

CARTOONIFY AN IMAGE (USING OPENCV AND PYTHON)

A Summer internship Report Submitted in partial fulfillment of the requirements for the
award of the degree of

**BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING-DATA SCIENCE**

Submitted by

Miss A. Sarika (21071A6770)

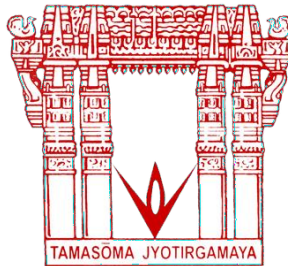
Miss G. Veda Sri (21071A6789)

Miss Hima Sameera. N (21071A6790)

Under the guidance of

R. Kranthi Kumar

(Professor, Department of CSE-(CYS, DS) and AI&DS)



DEPARTMENT OF CSE-(CYS, DS) and AI&DS

**VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF
ENGINEERING AND TECHNOLOGY**

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE,
EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi,
Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC
ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

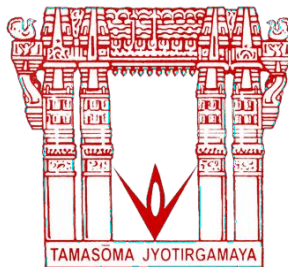
Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

**VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF
ENGINEERING AND TECHNOLOGY**

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE,
EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi,
Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC
ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF CSE-(CYS, DS) and AI&DS



CERTIFICATE

This is to certify that the project report entitled “ **Cartoonify an image (Using OpenCV and Python)** ” is a bonafide work done under our supervision and is being submitted by **Miss. A. Sarika (21071A6770), Miss. G. Veda Sri (21071A6789), Miss. Hima Sameera. N (21071A6790)** in partial fulfillment for the award of the degree of Bachelor of Technology in CSE-CYS, DS and AI&DS , of the VNRVJIET, Hyderabad during the academic year 2023-2024. Certified further that to the best of our knowledge the work presented in this thesis has not been submitted to any other University or Institute for the award of any Degree or Diploma.

R.Kranthi Kumar
Professor
Dept. of CSE-(CYS, DS) and AI&DS
VNR VJIET

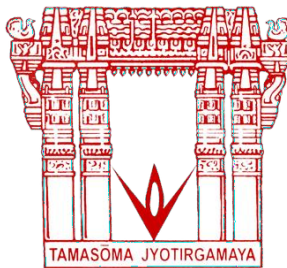
Dr. M. Raja Sekar
Professor and Head
Dept. of CSE-(CYS, DS) and AI&DS
VNR VJIET

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF CSE-(CYS, DS) and AI&DS



DECLARATION

We declare that the major project work entitled “**Cartoonify an image(Using OpenCV and Python)**” submitted in the department of **CSE-(CYS, DS) and AI&DS**, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **CSE-(CYS, DS) and AI&DS** is a bonafide record of our own work carried out under the supervision of **Dr.V.Prasanthi, Professor, Department of CSE-(CYS, DS) and AI&DS, VNRVJIET**. Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

Place: Hyderabad

A.Sarika
(21071A6770)

G.Vedasri
(21071A6789)

Hima Sameera.N
(21071A6792)

ACKNOWLEDGEMENT

Firstly, we would like to express our immense gratitude towards our institution VNR Vignana Jyothi Institute of Engineering and Technology, which created a great platform to attain profound technical skills in the field of Computer Science, thereby fulfilling our most cherished goal.

We are very much thankful to our Principal, **Dr.Challa Dhanunjaya Naidu**, and our Head of Department, **Dr.M.Raja Sekar**, for extending their cooperation in doing this project within the stipulated time.

We extend our heartfelt thanks to our guide , **Mr. R. Kranthi Kumar**, and the project coordinators **Dr.Vempaty Prashanthi and Mrs.B.Deepika** for their enthusiastic guidance throughout the course of our project.

Last but not least, our appreciable obligation also goes to all the staff members of the Computer Science & Engineering department and to our fellow classmates who directly or indirectly helped us.

Miss.A.Sarika (21071A6770)

Miss.G.Vedasri (21071A6789)

Miss.Hima Sameera.N (21071A6790)

ABSTRACT

The “Cartoonify an image (Using OpenCV and Python)” is a captivating application that encourages enthusiasts to unleash their creativity and try different versions of an image. This project aims to develop a python application that transforms ordinary images into cartoon like representations or cartoon like versions. This project leverages computer vision and image processing techniques to create a visually appealing and stylized output resembling traditional hand-drawn cartoons. The primary focus is on making the process accessible to users with minimal programming experience. Cartoonify an image is converting an image of any form to a cartoon version of it. This can be achieved by many ways and one of it is using opencv in python. OpenCV- Open Computer Vision is a library of programming functions mainly aimed at real-time computer vision, written in C/C++, and a cross-platform library. There are bindings in Python, Java, JavaScript, and MATLAB/OCTAVE. It is majorly used in image transformation, object detection, face recognition, and many other stunning applications. Computer Vision as you know is a very powerful tool with immense possibilities. The outcome of this project is to create a perfect code to convert an original image into its cartoonized version through tools by efficient down and up scaling of the given images. The platform's architecture integrates secure user authentication, sophisticated search functionalities, and collaborative tools to foster an interactive environment.

INDEX

1. Introduction	1
2. Literature Survey/ Existing System	2
2.1 Feasibility Study	2
2.1.1 Organizational Feasibility	2
2.1.2 Economic Feasibility	2
2.1.3 Technical Feasibility	2
2.1.4 Behavioral Feasibility	3
2.2 Literature Review	4
2.3 Existing System	5
2.4 Drawbacks Of the Existing System	5
3. Software Requirement Analysis	6
3.1 Introduction	6
3.1.1 Document Purpose	6
3.1.2 Definitions	6
3.2 System Architecture	7
3.3 Functional Requirements	8
3.4 System Analysis	8
3.5 Non-Functional Requirements	10
3.6 Software Requirement Specification	11
3.7 Software Requirements	11
3.8 Hardware Requirements	11
4. Software Design	12
4.1 UML Diagrams	12
4.1.1 Use Case Diagram	13
4.1.2 Sequence Diagram	17
4.1.3 Activity Diagram	21
4.1.4 Class Diagram	23
5. Proposed System	24
5.1 Methodology	24
5.2 Functionalities	25
5.3 Advantages Of Proposed System	25
6. Coding/Implementation	26
6.1 create a tkinter window	26
6.2 define functions	27
6.3 save images & update progress function	28
6.4 main loop	28

7. Testing	29
7.1 Types Of Testing	29
7.1.1 Manual Testing	29
7.1.2 Automated Testing	29
7.2 Testing Levels	30
7.2.1 Non-Functional Testing	30
7.2.1.1 Performance Testing	30
7.2.1.2 Stress Testing	30
7.2.1.3 Security Testing	30
7.2.1.4 Portability Testing	30
7.2.1.5 Usability Testing	31
7.2.1.6 Functional Testing	31
7.3 Test Cases	31
8. Results	32
9. Conclusion And Further Work	35
10. References	36

List of Tables

Table	Page No
Table. 7.4.1 Test cases	31

1. INTRODUCTION

The main aim of the project is to detect objects or convert a real-life image into cartoon effect or a cartoon effect to real image. The cartoon is the most popular, famous and entertaining art. Image to Image conversion is a task to establish a visual mapping between output and Input images. The Conversion of real-world images into cartoons with some tools, soft wares and materials of some products is known as image cartooned. Cartoons are many times in the form of 2D or 3D art formats. The research of conversion of image to cartoon consists of identifying objects in images, the number of objects, the number of dimensions, the image to blur effects are appreciated in media and communication. Each image is viewed in 2D matrix. To obtain a cartoon image as the same in real image it needs to obtain every line along with each shade and color in an image. Several methods have been used on the basis of Convolutional Neural Network (CNN). The framework which is utilize to single trained model to multiple cartoon styles are “Cartoon Renderer”. Turning various photos into its cartoon effect such problem studied in this paper. Cartoons are artistic forms used in day-to-day life. Like other forms of arts, many arts created cartoon effects using real world image. Our method takes a set of photos and a set of images for training data. Our method is also much more efficient to train than the existing model. Advanced technology has now part of our life. The real images processing appears in many real-life applications i.e., home security, banking system, education sector and railway. The basic concept of this algorithm is to convert RGB into its accurate, cartoon image with multiple filtrations or blurred image with proper edge detection.

2. LITERATURE SURVEY/ EXISTING SYSTEM

2.1 FEASIBILITY STUDY

The feasibility study for applying a cartoon effect to images using OpenCV in Python involves a targeted exploration of technical requirements and potential challenges. The objective is to enhance images with cartoon-like features, including pronounced edges and simplified colors. The study assesses existing solutions in Python libraries and OpenCV functions, emphasizing insights from online resources. Key considerations include image preprocessing techniques, algorithm selection, and user interaction. A prototype is implemented, tested for consistency and quality, and evaluated for performance. Documentation captures algorithms, parameters, and usage instructions. The study concludes with a summary, recommending the feasibility of integrating the cartoon effect using OpenCV in Python.

2.1.1 ORGANIZATIONAL FEASIBILITY

The organizational feasibility of implementing a cartoon effect using OpenCV in Python involves evaluating the internal readiness, technical expertise, and resource availability within the organization. This includes assessing the team's proficiency in Python and OpenCV, ensuring adequate computing resources, and evaluating budget considerations for potential costs. Effective communication channels and collaboration structures are crucial to support successful project implementation.

2.1.2 ECONOMIC FEASIBILITY

The project is financially feasible because it can be used without any charges. Since the model will be trained before using it, you need an internet connection to use the project model. The software used for building the project is economically feasible too. The application is built using OpenCV and python, which is handy, free and easy to use.

2.1.3 TECHNICAL FEASIBILITY

The technical feasibility of applying a cartoon effect to images using OpenCV in Python involves evaluating the practicality of implementing the required image processing techniques. This includes assessing the compatibility of OpenCV with the project's objectives, ensuring the availability of necessary algorithms, and confirming that the team possesses the technical skills to execute the task. A brief analysis will determine whether the technical aspects align with project goals and can be effectively implemented

2.1.4 BEHAVIORAL FEASIBILITY

Behavioral feasibility for implementing a cartoon effect using OpenCV in Python involves evaluating the willingness and adaptability of end-users to embrace the new image processing feature. It focuses on assessing how users perceive and interact with the cartoon effect, ensuring that it aligns with their preferences and usability expectations. Understanding user behavior and preferences are crucial for the successful adoption of the cartoon effect in real-world applications.

2.2 LITERATURE REVIEW

- **Cartoonify an Image using Open CV in Python[1]:**

The main objective is to put ahead a result for making over pictures or vids of realglobal into animated prints (Cartoon Images) or Videotape. The concept of the paper is grounded on unique shots and vids which are converted to an art shape similar as oil. Amongst all the methods usable, the operation of a Generative Adversarial Network (GAN) known as Cartoon GAN may be used for the styling actual-world pictures that use two loss functions videlicet, glad loss and inimical loss for getting a sharp and clear image. This project would follow an OpenCV library within the python whilst making the layout. Python is the collections of libraries and it has multiplex libraries for real- world operations. One corresponding library is OpenCV and it is a move-platform library used for Computer Vision.

