NAAC Accredited with 'A" Grade
Recognised by UGC under sections 2(f) and 12(b) & Affiliated to Bangalore University

DEPARTMENT OF COMPUTER APPLICATIONS

T. JOHN COLLEGE
(Affiliated to Bangalore University & Approved to AICTE)
Gottigere, Bengaluru – 560 083

OPEN-AI IMAGE GENERATOR

A project report submitted to the Bangalore University in the partial fulfilment
Of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATION

Submitted by

Naveen Kumar S R

(P03ML21S0047)

Under the guidance of

Dr. FELCY JUDITH
(Head of the Department, Computer Applications)

June – 2023

NAAC Accredited with 'A" Grade
Recognised by UGC under sections 2(f) and 12(b) & Affiliated to Bangalore University

DEPARTMENT OF COMPUTER APPLICATIONS

T. JOHN COLLEGE
(Affiliated to Bangalore University & Approved to AICTE)
Gottigere, Bengaluru – 560 083

PROJECT WORK

OPEN-AI IMAGE GENERATOR

Bonafide Work Done by

Naveen Kumar S R

(P03ML21S0047)

The project submitted in partial fulfilment of the requirements for the award of Master's of Computer Application, of Bangalore University, Bengaluru.

_____	_____
GUIDE	**HEAD OF THE DEPARTMENT**

Submitted for the Viva-Voce Examination held on _____

_____	_____
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

I, **Naveen Kumar S R (MCA215012)** hereby declare that the project entitled **OPEN-AI IMAGE GENERATOR** is my project work carried out during the 3rd Semester MCA at T. John College, Bengaluru, under the guidance of **Dr. FELCY JUDITH** HOD, Department of Computer Applications, T. John College and has not submitted previously for the award of any other degree or diploma by me to any institution or university according to the best of my knowledge.

SIGNATURE OF THE CANDIDATES

Place: Bengaluru
Date:

ACKNOWLEDGEMENT

The project titled **OPEN-AI IMAGE GENERATOR**

has been successfully completed my project with the help of my faculty. I thank her for providing great tips that I have incorporated in this project.

I would express my sincere thanks to **Sr. ROSE MARY BALAPPA**, Principal, T. John College for her valuable advice, co-operation and support and for providing all the faculties for the construction of the project.

I am thankful to **Dr. FELCY JUDITH**, Head, Department of Computer Applications department for her valuable advice and co-operation.

Last but not the least we also like to thank our friends and family for their valuable support and encouragement.

CONTENTS

“OPEN-AI IMAGE GENERATOR”

CHAPTER 1

1.1 INTRODUCTION

AI Image Generator : When people listen to or read a narrative, they quickly create pictures in their mind to visualize the content. Many cognitive functions, such as memorization, reasoning ability, and thinking, rely on visual mental imaging or “seeing with the mind’s eye”. Developing a technology that recognizes the connection between vision and words and can produce pictures that represent the meaning of written descriptions is a big step toward user intellectual ability.

Image-processing techniques and applications of computer vision (CV) have grown immensely in recent years from advances made possible by artificial intelligence and deep learning’s success. One of these growing fields is text-to-image generation. The term text-to image (T2I) is the generation of visually realistic pictures from text inputs. The model takes an input in the form of human written description and produces a RGB image that matches the description. T2I generation has been an important field of study due to its tremendous capability in multiple areas. Photo-searching, photo-editing, art generation, captioning, portrait drawing, industrial design, and image manipulation are some common applications of creating photo-realistic images from text.

The Image Generator project, developed by OpenAI, is an innovative research initiative that explores the intersection of natural language processing and image generation. "Drawing Artificial Intelligence that Learns and Generates Images from Text." It focuses on generating high-quality images based on textual descriptions or prompts.

The key idea behind Image Generator is to train a large-scale language model using a vast dataset of text-image pairs. By leveraging deep learning techniques and generative models, It learns to associate textual descriptions with corresponding visual representations. This enables the model to generate novel and coherent images based on given textual prompts.

