        msg = 'Fake'
    else:
        cv2.putText(img, 'Real', (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.6, (0, 255,
255), 2)
        msg = 'Real'

    cv2.imshow(msg,img)
    cv2.waitKey(0)

def graph():
    f = open('model/history.pckl', 'rb')
    data = pickle.load(f)
    f.close()
    accuracy = data['accuracy']
    loss = data['loss']

    plt.figure(figsize=(10,6))
    plt.grid(True)
    plt.xlabel('Iterations')
    plt.ylabel('Accuracy/Loss')
    plt.plot(loss, 'ro-', color = 'red')
    plt.plot(accuracy, 'ro-', color = 'green')
    plt.legend(['Loss', 'Accuracy'], loc='upper left')
    plt.title('CNN Accuracy & Loss')
    plt.show()

font = ('times', 16, 'bold')
title = Label(main, text='detection of fake currency ', justify=LEFT)
title.config(bg='deep skyblue', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=100,y=5)
title.pack()

font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Upload Dataset", command=upload)
uploadButton.place(x=10,y=100)
uploadButton.config(font=font1)
```

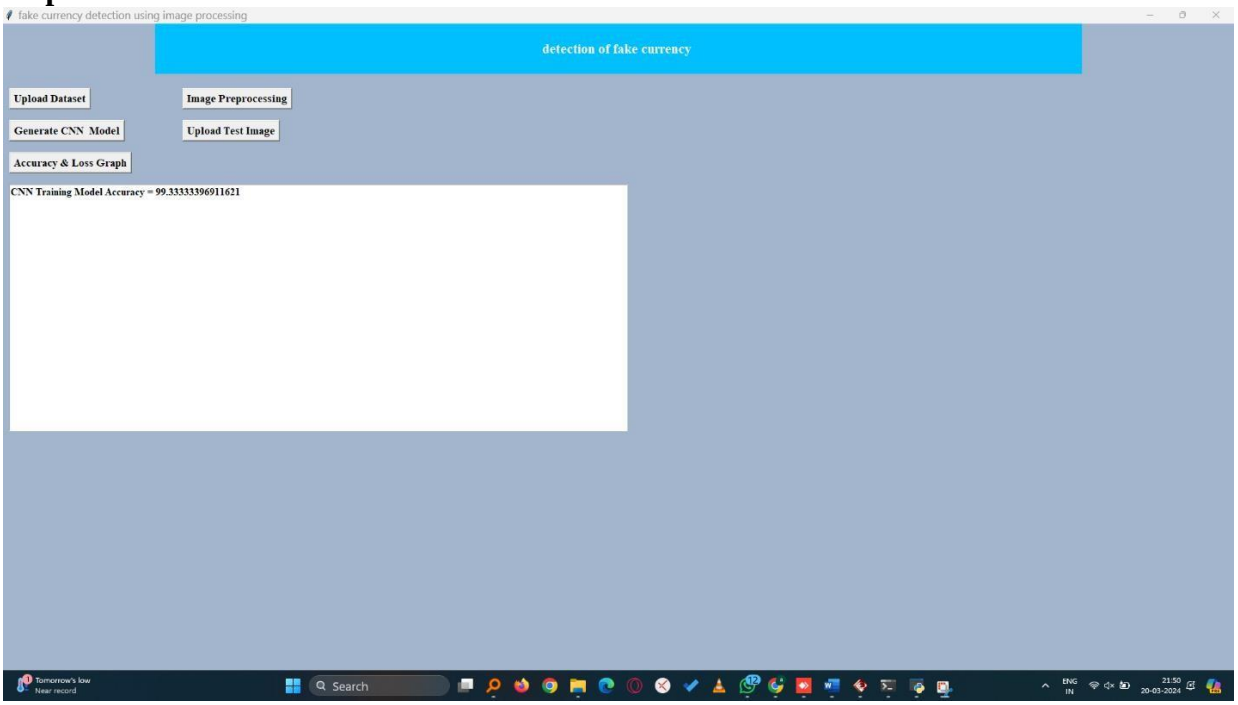
```
processButton = Button(main, text="Image Preprocessing", command=processImages)
processButton.place(x=280,y=100)
processButton.config(font=font1)

cnnButton = Button(main, text="Generate CNN  Model", command=generateModel)
cnnButton.place(x=10,y=150)
cnnButton.config(font=font1)

predictButton = Button(main, text="Upload Test Image", command=predict)
predictButton.place(x=280,y=150)
predictButton.config(font=font1)