- **Image Transformation into Cartoon Using OpenCV[2]:**

The objective of this paper is to give a solution regarding transformation of images into cartoon images. The previous methods of image transformation require convoluted computer graphics and programming skills. An image transformation system that can produce a adapted cartoon face from the input image is presented in this paper. This proposed system is easy to use and requires less programming skills with little user interaction. The basic concept of this paper is particularly on nominated images that are changed to cartoons. This focuses on using OpenCV to transform input image to cartoon image, which will be used for more image processing systems for various applications.

- **Animating Static Pictures: A Cartoon Transformation Approach with OpenCV and Python[3]:**

Image processing is a versatile methodology employed for enhancing images, extracting valuable information, and creating new representations. This process involves a range of tools, including OpenCV, Scikit Image, and NumPy, which play pivotal roles in performing various transformations on images. A significant aspect of our approach employs Generative Adversarial Networks (GANs) to learn and animate these images. Our primary objective is to enhance the versatility and controllability of our framework. Generative modeling, a fundamental aspect of this research, falls under the domain of unsupervised machine learning, wherein the model autonomously identifies regularities and patterns in the input data.

- **An Effective Cartoonifying of an Image using Machine Learning[4]:**

Cartoonifying an image is the process of transforming a regular photograph into a cartoon-style image. This research paper proposes a method to cartoonify images using OpenCV, a popular open-source computer vision library using Python. The proposed method involves several steps, including edge detection, color quantization, and image smoothing. The edge detection step is used to extract edges from the input image. Then, in the color quantization step, the image palette is reduced to a fixed number of colors using the k-means clustering algorithm. Finally, the image is smoothed using a bilateral filter to create a cartoon-like effect. The proposed method is evaluated on several images, and the results show that the proposed method produces high-quality cartoon images with reduced noise and better visual appeal compared to existing methods.

2.3 EXISTING SYSTEM

The existing system Cartoonifying an image gives users the chance to create cartoonised form of an image and to view it in a different version. It's consisting of OpenCv, Tkinter, imageio, tqdm and other libraries and python modules. It is used to get a different version of the image. We can get desired outputs as we can change the values in the code conveniently. These cartoonised images are used in magazines and newspaper articles.

2.4 DRAWBACKS OF THE EXISTING SYSTEM

- Code should be accurate or there will be efficiency problems. Time may not be efficient in these cases.
- Scalability is not provided.

3. SOFTWARE REQUIREMENT ANALYSIS

3.1 INTRODUCTION

Following elicitation, requirement analysis is an important and necessary process. To create uniform and unambiguous requirements, we examine, improve, and scrutinize the obtained requirements. This exercise goes over all the requirements and may show a graphical representation of the full system.

3.1.1 DOCUMENT PURPOSE

The purpose of this Python script utilizing OpenCV is to cartoonify an input image. It achieves this by converting the image to grayscale, applying a bilateral filter for smoothing, generating an edge mask using adaptive thresholding, and combining the edges with the original image to produce a cartoon effect. The script provides a quick and effective way to enhance visual aesthetics in images.

3.1.2 DEFINITIONS

1. Cartoonify:

- The process of transforming a digital image into a stylized representation resembling a cartoon. In the context of OpenCV and Python, this involves applying various image processing techniques to enhance edges and simplify color gradients.

2. OpenCV (Open Source Computer Vision Library):

- An open-source computer vision and machine learning library. OpenCV provides a wide range of tools and functions for image processing, computer vision, and machine learning tasks.

3. Python:

- A high-level, versatile programming language commonly used for scripting, software development, and data analysis. Python is the scripting language of choice in this context.

4. Grayscale Conversion:

- The process of converting a color image into a grayscale image, where each pixel is represented by a single intensity value. This simplification is often the first step in image processing.
- A technique in image processing where the threshold for pixel intensity is dynamically adjusted based on the local characteristics of the image. It is commonly used to create an edge mask that highlights significant features.

5. Edge Mask:

- A binary image where pixels represent edges in the original image. In cartoonification, the edge mask is generated through adaptive thresholding and serves as a guide for combining edges with the original image.

.

3.2 SYSTEM ARCHITECTURE

The system architecture for applying a cartoon effect to images using OpenCV in Python involves a straightforward yet effective design. The process typically begins with image input, where the chosen image undergoes preprocessing steps such as blurring and edge detection. The core of the architecture lies in the application of specific OpenCV algorithms for cartoonization enhancements, transforming the input image into a cartoon-like representation. The final step in these algorithms, often a combination of filters, is to display or save the cartoonized image. The architecture leverages OpenCV's image processing capabilities, and the Python script serves as the orchestrator, connecting the preprocessing and cartoonization steps to achieve the desired effect. .

3.3 FUNCTIONAL REQUIREMENTS

1. **Upload:** Users can upload images.
2. **Cartoonify:** Apply processing steps for cartoonification.
3. **GUI:** User-friendly interface with upload and save buttons.
4. **Progress Bar:** Show processing progress.
5. **Display:** View original and cartoonified images.
6. **Custom Save:** Option to set custom filenames.
7. **Error Handling:** Graceful error feedback.
8. **Save Images:** Save cartoonified images.
9. **Feedback:** Labels for user greeting and feedback.
10. **Usability:** User-friendly design.

3.4 SYSTEM ANALYSIS

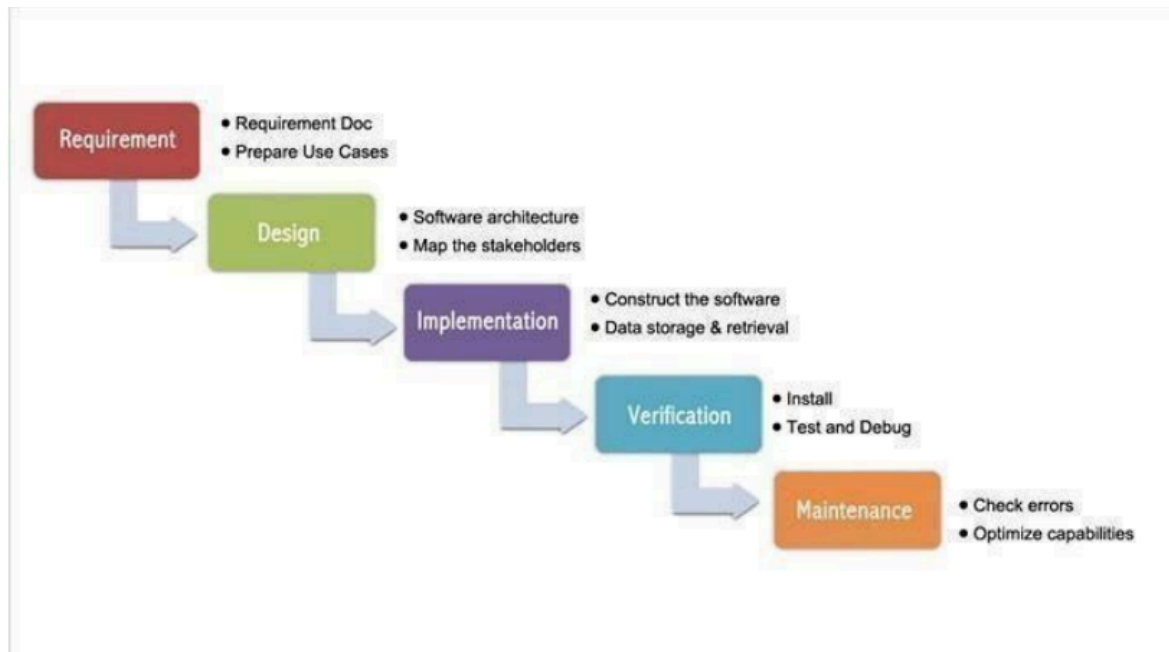


Fig.3.4.1 System Analysis of the proposed system

System analysis for implementing a cartoon effect on images using OpenCV in Python involves a comprehensive review of requirements and functionalities. It begins with a detailed examination of image processing needs, identifying key steps like blurring and edge detection. Algorithmic choices, particularly those within OpenCV, are scrutinized for their effectiveness in achieving the cartoon effect. The analysis also considers user interaction aspects, ensuring a user-friendly experience with adjustable parameters. Compatibility, performance optimization, and scalability are critical factors addressed to create a robust system. This systematic analysis lays the groundwork for a streamlined implementation, combining technical efficiency with user satisfaction.

Requirements: The search has become more intense and concentrated on the software's requirements at this time. To comprehend the nature of the programs to be developed, the software engineer must

first comprehend the software's information domain, which includes the required functionalities, user interface, and so on. The customer must be informed about the second activity, which must be recorded and presented

Design: This step is used to transform the above criteria as a representation in the form of "blueprint" software before coding begins. The design must be able to meet the criteria laid out in the previous stage.

Implementation: The design was converted into a machine-readable format in order for it to be interpreted by a computer in some circumstances, i.e., through the coding process into a programming language. This was the stage in which the programmer will put the technical design phase into action.

Verification: It, like anything else constructed, must first be put to the test. The same may be said for software. To ensure that the application is error-free, all functions must be checked, and the results must closely comply to the previously specified requirements.

Maintenance: Software maintenance, including development, is essential since the software that is being generated is not always exactly like that. It may still have minor faults that were not identified previously when it runs, or it may require additional capabilities that were not previously available in the software.

Useful factors: : The Extreme prototype model has its advantages as it is simple to use. Additionally, while using the model all the system requirements can be added any time explicitly and at the start the product can run without many issues. It is economic to add changes to the project when there are problems with system requirements.

3.5 NON-FUNCTIONAL REQUIREMENTS

- **Performance** - Specify the maximum acceptable time for the platform to respond to user actions.
- **Reliability**- Determine the required uptime percentage for the platform.
- **Security**- Specify the encryption standards for data transmission and storage. Define user access controls and authentication mechanisms.
- **Usability**- Ensure a consistent and intuitive user interface across different devices. Ensure the platform is accessible to users with disabilities.

SOFTWARE REQUIREMENT SPECIFICATION

The software requirements for implementing a cartoon effect using OpenCV in Python include functions for image input, preprocessing (blurring, edge detection), application of OpenCV algorithms for cartoonization, user interaction with adjustable parameters, and output display or storage. Non-functional requirements emphasize real-time processing, compatibility with various image formats and sizes, error handling, and documentation. The user interface should be intuitive, with optimal performance, security considerations, and testing requirements ensuring a robust and scalable system. Legal and ethical aspects, system constraints, and maintenance considerations are also addressed for a comprehensive development framework.

3.6 SOFTWARE REQUIREMENTS

- · Software : OpenCV, Image file, Code editor.
- · Operating System : Windows, mac , Linux
- · Technology : Python

3.7 HARDWARE REQUIREMENTS

- · Minimum 8GB Ram Computer
- · Internet Connection
- · Adequate Storage

4. SOFTWARE DESIGN

4.1 UML DIAGRAMS

The Device Architecture Manual describes the application requirements, operating state, application and subsystem functionality, documents and repository setup, input locations, yield types, human-machine interfaces, management reasoning, and external interfaces. The Unified Modeling Language (UML) assists software developers in expressing an analysis model through documents that contain a plethora of syntactic and semantic instructions. A UML context is defined as five distinct viewpoints that present the system from a particularly different point of view. The components are like modules that can be combined in a variety of ways to create a complete UML diagram. As a result, comprehension of the various diagrams is essential for utilizing the knowledge in real-world systems. The best method to understand any complex system is to draw diagrams or images of it. These designs have a bigger influence on our understanding. Looking around, we can see that infographics are not a new concept, but they are frequently utilized in a variety of businesses in various ways.