1.2PURPOSE OF THE PROJECT

The purpose of the Image Generator project, developed by OpenAI, is to explore and push the boundaries of generative AI by creating a model capable of generating images from textual descriptions. The project aims to demonstrate the power of combining natural language processing and image generation, highlighting the potential for multimodal AI applications.

AI system that can create realistic images and art from a description in natural language, Text-to-image generation is a method used for generating images related to given textual descriptions, a deep learning models developed by OpenAI to generate digital images from natural language descriptions, called "prompts". It can produce images for a wide variety of arbitrary descriptions from various viewpoints with only rare failures. OpenAI has the ability to manipulate and rearrange objects in generated imagery and also create things that don’t exist.

In early November 2022, OpenAI released an API, allowing developers to integrate the model into their own applications.

What this module actual work is capturing some element of human imagination. It's not actually that different than how humans can read a book and imagine things, but it's being able to capture that intelligence with an algorithm. This allows users to create images quickly and easily, and artists and creative professionals are using to inspire and accelerate their creative processes. It has a significant influence on many research areas as well as a diverse set of applications (e.g., photo-searching, photo-editing, art generation, computer-aided design, image reconstruction, captioning, and portrait drawing). The most challenging task is to consistently produce realistic images according to given conditions.
Existing algorithms for text-to-image generation create pictures that do not properly match the text. OpenAI considered this issue and study and built a deep learning-based architecture for semantically consistent image generation.

One remarkable aspect of DALL·E is its ability to generate highly imaginative and creative images. It can generate images of nonexistent objects, scenes, or concepts, often incorporating intricate details and realistic textures. The model has demonstrated its capabilities in producing visually appealing and semantically aligned images, showing potential for various applications in design, storytelling, and creative expression.

It has sparked significant interest in the research and AI communities, showcasing the power of combining language understanding with image synthesis. While specific details of the AI architecture and training methodology remain proprietary, the project represents a significant milestone in advancing the field of multimodal AI, pushing the boundaries of what is possible in generating visual content from textual inputs.

In summary, the purpose of this project is to explore the capabilities of generative AI, showcase the potential of combining natural language processing and image generation, and advance the field of multimodal AI, particularly in the domain of creative content generation.

Introduction to OpenAI

OpenAI is a leading artificial intelligence (AI) research lab and company that aims to ensure that artificial general intelligence (AGI) benefits all of humanity. Founded in December 2015 by Elon Musk, Sam Altman, Greg Brockman, Ilya Sutskever, John Schulman, and Wojciech Zaremba, OpenAI has made significant contributions to the field of AI and has played a crucial role in advancing the capabilities of machine learning models.

OpenAI's Research and Achievements

OpenAI is renowned for its groundbreaking research and advancements in the field of AI. One of its notable contributions is the development of the GPT (Generative Pre-trained Transformer) series of models. GPT-3, released in June 2020, is one of the largest and most powerful language models to date. It has demonstrated impressive capabilities in natural language understanding and generation, enabling it to perform a wide range of tasks such as language translation, question answering, and text completion.

OpenAI has also made significant strides in reinforcement learning, an area of AI concerned with training agents to make sequential decisions. OpenAI's research has led to breakthroughs in areas like robotics, game playing, and autonomous systems. Notably, OpenAI's AI agents have achieved remarkable results in complex games like Dota 2 and have shown the ability to learn from scratch and surpass human performance.

OpenAI's Commitment and Impact

OpenAI is driven by a strong commitment to ensuring the benefits of AGI are shared widely. They prioritize long-term safety and seek to prevent any potential harmful consequences that may arise from

AGI development. OpenAI has committed to using any influence they obtain over AGI to ensure it benefits everyone and avoids causing harm or concentrating power.