graphButton = Button(main, text="Accuracy & Loss Graph", command=graph)
graphButton.place(x=10,y=200)
graphButton.config(font=font1)

font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=120)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=250)
text.config(font=font1)

main.config(bg='LightSteelBlue3')
main.mainloop()
```

5.4 Screen Shots:

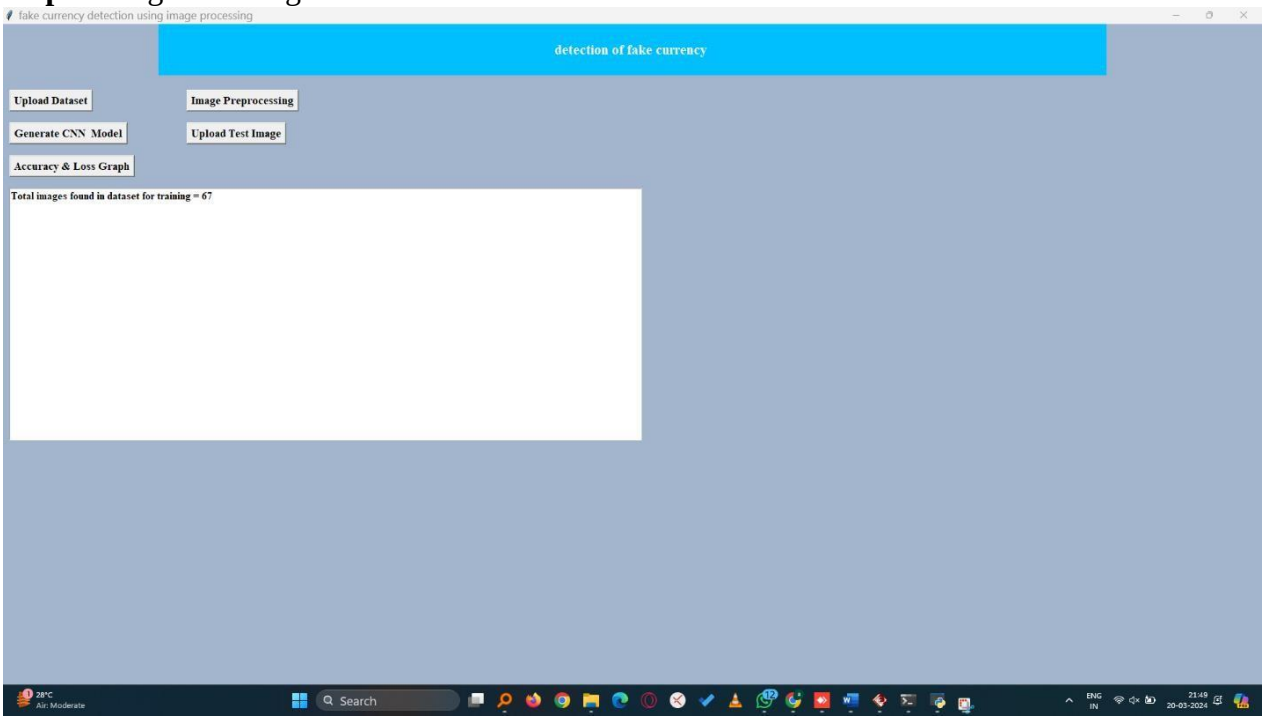
Step:1 This is to open the fake currency



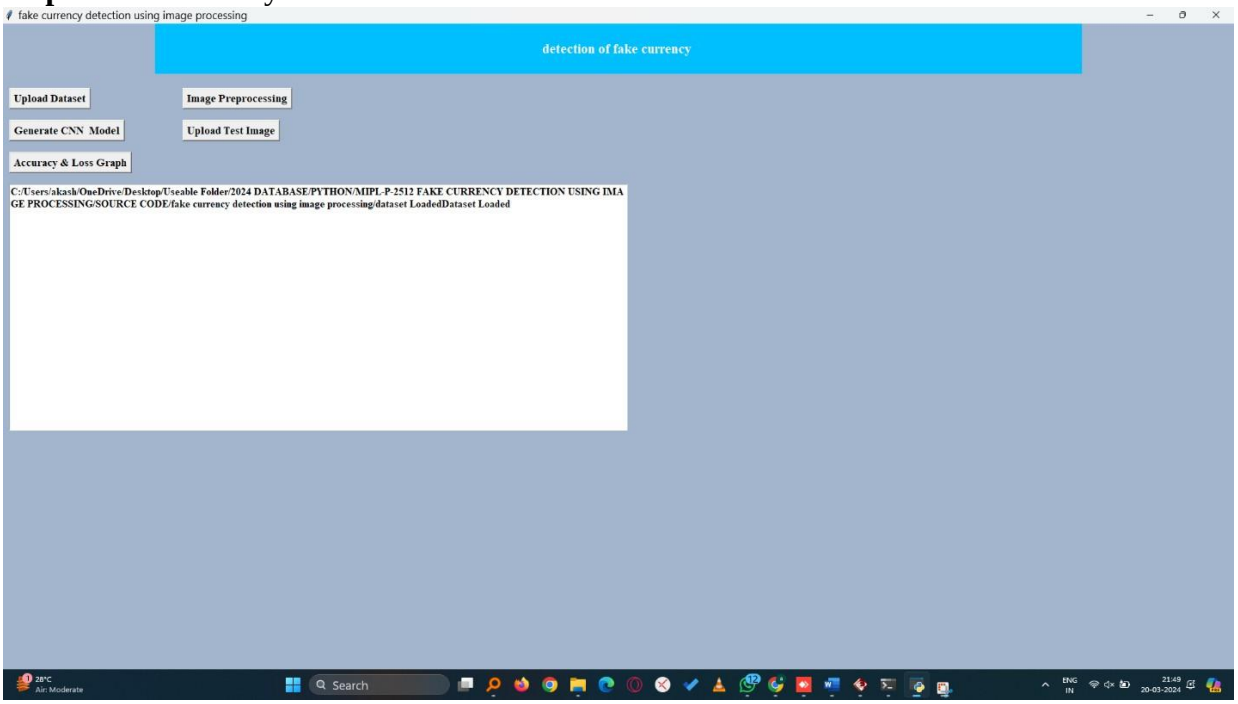
Step:2 This is CNN Model to dataset



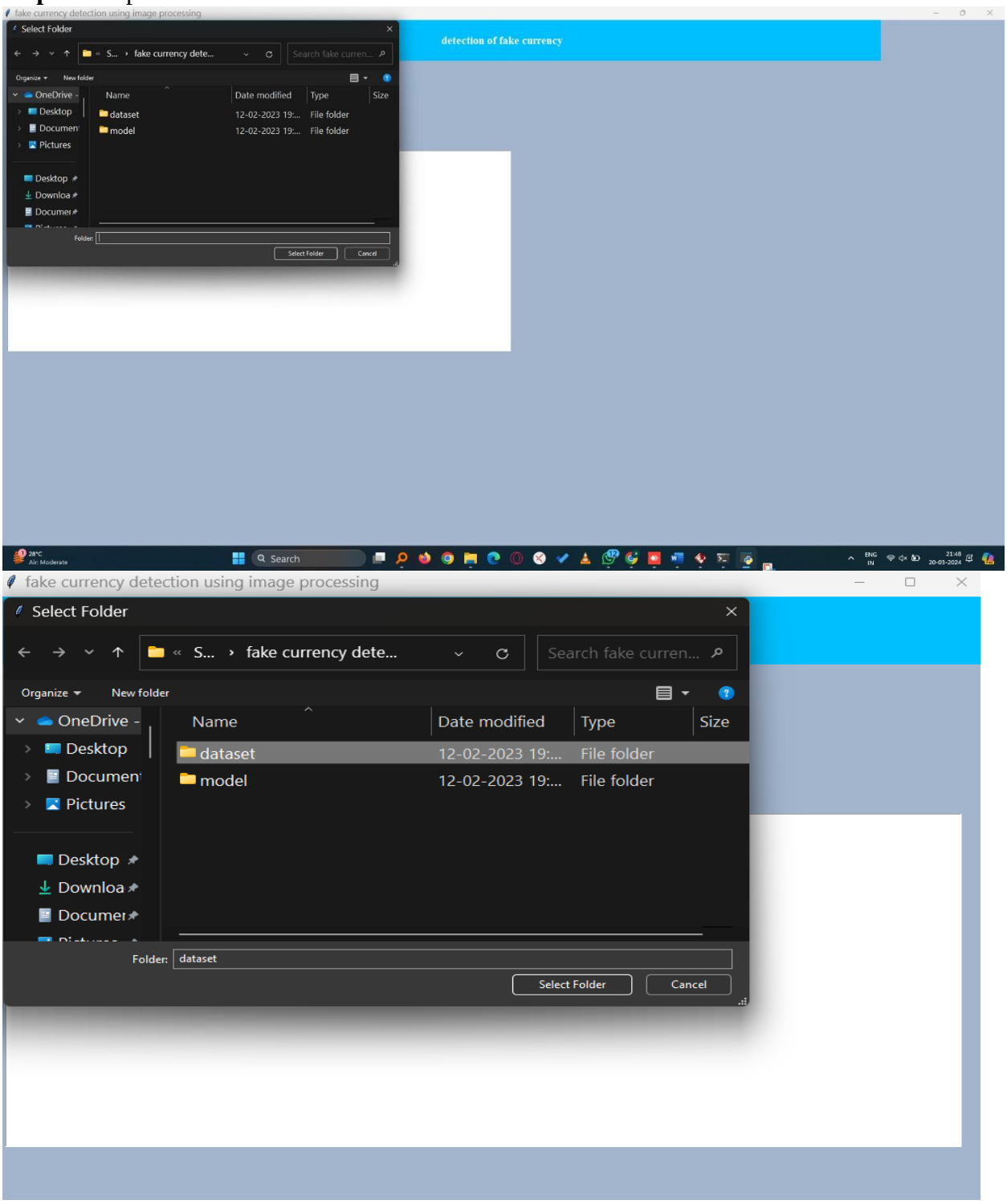
Step:4 Image founding



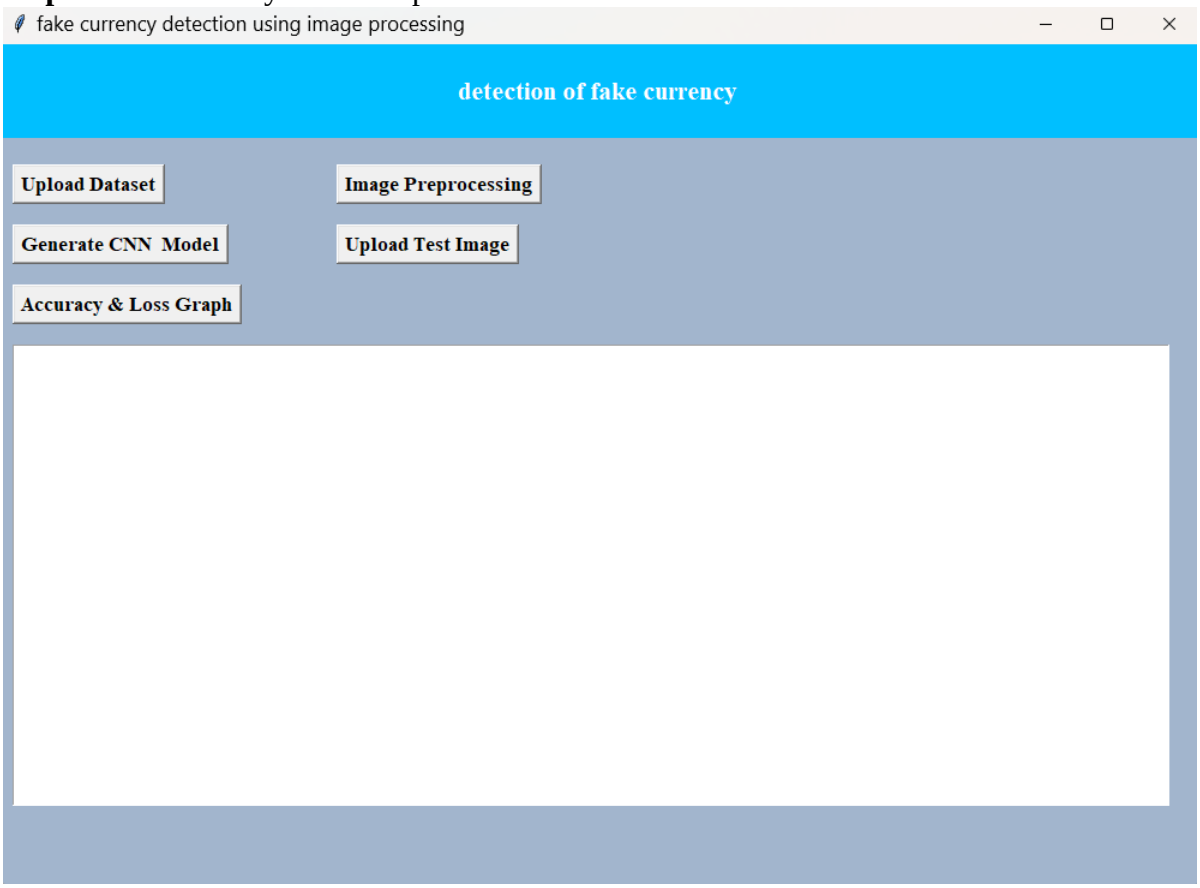
Step:5 Fake currency detection



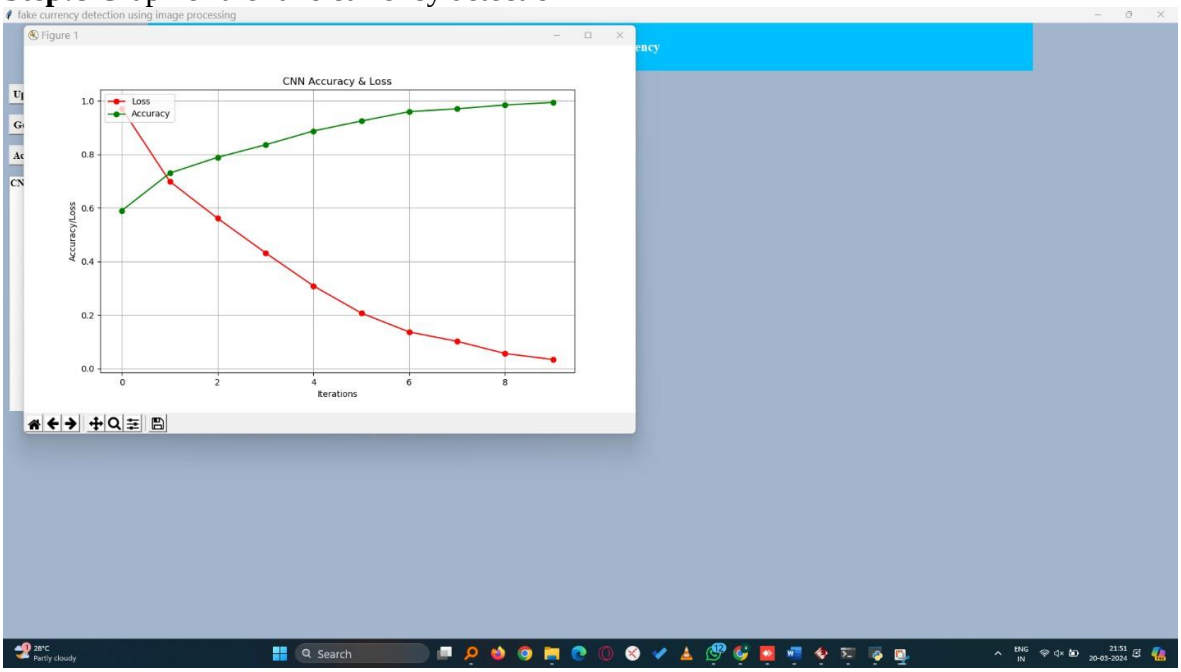
Step:6 To upload the dataset



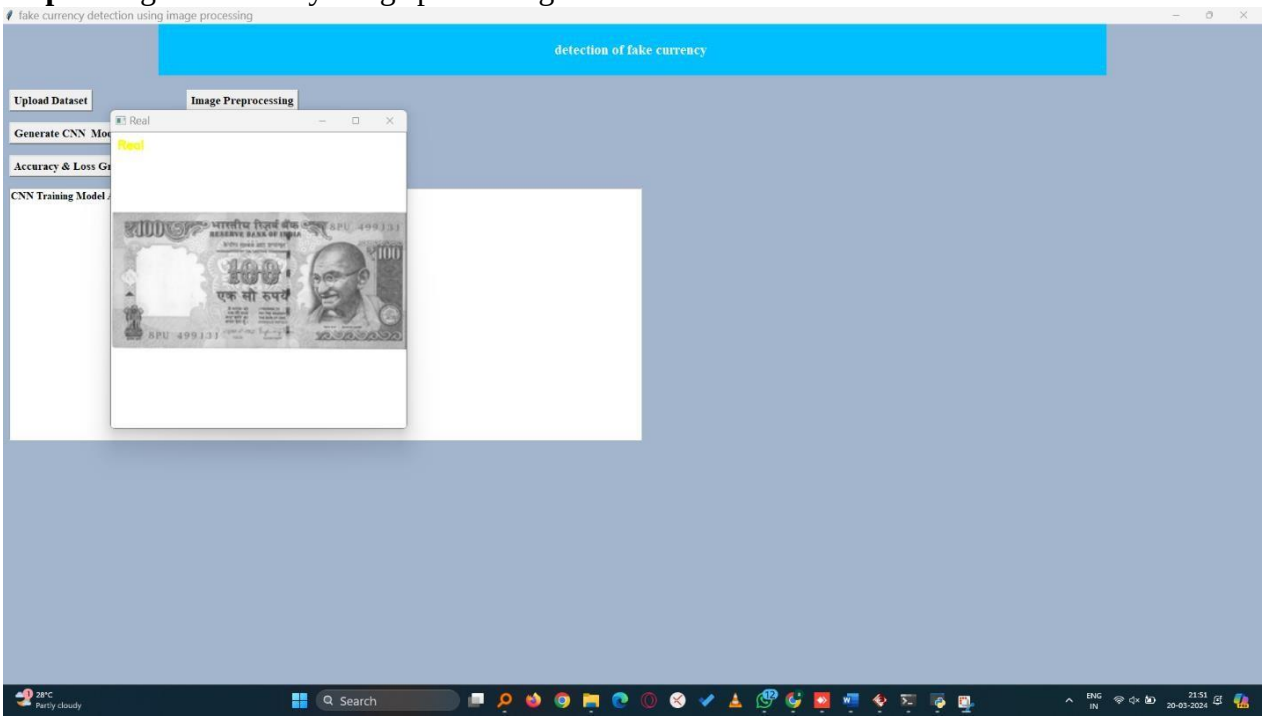