User Model View

The perspective refers to the system from the clients' point of view. The exam's depiction depicts a situation of utilization from the perspective of end-clients. The user view provides a window into the system from the perspective of the user, with the system's operation defined considering the user and what the user wants from it.

Structural model view

This layout represents the details and functionality of the device. This software design maps out the static structures. This view includes activity diagrams, sequence diagrams and state machine diagrams

Behavioral Model View

It refers to the social dynamics as framework components, delineating the assortment cooperation between various auxiliary components depicted in the client model and basic model view. UML Behavioral Diagrams illustrate time-dependent aspects of a system and communicate the system's dynamics and how they interact. Behavioral diagrams include interaction diagrams, use case diagrams, activity diagrams and state-chart diagrams.

Implementation Model View

The essential and actions as frame pieces are discussed in this when they are to be manufactured. This is also referred to as the implementation view. It uses the UML Component diagram to describe system components. One of the UML diagrams used to illustrate the development view is the Package diagram.

Environmental Model View

The systemic and functional component of the world where the program is to be introduced was expressed within this. The diagram in the environmental view explains the software model's after-deployment behavior. This diagram typically explains user interactions and the effects of software on the system. The following diagrams are included in the environmental model: Diagram of deployment.

The UML model is made up of two separate domains:

- Demonstration of UML Analysis, with a focus on the client model and auxiliary model perspectives on the framework.
- UML configuration presenting, which focuses on demonstrations, usage, and natural model perspectives.

4.1.1 USE CASE DIAGRAM

The objective of a use case diagram is to portray the dynamic nature of a system. However, because the aim of the other four pictures is the same, this description is too broad to characterize the purpose. We'll look into a specific purpose that distinguishes it from the other four diagrams. The needs of a system, including various factors, are collected using use case diagrams. The majority of these specifications are design specifications. As a result, use cases are constructed and actors are identified when examining a system to gather its functions. Use case diagrams are made to represent the outside view once the primary task is completed.

In conclusion, use case diagrams are useful for the following purposes:

- Used to collect a system's requirements.
- Used to get a bird's-eye view of a system.
- Determine various factors that are influencing the system.
- Display the interaction of the requirements as actors.

Use case diagrams are used to analyze a system's high-level requirements. The functionality of a system is recorded in use cases when the requirements are examined. Use cases can be defined as "system functionalities written in a logical order." The actors are the second pillar of use cases that is important. Any entities that interact with the system are referred to as actors.

Internal applications, human users and external applications can all be actors.

The following factors should be kept in mind when constructing a use case diagram.

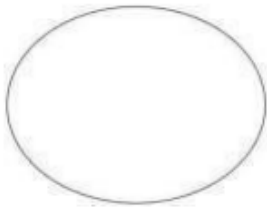
- As a use case, functionalities will be represented.
- Actors.
- Relationships between use cases and actors.

Use Cases

A use case is a written depiction of how visitors will execute tasks on your website. From the standpoint of a user, it defines how a system responds to a request. Each use case is characterized by a sequence of basic actions that start with the user's goal and finish when that goal is achieved.

Graphical Representation

Use cases are represented by an oval shape.



The following is a more precise analysis of a use case:

- A pattern of behavior displayed by the system.
- A series of related transactions performed by an actor as well as the system.
- Delivering something useful to the actor.

You can utilize use cases to document system requirements, connect with top users and domain experts, and test the system. Looking at the actors and defining what they can accomplish with the system is the greatest way to uncover use cases.

Flow of events

A sequence of times can be thought of as a collection of interactions (or opportunities) carried out by the system. They provide daily point-by-point details, published in terms of what the framework can do rather than whether the framework performs the task.

- When and how the employment case begins and ends.
- Interactions between the use case and the actor.
- Information required by the employment case.
- The employment case's normal sequence of events.
- A different or exceptional flow.

Construction of Use case

The behavior of the framework is graphically illustrated in use-case outlines. These graphs show how the framework is utilized at a high level, when seen through the perspective of an untouchable (actor). A utilization case graph can depict all or some of a framework's work instances. A use-case diagram may include the following elements:

- Actors.
- Use cases.

Relationships in use cases

Active relationships, also known as behavioral relationships, are a type of interaction that is frequently shown in use case diagrams. The four main types of behavioral relationships are inclusion, communication, generalization and extension.

1) Communicates

The behavioral relationship communicates connects an actor to a use case. Remember that the purpose of the use case is to provide some sort of benefit to the system's actor. As a result, it is critical to document these interactions between actors and use cases. A line with no arrow heads connects an actor to a use case.

2) Includes

The includes relationship (also known as the uses relationship) describes a situation in which a use case contains behavior that is shared by multiple use cases. To put it another way, the common use case is included in the other use cases. The included relationship is indicated by a dotted arrow pointing to the common use case.

3) Extends

The extended connection describes when one use case contains behavior that allows a new use case to handle a variant or exception to the basic use case. A distinct use case handles exceptions to the basic use case. The arrow connects the basic and extended use cases.

4) Generalizes

The generalized relationship indicates that one thing is more prevalent than another. This link could be between two actors or between two use cases. The arrow points to a "thing" in UML that is more general than another "thing."

In this system Use Case diagram:

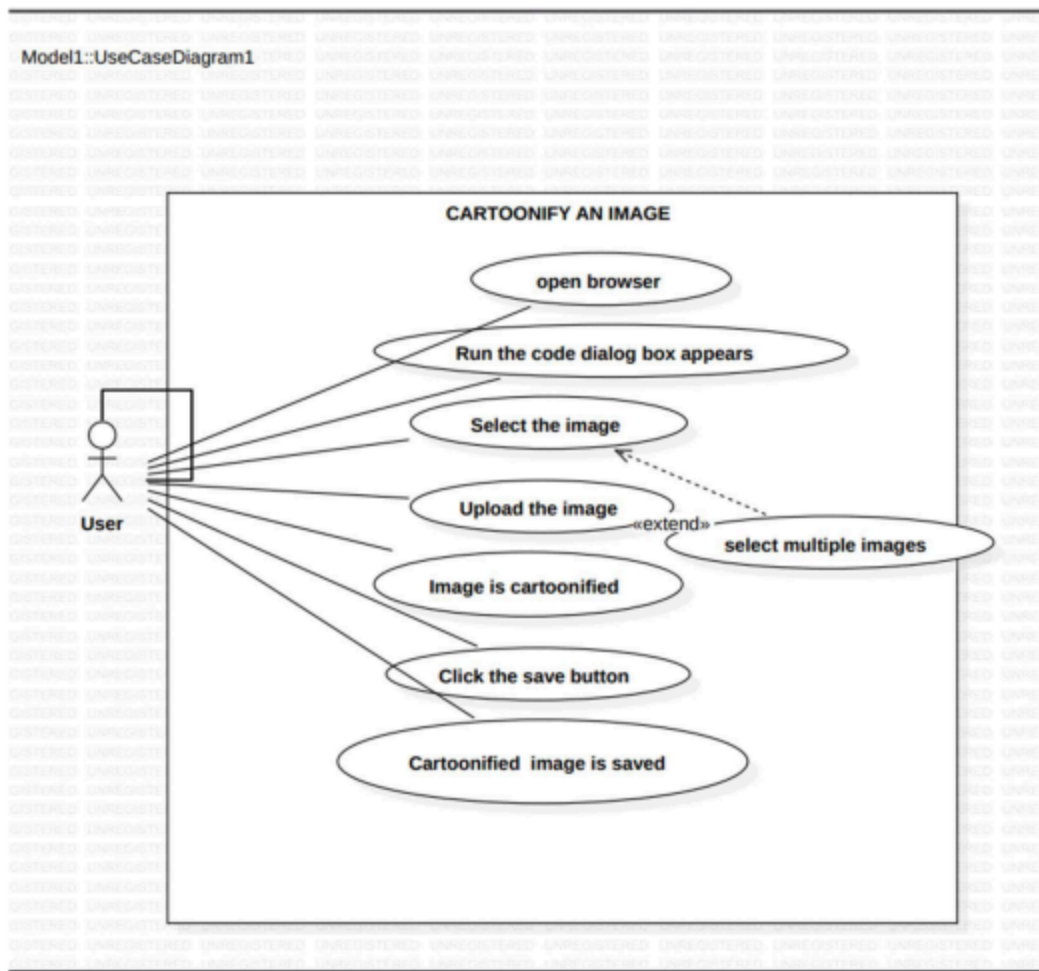


Fig4.1.1.1: Use case diagram for the application.

Actors:

- User

Use Cases:

- Open browser
- Run code
- Select image
- Upload image
- Image is cartoonified
- Save button
- Cartoonized image is saved

Connections:

- The user must browse the application.
- The user must select and upload the image.
- Multiple images can also be uploaded.
- Image is cartoonified and User saves the cartoonized image.

4.1.2 SEQUENCE DIAGRAM

Because it illustrates how a group of items interact with one another, a sequence diagram is a form of interaction diagram. These diagrams are used by software engineers and business people to comprehend the requirements for a new system or to document a current process. Sequence diagrams are sometimes known as event diagrams or event scenarios. Sequence diagrams can be useful as a reference for businesses and other organizations. Make the diagram to show:

- Describe the specifics of a UML use case.
- Create a model of the logic of a complex procedure, function, or operation.
- Examine how objects and components interact with one another in order to complete a process.
- Plan and comprehend the specific functionality of a current or future scenario.

Method logic: A UML sequence diagram can be used to study the logic of any function, method, or complex process, just as it can be used to examine the rationale of a use case. If you view a service to be a high-level method utilized by several customers, a sequence diagram is a fantastic approach to map out service logic.

Object: An object has a state, a lead, and a personality. The structure and direction of objects that are, for all intents and purposes, indistinguishable are depicted in their fundamental class. Each object in a diagram represents a specific instance of a class. An order case is an object that is not named.

Message: A message is the exchange of information between two articles that causes an event to occur. A message transmits information from the source point of control convergence to the objective

point of control convergence.

Link: An existing association between two objects, including class, implies that there is an association between their opposing classes. If an object associates with itself, use the image's hover adjustment.

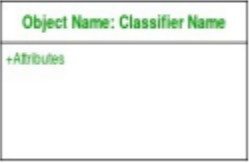
Lifeline: It reflects the passage of time as it goes downward. The events that occur consecutively to an object during the monitored process are depicted by this dashed vertical line. A designated rectangle shape or an actor symbol could be the starting point for a lifeline.







Actor: Entities that interact with the system or are external to it are shown.

Synchronous message: This is represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to queries before proceeding. Both the call and the response should be depicted in the diagram.

Asynchronous message: A solid line with a lined arrowhead is used to represent this. Asynchronous messages do not necessitate a response before the sender can proceed. The diagram should only include the call.

Delete message: An X follows a solid line with a solid arrowhead. This message has the effect of causing an object to be destroyed. Sequence diagram components:

Name	Description	Symbol
Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.	 The diagram shows a rectangular box representing an object. The top portion of the box is a header with the text 'Object Name: Classifier Name' in green. Below the header, the text '+Attributes' is written in green. The rest of the box is empty, representing the area for messages or actions.

The activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.	
Actor symbol	Shows entities that interact with or are external to the system.	
Lifeline symbol	Represents the passage of time a site extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labelled rectangle shape or an actor symbol.	
Alternative symbol	Symbolises a choice(that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labeled rectangle shape with a dashed line inside.	
Message symbol	This symbol is used when a sender needs to send a message.	
Reply message symbol	Represented by a dashed line with a lined arrowhead, these messages are replies to calls.	

Sequence diagram symbols

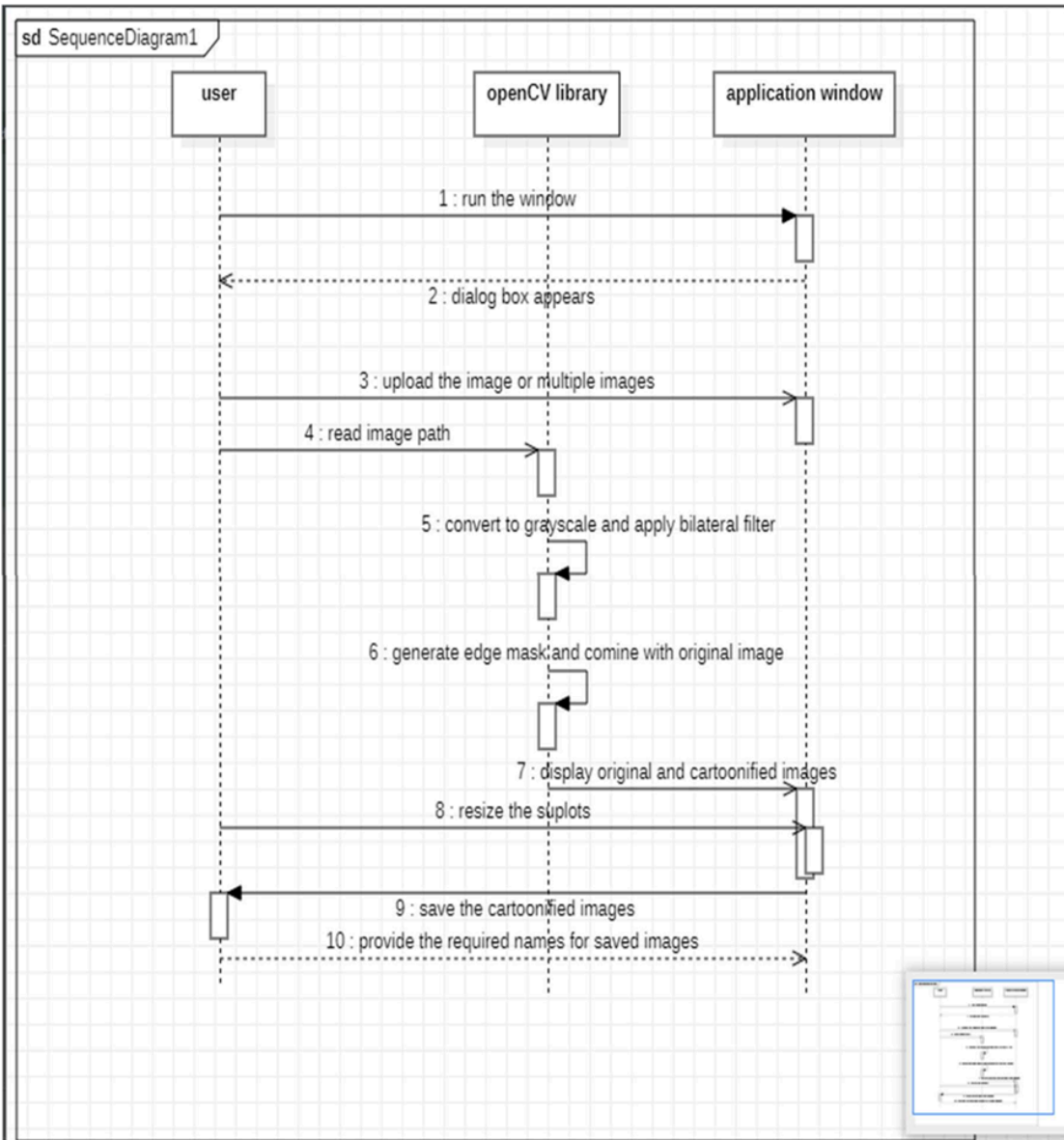


Fig 4.1.2.1: Sequence Diagram for application

In the above sequence diagram, the lifelines are:

- User
- OpenCV library
- Application window

The sequence starts from user running the application. A dialog box appears. The user can upload an image or multiple images at once. OpenCV library reads the image path and converts into grayscale followed by applying a bilateral filter. Then it generates an edge mask combining with original image.

Original and cartoonified images are displayed on the application window. User can resize subplots and save the images with required names.

4.1.3 ACTIVITY DIAGRAM

An activity diagram is a flowchart that displays the movement of information from one action to the next. A system operation can be used to describe the activity. From one operation to the next, the control flow is guided. In nature, this flow might be sequential, branching, or concurrent. By employing numerous parts such as join, fork and so on, activity diagrams cope with all sorts of flow control. Activity diagrams provide the same basic functions as the other four diagrams. It captures the dynamic behavior of the system. The other four diagrams depict message flow from one item to the next, whereas the activity diagram depicts. A specific system operation is referred to as an activity. It doesn't show any communication flow from one activity to the next. The phrases activity diagrams and flowcharts are often used interchangeably. Although the diagrams resemble flowcharts, they are not.

Notations

Initial point or start point. A small, filled circle, followed by an arrow, represents the beginning action state or starting point for any activity diagram. Make sure the start point of an activity diagram with swim lanes is in the top left corner of the first column.

Activity or Action state An action state is a representation of an object's non-interruptible action. You can make an action state in Smart Draw by sketching a rectangle with rounded corners.

Action flow Transitions from one action state to another are depicted by action flows, also known as edges and routes. An arrowed line is commonly used to depict them.

Decision sand branching A diamond signifies a multiple-choice decision. Place a diamond between the two activities when one requires a decision before moving onto the next. A condition or guard expression should be used to label the outgoing alternates. One of the paths can also be labeled "else."

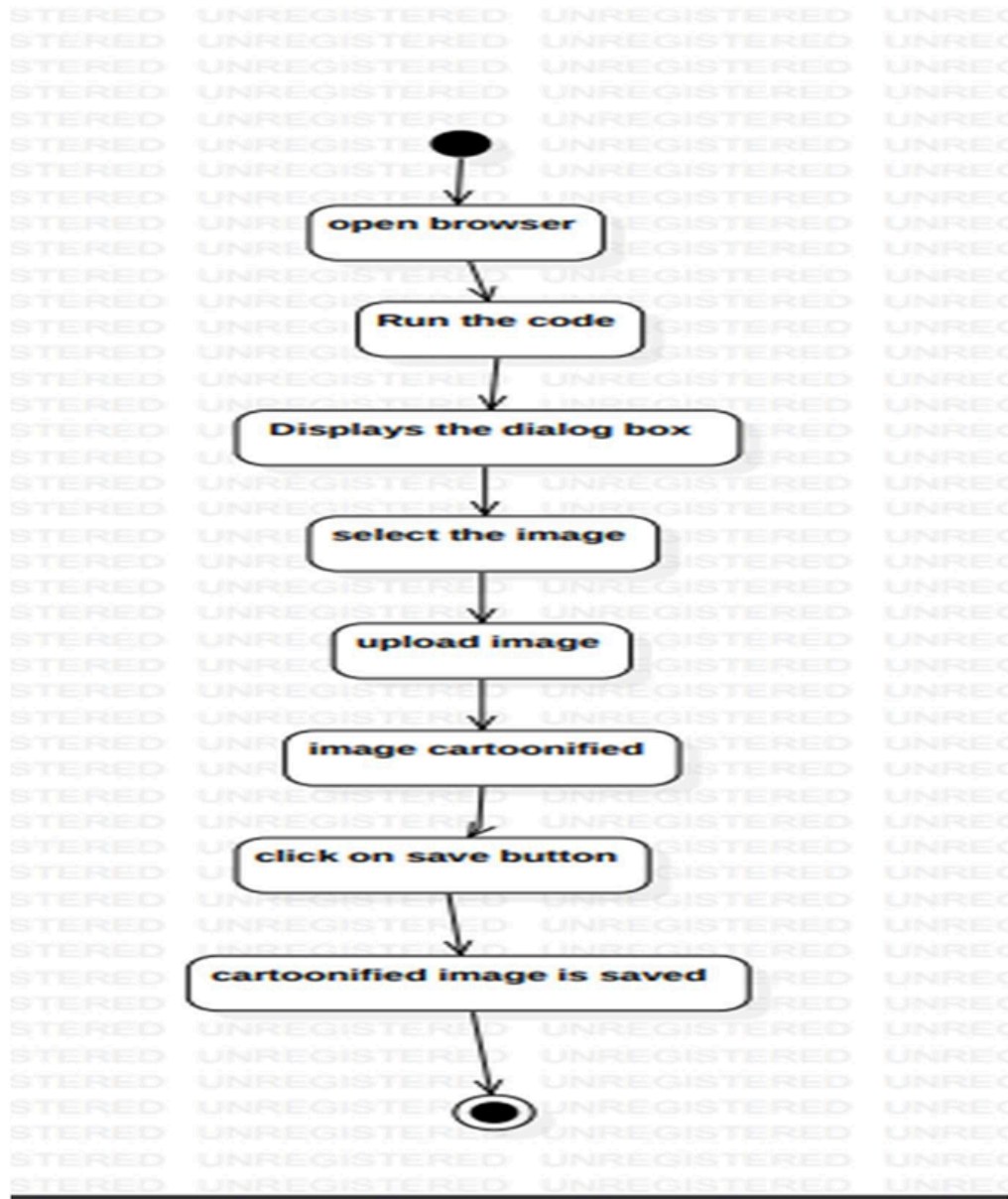


Fig4.1.3.1: Activity diagram for the system

This activity diagram shows the whole activity of the system. The Activity starts with the client submitting the data required for prediction and data validation, data preprocessing followed by predicting the results.

4.1.4 CLASS DIAGRAM

A static diagram is also referred to as a class diagram. It depicts the static view of an application. A class diagram can be used to visualize, describe, and document various parts of a system, as well as to create executable code for a software programmer.

The traits and activities of a class, as well as the constraints, are described in a class diagram.