Step:7 Fake currency detection process



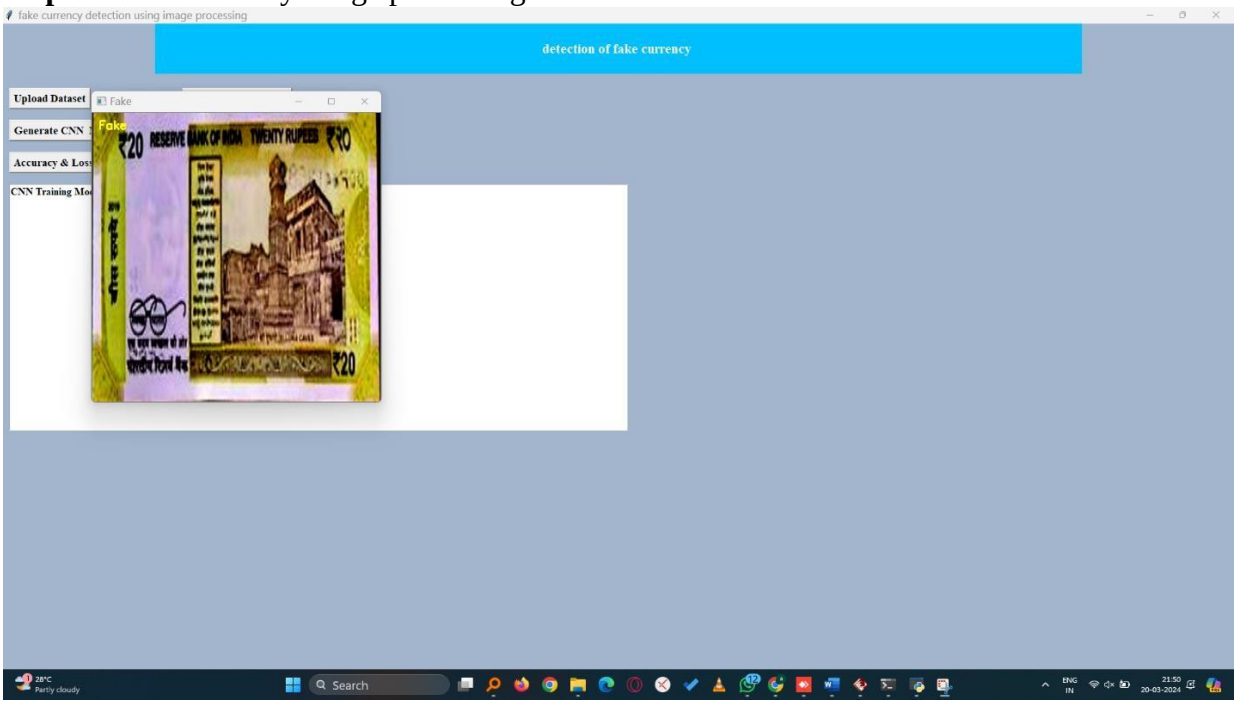
Step:8 Graph of the fake currency detection



Step:9 Original currency image processing



Step:10 Fake currency image processing



6. SYSTEM TESTING

6.1. Software Testing

Testing

- The process of executing a system with the intent of finding an error.
- Testing is defined as the process in which defects are indentified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.
- Quality is defined as justification of the requirements
- Defect is nothing but deviation from the requirements
- Defect is nothing but bug.
- Testing---The presence of bugs
- Testing can demonstrate the presence of bugs, but not their absence
- Debugging and Testing are not the same thing!
- Testing is a systematic attempt to break a program or the AUT
- Debugging is the art or method of uncovering why the script/program did not execute properly.

6.2 Testing Approaches

Testing can be done in two ways

- Bottom up approach
- Top down approach

Bottom up Approach

Testing can be performed starting from smallest and lowest level modules and proceeding ne at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested

attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

Top down Approach

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module. The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

6.3 Testing Methods

1. Black box or functional testing
2. White box testing or structural testing

Black Box Testing

This method is used when knowledge of the specified function that a product has been designed to perform is known. The concept of black box is used to represent a system whose inside workings are not available to inspection. In a black box the test item is a ”Black”, since its logic is unknown, all that is known is what goes in and what comes out, or the input and output.

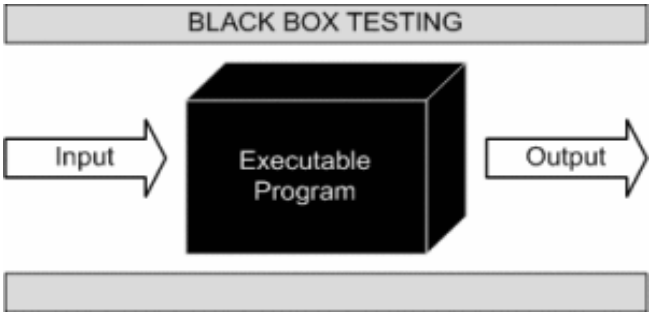
Black box testing attempts to find errors in the following categories:

a. Incorrect or missing functions
b. Interface errors
c. Errors in data structure
d. Performance errors
e. Initialization and termination errors

As shown in the following figure of Black box testing, we are not thinking of the internal workings, just we think about

What is the output to our system?