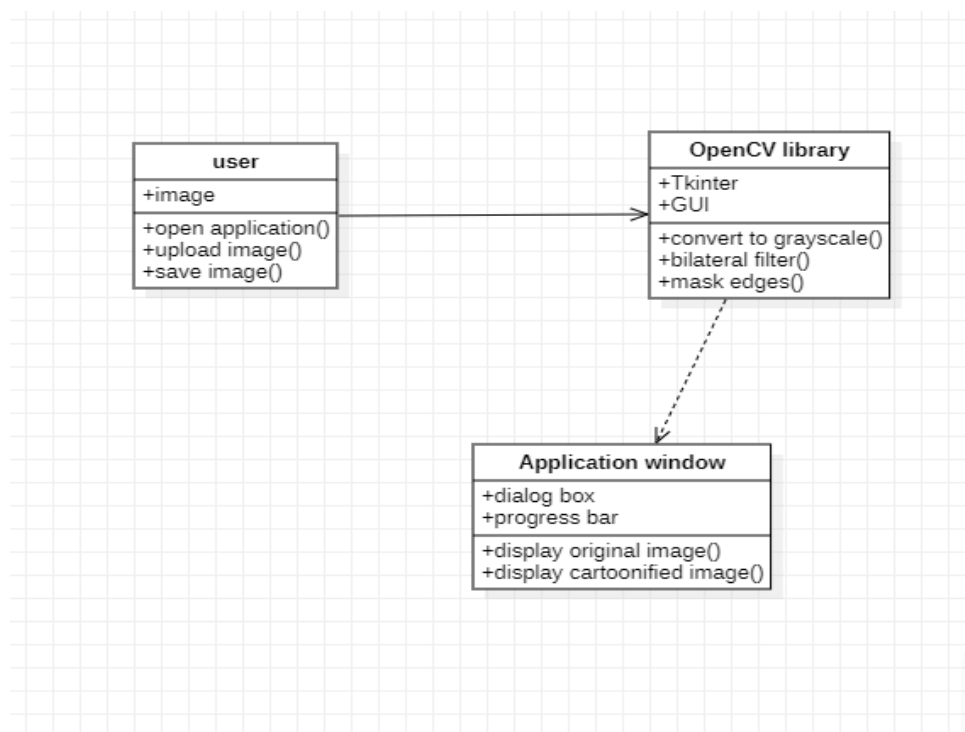


Fig 4.1.4.1: Class Diagram

A class diagram for recognizing the attributes and operations of the various classes in the application window. It is in an automated system would capture the essential classes and their relationships to model the structure and behavior of the system.

5. PROPOSED SYSTEM

5.1 METHODOLOGY

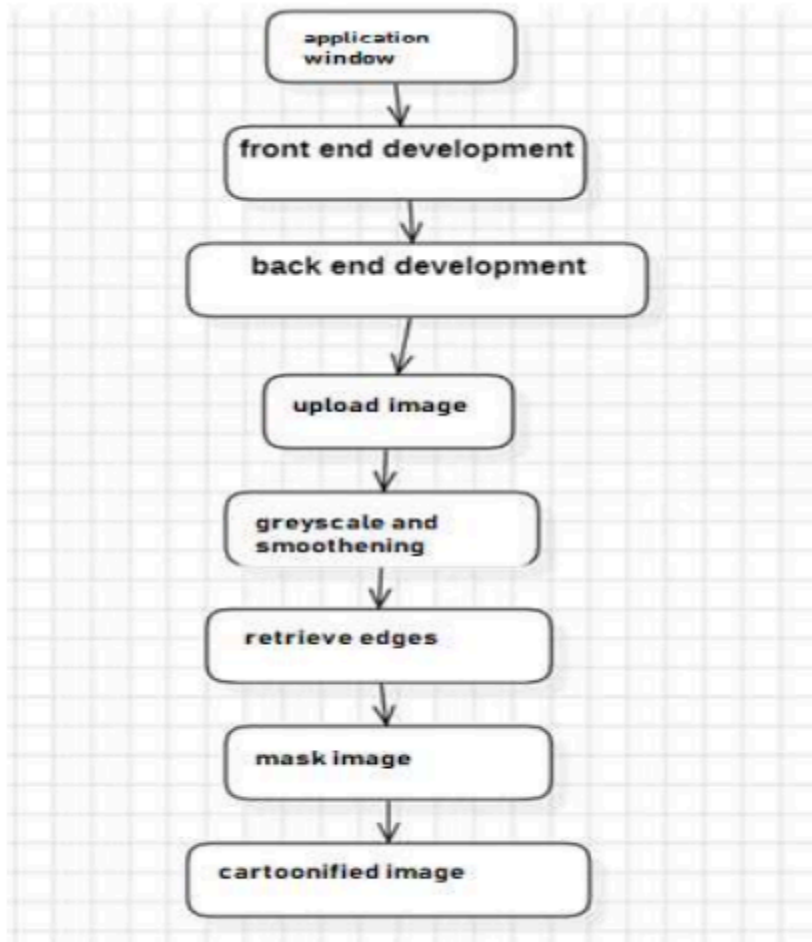


Fig 5.1.1 : Methodology

Model: When referring to a "model" for cartoonifying an image using OpenCV and Python, we are not discussing a machine learning model but a sequence of image processing operations.

- 1.OpenCV
- 2.Python Image file
- 3.Code editor

5.1 FUNCTIONALITIES

Read the Image:

Load the input image using OpenCV's `cv2.imread()` function. Convert to Grayscale: Convert the color image to grayscale using `cv2.cvtColor()`.

Apply Bilateral Filter:

Utilize a bilateral filter (`cv2.bilateralFilter()`) to smooth the image while preserving edges. This step reduces noise while retaining important features.

Generate Edge Mask:

Create an edge mask by applying adaptive thresholding (`cv2.adaptiveThreshold()`) to the smoothed grayscale image. This mask highlights significant edges in the image.

Combine Edges with Original Image:

Use the bitwise AND operation (`cv2.bitwise_and()`) to combine the edge mask with the original color image. This operation emphasizes edges, creating a cartoon-like effect.

Display Result:

Show the original and cartoonified images using OpenCV's `cv2.imshow()`.

Wait for User Input:

Utilize `cv2.waitKey()` to wait for user input before closing the display windows.

Close Windows:

Use `cv2.destroyAllWindows()` to close the display windows.

5.2 ADVANTAGES OF PROPOSED SYSTEM:

Simplicity: The system employs straightforward image processing techniques and is easy to understand and implement.

No Machine Learning Required: Unlike some advanced cartoonification methods, this system does not rely on complex machine learning models, making it accessible and lightweight.

Open Source and Free: OpenCV is an open-source library, and Python is a free programming language. The proposed system leverages these freely available tools, reducing software costs.

Customization: Users can easily adjust parameters in the script, such as filter sizes and thresholds, to customize the cartoonification effect according to their preferences.

Real-Time Processing: The system can process images in real-time, making it suitable for applications where quick cartoonification of images is required.

Cross-Platform Compatibility: The system can run on various operating systems, including Windows, Linux, and macOS, enhancing its compatibility.

No Internet Connection Needed: Since the system relies on local image processing techniques, it doesn't require an internet connection or external services.

Educational Value: The simplicity of the system makes it a great educational tool for learning about basic image processing concepts and techniques.

Quick Results: Users can obtain cartoonized images with minimal effort, making it a practical solution for individuals who need a quick and visually appealing effect.

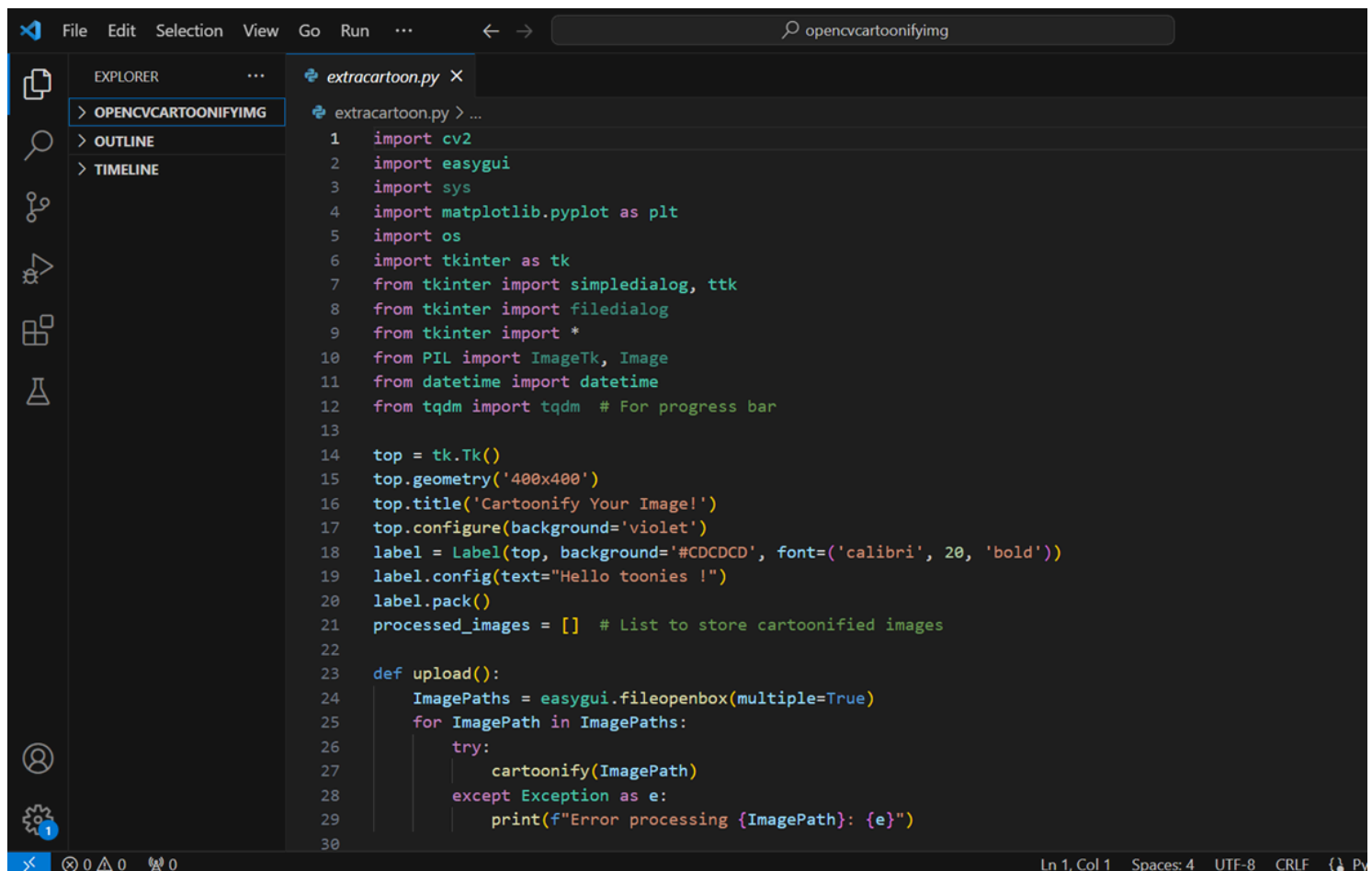
Community Support: OpenCV has a large and active community, providing support and resources for users seeking assistance or further customization.