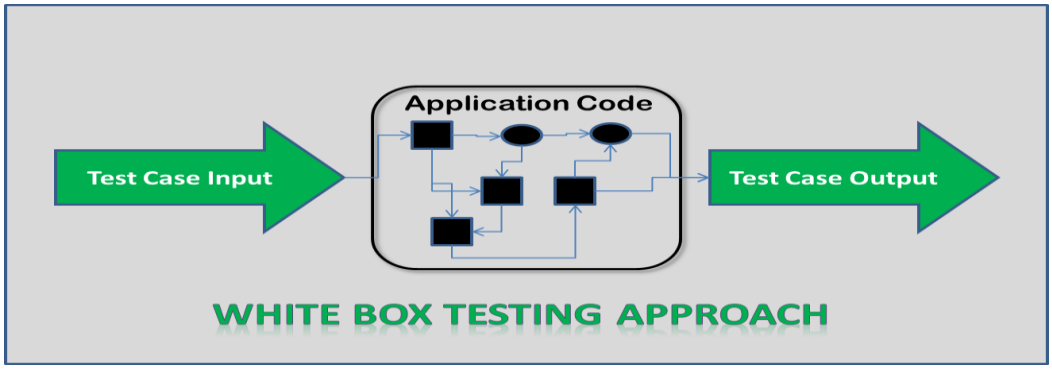
What is the output for given input to our system?



The Black box is an imaginary box that hides its internal workings

White Box Testing

White box testing is concerned with testing the implementation of the program. The intent of structural is not to exercise all the inputs or outputs but to exercise the different programming and data structure used in the program. Thus structural testing aims to achieve test cases that will force the desire coverage of different structures. Two types of path testing are statement testing coverage and branch testing coverage.



The white box testing strategy, the internal workings

6.4. Testing Types

Different levels of testing are used in the testing process, each level of testing aims to test different aspects of the system. The basic levels are:

- Unit testing
- Integration testing
- Functionality testing
- System testing
- User Acceptance testing

Unit Testing

Unit testing focuses on the building blocks of the software system, that is, objects and sub system. There are three motivations behind focusing on components. First, unit testing reduces the complexity of the overall tests activities, allowing us to focus on smaller units of the system. Second, unit testing makes it easier to pinpoint and correct faults given that few components are involved in this test. Third, unit testing allows parallelism in testing activities that is each component can be tested independently of one another. Hence the goal is to test the internal logic of module.

Integration Testing

In the integration testing, many test modules are combined into sub systems, which are then tested. The goal here is to see if the modules can be integrated properly, the emphasis being on testing module interaction.

After structural testing and functional testing we get error free modules. These modules are to be integrated to get the required results of the system. After checking a module, another module is tested and is integrated with the previous module. After the integration, the test cases are generated and the results are tested.

Functionality Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

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

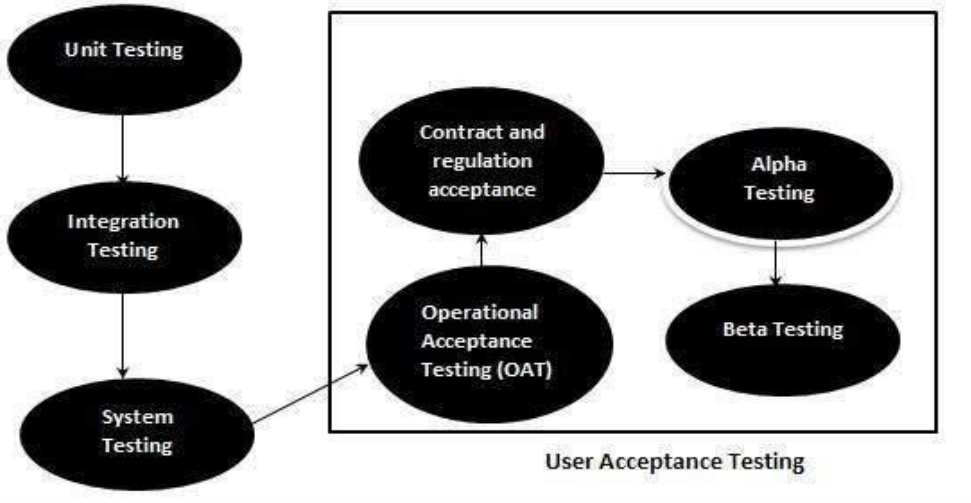
Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional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

System Testing

In system testing the entire software is tested. The reference document for this process is the requirement document and the goal is to see whether the software meets its requirements. The system was tested for various test cases with various inputs.

User Acceptance Testing

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactory. Testing here focus on the external behavior of the system, the internal logic of the program is not emphasized.
In acceptance testing the system is tested for various inputs.



Testing is the process of finding difference between the expected behavior specified by system models and the observed behavior of the implemented system.

6.5. Test Cases

A Test case is a set of input data and expected results that exercises a component with the purpose of causing failure and detecting faults. Test case is an explicit set of instructions designed to detect a particular class of defect in a software system, by bringing about a failure. A Test case can give rise to many tests.

Test cases can be divided in to two types. First one is positive test cases and second one is negative test cases. In positive test cases are conducted by the developer intention is to get the output in negative test cases are conducted by the developer intention is to don't get the output.

Writing Good Test Cases:

- As far as possible, write test cases in such a way that you test only one thing at a time. Do not overlap or complicate test cases. Attempt to make your test cases 'atomic'.

- Ensure that all positive scenarios and negative scenarios are covered.

Language:

- Write in simple and easy to understand language.
- Use active voice: Do this, Do that.
- Use exact and consistent names (of forms, fields, etc).

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary.

Test Cases

Test Case ID	Test Case Description	Testing Type	Steps to Execute	Expected Results	Actual Result	Result
UT01	Check for the upload dataset	Unit Testing	Check the code for upload dataset	Dataset should be uploaded	Dataset is uploaded.	Pass
UT02	Check for the Generate CNN model	Unit Testing	Check the code for Generate CNN model	CNN model should be Generated	CNN model Generated	Pass
IT01	Check for the image pre-processing	Integration Testing	Upload the images and click on the pre-processing	Image Pre-processing should be completed	Image pre-processing completed	Pass
FT01	Check for upload test image	Functionality Testing	Upload test image for fake currency	Fake currency should be identified	Fake currency is detected	Pass
ST01	Check for Accuracy of the graph	System Testing	Upload test image for accuracy	Accuracy graph should be displayed	Accuracy graph is displayed	Pass
UAT01	Check for the loss graph	User Acceptance Testing	Upload test image for loss graph	Loss graph should be displayed	Loss graph is displayed	Pass

7.CONCLUSION

In this project, a fake currency detection model has been proposed for authentication of Indian currency notes of de- nomination 500 and 2000 and implemented using OpenCV image processing library in Python3. In this model, 10 features of the input currency note are considered and then analyzed using 3 different algorithms. The input image is taken through a GUI which allows the user to browse the image in his/ her system. Then the results of the implemented model are computed and the analysis of each feature is displayed in detail through a graphical user interface (GUI) created using Tkinter GUI library.

The model takes less time (about 5 sec- when only final results are shown leaving unnecessary details) for processing an input image. The results are also quite decent giving almost 79% accuracy in detecting genuine currency and 83% accuracy in detecting counterfeit currency.

8.FUTURE ENHANCEMENT

1. Integration of Advanced Deep Learning Models Leverage CNNs and Transformer-based models for enhanced accuracy in currency recognition and counterfeit detection.

Multimodal Analysis Combine image processing with UV or IR imaging to detect hidden security features.

Enhanced Feature Extraction Techniques Use methods like SIFT and HOG for detailed feature analysis of currency notes.

Real-Time Processing and Mobile Integration Optimize algorithms for real-time processing on mobile devices for on-the-go currency verification.

Continuous Learning and Adaptation Implement online learning to regularly update the model with new counterfeit data and stay current with evolving counterfeiting techniques.

9.BIBLOGRAPHY

REFERENCES

[1] S. R. Darade and G. Gidveer, “Automatic recognition of fake indian currency note,” in 2016 international conference on Electrical Power and Energy Systems (ICEPES). IEEE, 2016, pp. 290–294.

[2] B. P. Yadav, C. Patil, R. Karhe, and P. Patil, “An automatic recognition of fake indian paper currency note using matlab,” Int. J. Eng. Sci. Innov. Technol, vol. 3, pp. 560–566, 2014.

[3] A. Zarin and J. Uddin, “A hybrid fake banknote detection model using ocr, face recognition and hough features,” in 2019 Cybersecurity and Cyberforensics Conference (CCC). IEEE, 2019, pp. 91–95.

[4] M. S. Veling, M. J. P. Sawal, M. S. A. Bandekar, M. T. C. Patil, and

[5] M. A. L. Sawant, “Fake indian currency recognition system by using matlab.”

[6] F. A. B, P. Mangayarkarasi, Akhilendu, A. A. S, and M. K, “Fake indian currency note recognition,” vol. 7, pp. 4766–4770, 2020. [Online]. Available: https://www.irjet.net/archives/V7/i5/IRJET-V7I5915.pdf

10.GANTT CHART

S.NO	TASK	Week 1,2 Apr 15- May 03	Week 3,4 May 04- May 17	Week 5 May 18-May 24	Week 6 May 25- May 31	Week 7 Jun 01 -Jun 09	Week 8 Jun 10 -Jun 16	Week 9,10 Jun 17- Jun 28	Week 11 Jun 29- Jul 05	Week 12 Jul 06- Jul 12	Week 13, 14 Jul 13- Jul 22	Week 15 Jul 23- Jul 27
1	**REQURIMENT ANALYSIS**											
	• Fact Finding											
	• Requirements specification											
2	**DESIGN**											
	• Class											
	• Use case											
	• Sequence											
	• Activity											
3	**CODING**											
	• Image Acquisition Module											
	• Preprocessing Module											
	• Feature Extraction Module											
	• Classification/Decision Making											
4	**TESTING**											
	• Unit Testing											
	• Integration Testing											
	• Functionality Testing											
	• System Testing											
	• User Acceptance Testing											
5	**DOCUMENTATION**											