6. CODING AND IMPLEMENTATIONS

IMPLEMENTATION

CREATE A TKINTER WINDOW

Initially we created a tkinter window for users to place their images from selected folders out of their respective drives.



```
File Edit Selection View Go Run ... ← → 🔍 opencvcartoonifying

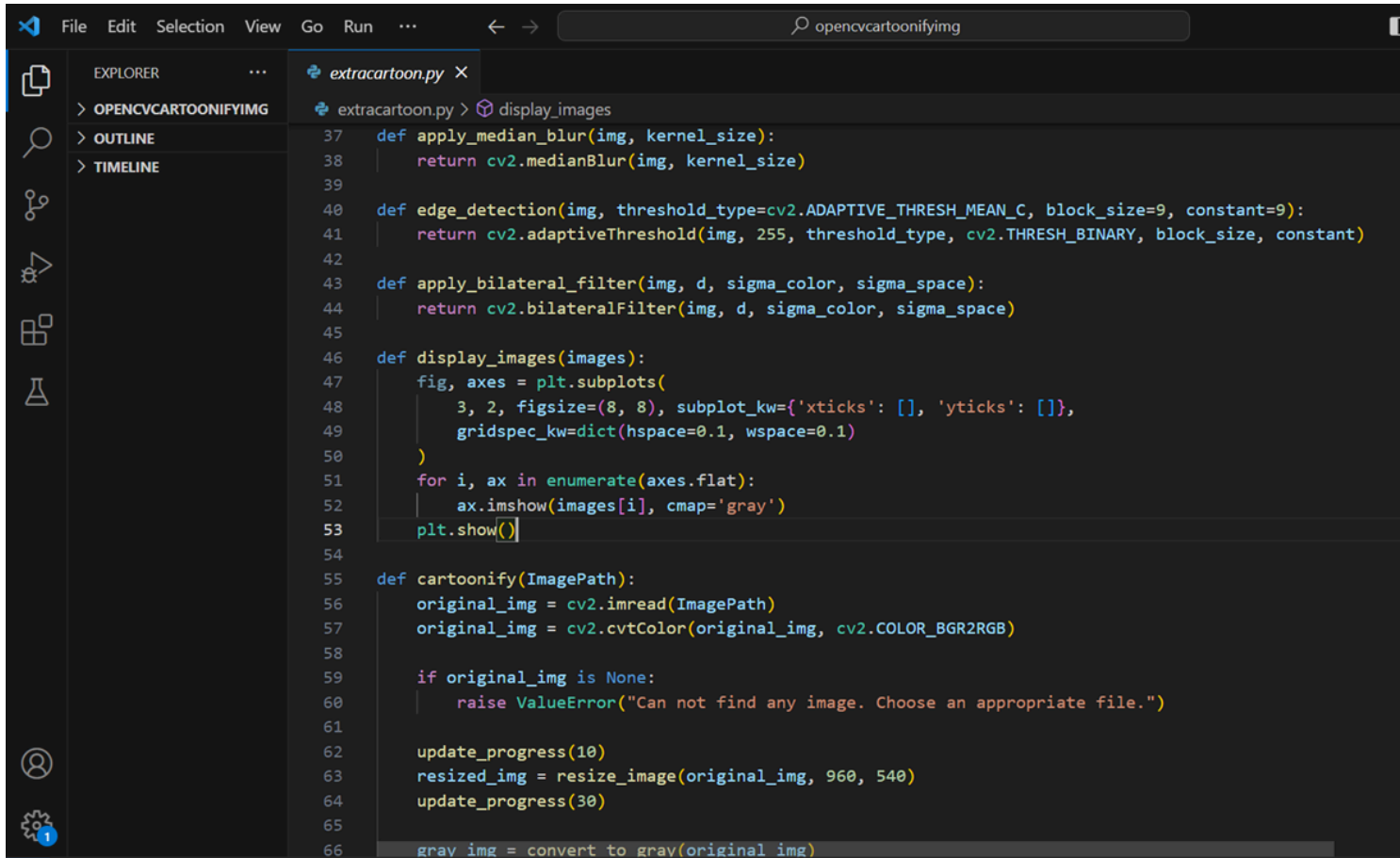
EXPLORER
> OPENCVCARTOONIFYIMG
> OUTLINE
> TIMELINE

extracartoon.py
1 import cv2
2 import easygui
3 import sys
4 import matplotlib.pyplot as plt
5 import os
6 import tkinter as tk
7 from tkinter import simpledialog, ttk
8 from tkinter import filedialog
9 from tkinter import *
10 from PIL import ImageTk, Image
11 from datetime import datetime
12 from tqdm import tqdm # For progress bar
13
14 top = tk.Tk()
15 top.geometry('400x400')
16 top.title('Cartoonify Your Image!')
17 top.configure(background='violet')
18 label = Label(top, background='#CDCDCD', font=('calibri', 20, 'bold'))
19 label.config(text="Hello toonies !")
20 label.pack()
21 processed_images = [] # List to store cartoonified images
22
23 def upload():
24     ImagePaths = easygui.fileopenbox(multiple=True)
25     for ImagePath in ImagePaths:
26         try:
27             cartoonify(ImagePath)
28         except Exception as e:
29             print(f"Error processing {ImagePath}: {e}")
30
```

Fig6.1.1 create a tkinter window

DEFINE FUNCTIONS

It includes applying medianblur, edge detection, bilateral filters, display images functions along with defining cartoonify images to apply different effects on original images to get an accurate cartoonified image.

A screenshot of a code editor interface with a dark theme. The Explorer panel on the left shows a project named 'OPENCVCARTOONIFYIMG' with subfolders 'OUTLINE' and 'TIMELINE'. The main editor area displays the file 'extracartoon.py' with the following Python code:

```
37 def apply_median_blur(img, kernel_size):
38     return cv2.medianBlur(img, kernel_size)
39
40 def edge_detection(img, threshold_type=cv2.ADAPTIVE_THRESH_MEAN_C, block_size=9, constant=9):
41     return cv2.adaptiveThreshold(img, 255, threshold_type, cv2.THRESH_BINARY, block_size, constant)
42
43 def apply_bilateral_filter(img, d, sigma_color, sigma_space):
44     return cv2.bilateralFilter(img, d, sigma_color, sigma_space)
45
46 def display_images(images):
47     fig, axes = plt.subplots(
48         3, 2, figsize=(8, 8), subplot_kw={'xticks': [], 'yticks': []},
49         gridspec_kw=dict(hspace=0.1, wspace=0.1)
50     )
51     for i, ax in enumerate(axes.flat):
52         ax.imshow(images[i], cmap='gray')
53     plt.show()
54
55 def cartoonify(ImagePath):
56     original_img = cv2.imread(ImagePath)
57     original_img = cv2.cvtColor(original_img, cv2.COLOR_BGR2RGB)
58
59     if original_img is None:
60         raise ValueError("Can not find any image. Choose an appropriate file.")
61
62     update_progress(10)
63     resized_img = resize_image(original_img, 960, 540)
64     update_progress(30)
65
66     gray_img = convert_to_gray(original_img)
```

Fig6.1.2 Define functions

SAVE IMAGES & UPDATE PROGRESS FUNCTION

Saves the cartoonified images with customizable filenames. Updates the value of the progress bar and refreshes the Tkinter window to show the progress.

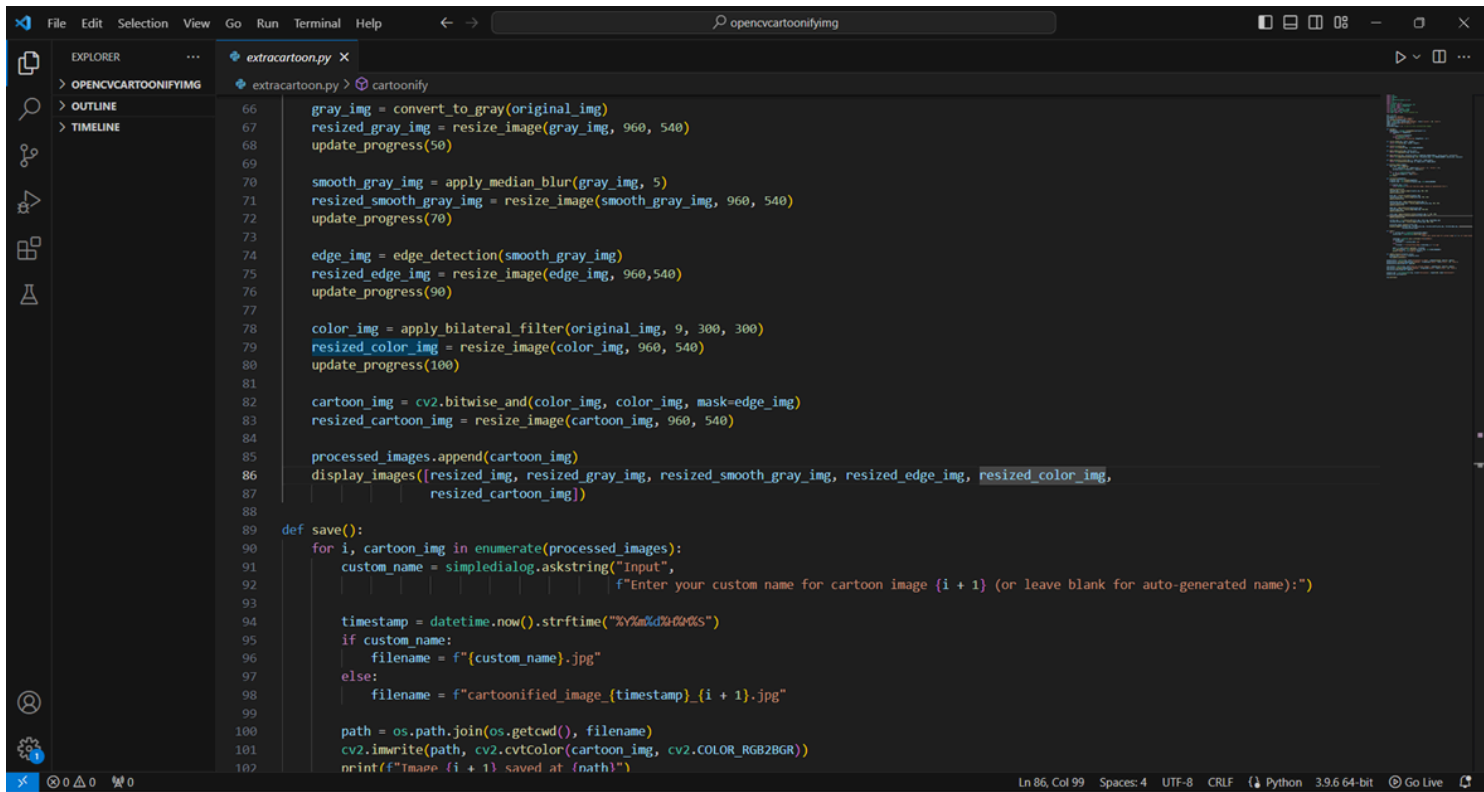


Fig6.1.3 Save Images & Update Progress Function

MAIN LOOP

Enters the Tkinter main event loop, allowing the GUI to respond to user actions.

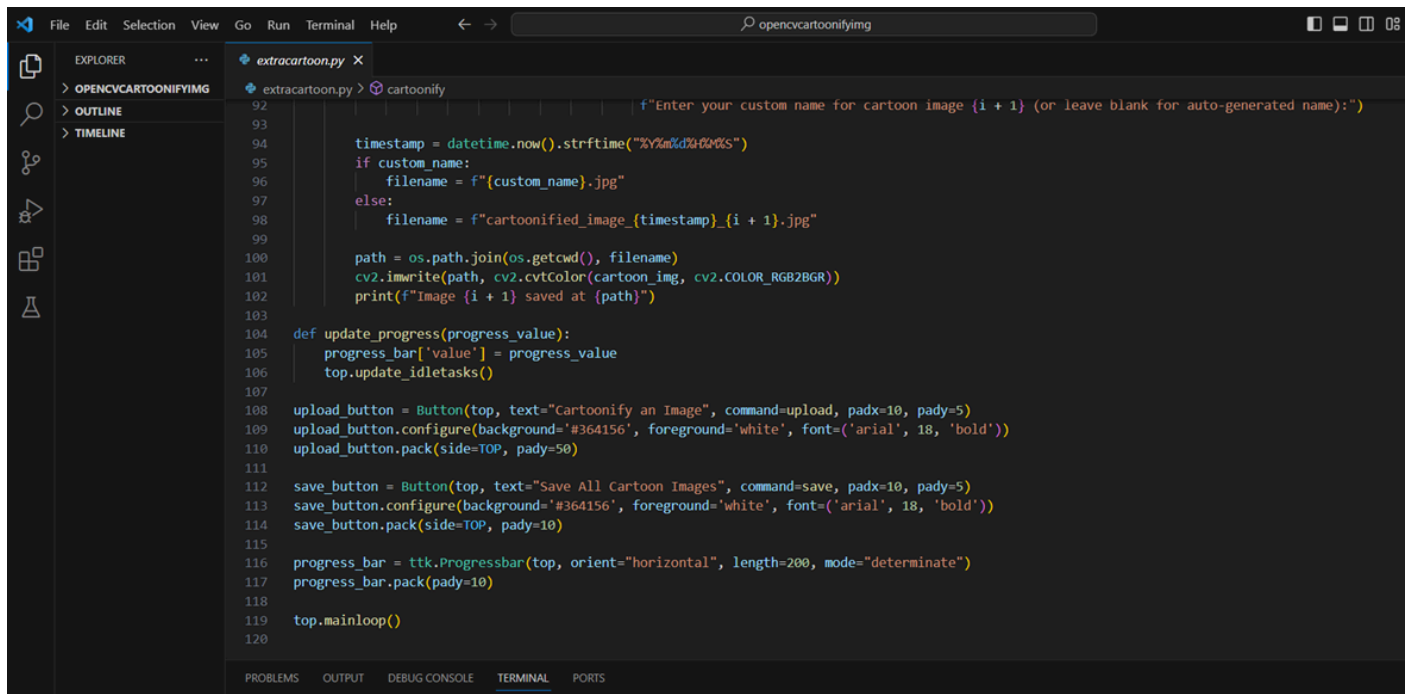


Fig6.1.4 Main Loop

7. TESTING

In machine learning, testing is mainly used to validate raw data and check the ML model's performance. The main objectives of testing machine learning models are:

- Quality Assurance
- Detect bugs and flaws

Once your machine learning model is built (with your training data), you need unseen data test your model. This data is called testing data, and you can use it to evaluate the performance and progress of your algorithms' training and adjust or optimize it for improved results.

Testing data has two main criteria. It should:

- Represent the actual dataset
- Be large enough to generate meaningful predictions

7.1 TYPES OF TESTING

7.1.1 MANUAL TESTING

Manual Testing is a type of software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identify the bugs, issues, and defects in the software application. Manual software testing is the most primitive technique of all testing types and it helps to find critical bugs in the software application.

Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual Testing concepts does not require knowledge of any testing tool. One of the Software Testing Fundamental is “**100% Automation is not possible**“. This makes Manual Testing imperative.

7.1.2 AUTOMATED TESTING

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Software Test Automation demands considerable investments of money and resources.

7.2 TESTING LEVELS

7.2.1 NON-FUNCTIONAL TESTING

Non-functional testing is a type of software testing to test non-functional parameters such as reliability, load test, performance and accountability of the software. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing. Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

7.2.1.1 PERFORMANCE TESTING

Performance testing is a form of software testing that focuses on how a system running the system performs under a particular load. This is not about finding software bugs or defects. Different performance testing types measure according to benchmarks and standards. Performance testing gives developers the diagnostic information they need to eliminate bottlenecks.

7.2.1.2 STRESS TESTING

Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions.

7.2.1.3 SECURITY TESTING

Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss. Security testing of any system is focused on finding all possible loopholes and weaknesses of the system which might result in the loss of information or reputation of the organization.

7.2.1.4 PORTABILITY TESTING

Portability Testing is one of Software Testing which is carried out to determine the degree of ease or difficulty to which a software application can be effectively and efficiently transferred from one hardware, software or environment to another one. The results of portability testing are measurements of how easily the software component or application will be integrated into the environment and then these results will be compared to the non-functional requirement of portability of the software system.

7.2.1.5 USABILITY TESTING

Usability Testing, also known as User Experience (UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use software applications to expose usability defects. Usability testing mainly focuses on the user's ease of using application, flexibility of application to handle controls and ability of application to meet its objectives. This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

7.2.2 FUNCTIONAL TESTING

It is a type of software testing which is used to verify the functionality of the software application, whether the function is working according to the requirement specification. In functional testing, each function is tested by giving the value, determining the output, and verifying the actual output with the expected value. Functional testing performed as black-box testing which is presented to confirm that the functionality of an application or system behaves as we are expecting. It is done to verify the functionality of the application. Functional testing is also called as black-box testing.

7.3 TEST CASES

Sl.no	Testcase	Expected Result	Actual Result	Pass/Fail
1.	After Unit testing with one image from a specific folder in any format	The outcomes are right up to 90% accuracy	Cartoonified image saved with given name. If not saved by default timestamp at desired folder mentioned	PASS
2.	After training testing with Multiple images given at once from a folder in any format	All of the outcomes are right up to 90% accuracy	Cartoonified image saved with given name. If not saved by default timestamp at desired folder mentioned	PASS

Table7.3.1: Testcases

8. RESULTS

To show a dialog box for image selection from their desired folders of their respective drives

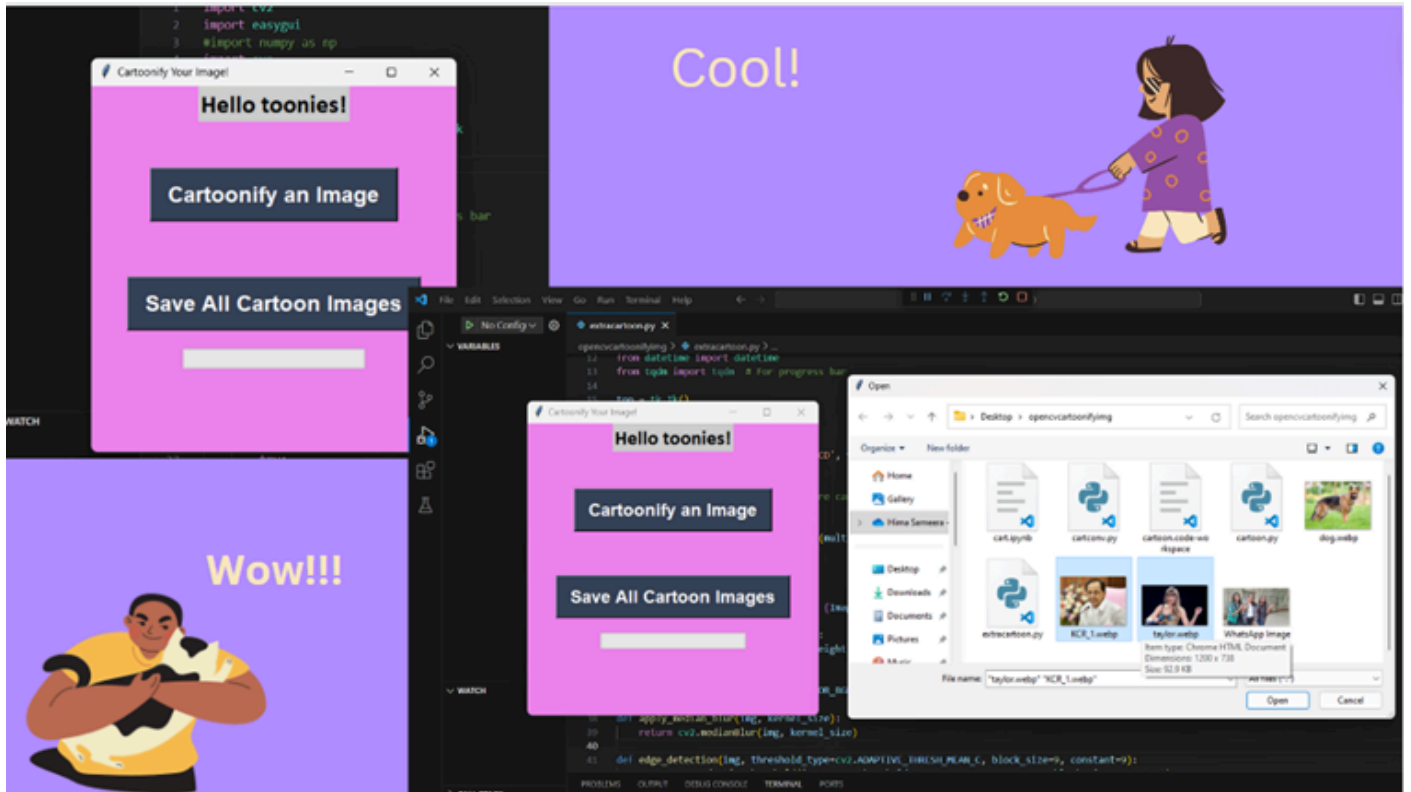


Fig8.1 User display to select needed images from the folder

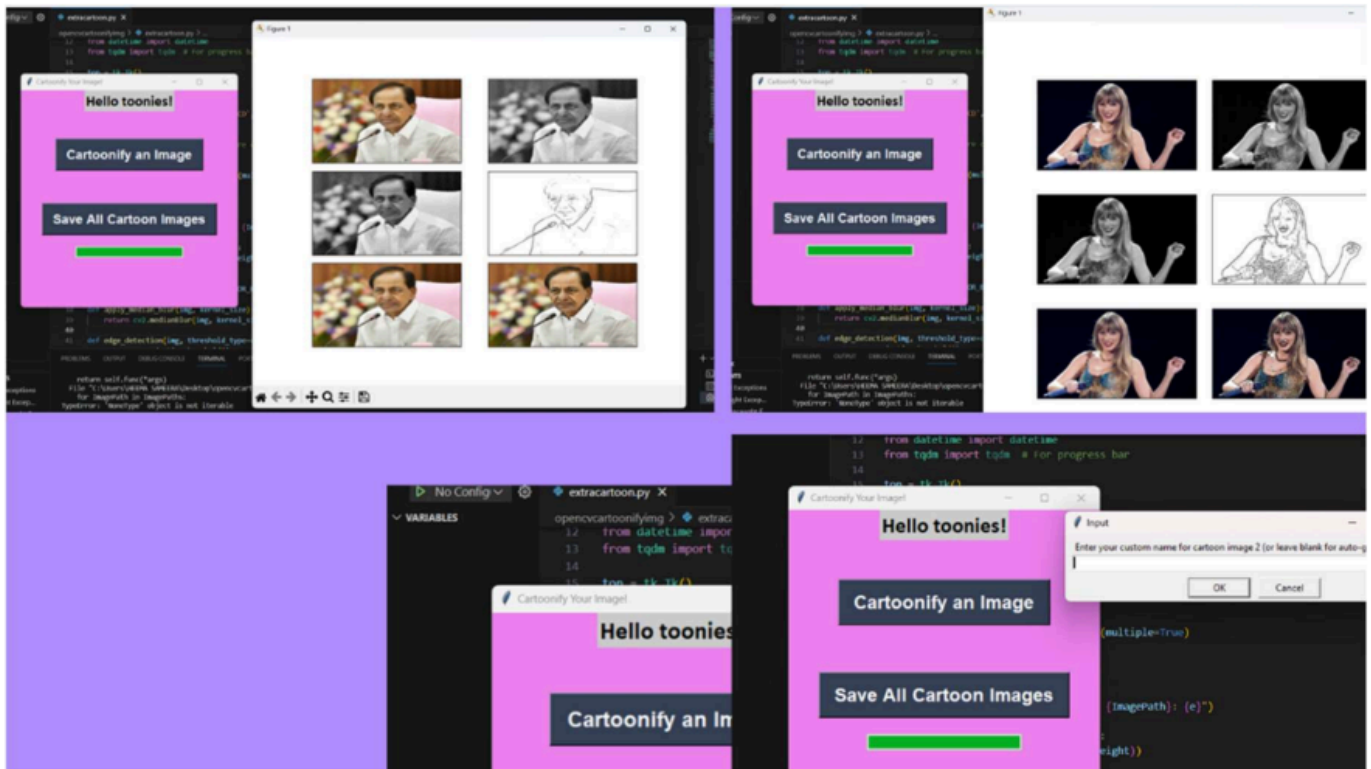


Fig8.2 Asking user to save cartoonified images by desired names in the folders also showing the process involved

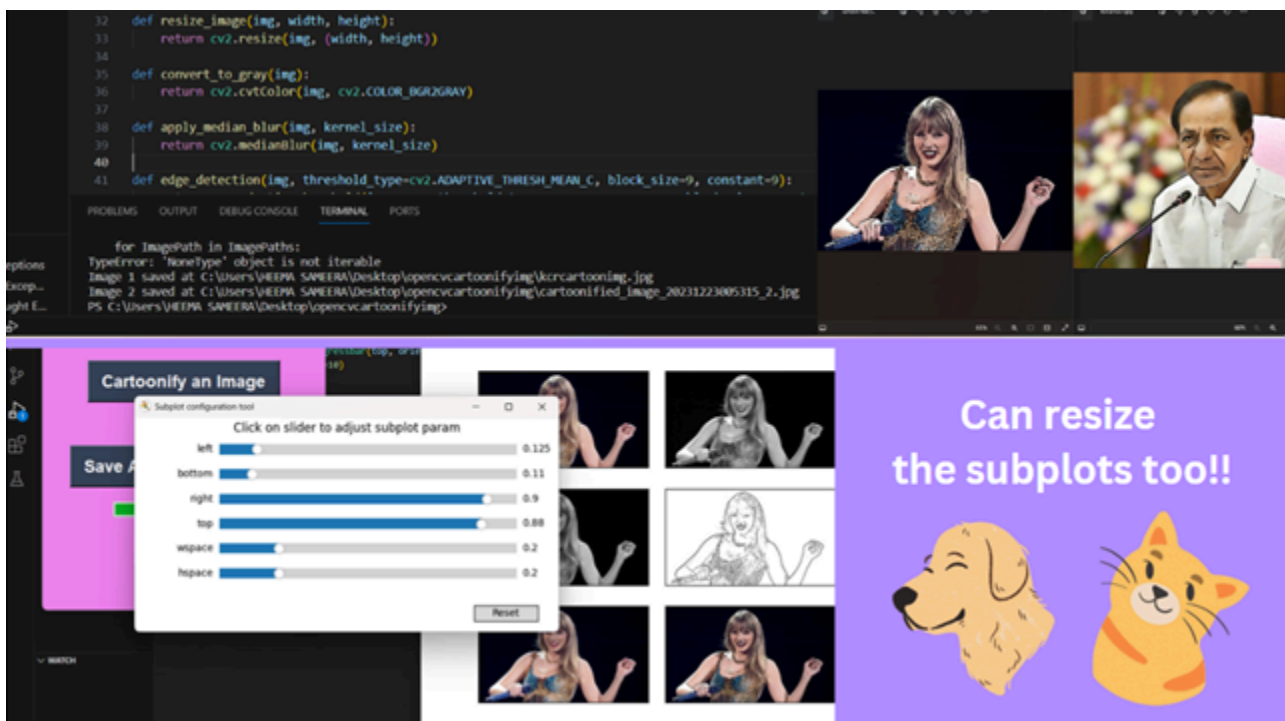


Fig8.3 Can Resize the subplots too

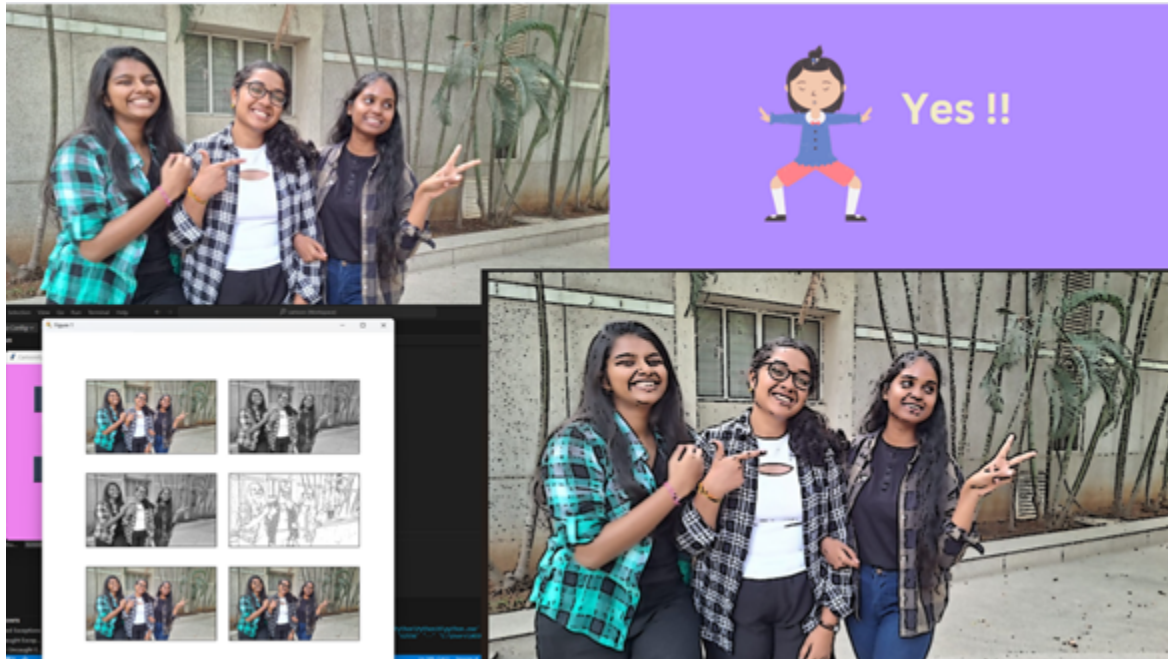


Fig8.4 Cartoonifying our very own image

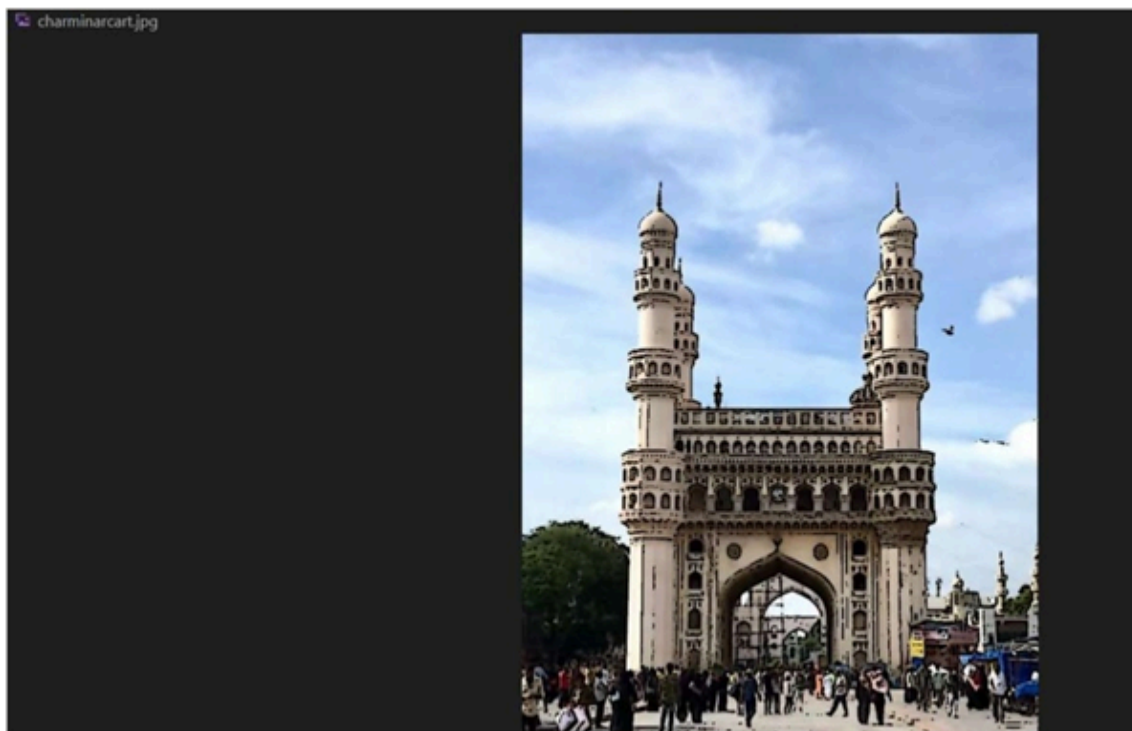


Fig8.5 Charminar's converted cartoon image

9.CONCLUSION AND FURTHER WORK

In conclusion, the proposed system for cartoonifying images using OpenCV and Python presents a simple and accessible solution for transforming images into a cartoon-like style. Leveraging the image processing capabilities of OpenCV, the system offers an efficient and lightweight approach, making it suitable for both educational purposes and practical applications. The straightforward implementation, absence of complex machine learning models, and ease of customization make it an attractive choice for users seeking a quick and visually appealing cartoon effect.

Further Work: Future work can focus on enhancing user interaction through the development of a graphical user interface, enabling real-time video processing, and exploring advanced techniques like neural style transfer for diverse artistic styles. Additionally, efforts can be directed towards optimizing system performance, fine-tuning parameters for different image types, and creating cross-platform applications for wider accessibility. Documentation and tutorials should be provided to assist users in understanding and customizing the system effectively. These improvements would contribute to making the system more versatile, user-friendly, and adaptable to various use cases.

REFERENCES

- [1] Kumar, Saurabh, Utkarsh Bhardwaj, and T. Poongodi. "Cartoonify an Image using OpenCV in Python." 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM). IEEE, 2022.
- [2]Soni, Aaditaa, and Anand Sharma. "Image Transformation into Cartoon Using OpenCV."Contemporary Issues in Communication, Cloud and Big Data Analytics: Proceedings of CCB 2020.Springer Singapore, 2022.
- [3]Smith, John. "Animating Static Pictures: A Cartoon Transformation Approach with OpenCV and Python." International Journal of Creative Research In Computer Technology and Design 5.5 (2023).
- [4]Tomar, Suryansh, et al. "An Effective Cartoonifying of an Image using Machine Learning." 2023 IEEE 4th Annual Flagship India Council International Subsections Conference (INDISCON). IEEE, 2023.
-