OpenAI also values cooperation and collaboration with other research and policy institutions. They actively publish most of their AI research, aiming to contribute to the global knowledge base and foster the development of safe and beneficial AI technologies. However, there may be certain safety and security concerns that limit the immediate disclosure of certain findings.

In addition to research, OpenAI has developed various initiatives and programs to promote AI education and accessibility. They have created resources, including AI textbooks and tutorials, to help individuals learn about AI and its applications. OpenAI also offers the OpenAI API, which allows developers to access and utilize GPT-3's capabilities in their own applications.

Overall, OpenAI is a pioneering AI research lab and company that has made significant contributions to the field of AI. With a strong commitment to safety, ethics, and the wide distribution of benefits, OpenAI continues to push the boundaries of AI research and development to create a positive impact on society.

CHAPTER 2

2.1 LITERATURE SURVEY

A brief overview of the literature survey related to the image generation model developed by OpenAI. the details of its architecture and training process have not been disclosed publicly. However, there have been research papers and articles discussing related topics and techniques. Here is a literature survey related to image generation and language models:

1. "Image Generation with Conditional Generative Adversarial Networks" by Mirza and Osindero (2014): This paper introduced conditional generative adversarial networks (cGANs), a framework for generating images based on textual descriptions or other conditional input.

2. "Generative Adversarial Text to Image Synthesis" by Reed et al. (2016): This work explored the synthesis of images from textual descriptions using the combination of generative adversarial networks (GANs) and recurrent neural networks (RNNs).

3. "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks" by Zhang et al. (2017): The paper proposed a two-stage GAN architecture that generates high-resolution images from text descriptions by progressively refining the image quality.

4. "AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks" by Xu et al. (2018): This work introduced an attentional generative adversarial network (AttnGAN) that pays attention to specific words in a text description to generate more detailed and realistic images.

5. "ControlGAN: Fine-grained Text-to-Image Generation with Controllable Styles" by Shen et al. (2019): The paper presented a controllable image generation model that allows users to manipulate specific attributes or styles of the generated images based on textual input.

6. "CLIP: Connecting Text and Images" by Radford et al. (2021): This influential paper introduced CLIP, a model that learns to associate images and textual descriptions. It demonstrated impressive performance on a range of language and vision tasks.

7. "Generating Sequences With Recurrent Neural Networks" by Graves (2013): This paper introduced

the concept of recurrent neural networks (RNNs) for generating sequences, laying the foundation for subsequent advancements in text generation models.

8. "Generative Adversarial Networks" by Goodfellow et al. (2014): This seminal paper proposed the concept of generative adversarial networks (GANs), which have been widely adopted in image generation tasks. GANs consist of a generator and a discriminator that compete against each other to improve the quality of generated images.

10. "Attention Is All You Need" by Vaswani et al. (2017): This influential paper introduced the transformer architecture, which leverages self-attention mechanisms to capture relationships between words in a text sequence. Transformers have significantly improved language modeling tasks and have potential applications in image generation.

11. "Image Transformer" by Parmar et al. (2018): This paper extended the transformer architecture to image generation, demonstrating that transformers can generate high-quality images without the need for convolutional layers traditionally used in computer vision tasks.

12. "BigGAN: Large-Scale Generative Adversarial Networks for Image Synthesis" by Brock et al. (2019): This work introduced BigGAN, a GAN-based image generation model capable of synthesizing high-resolution and diverse images. It highlighted the importance of scale and architectural choices in improving image generation performance.

13. "Generative Pre-trained Transformer" (GPT): This paper, authored by Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever, introduced the GPT series of models. The original GPT, GPT-2, and GPT-3 models have revolutionized natural language processing tasks, demonstrating exceptional performance in language understanding, text generation, and various downstream applications.

14. "Proximal Policy Optimization Algorithms": This paper, written by OpenAI researchers Schulman, Wolski, Dhariwal, Radford, and Klimov, proposed the Proximal Policy Optimization (PPO) algorithm for reinforcement learning. PPO has become a popular algorithm in the field, known for its simplicity and effectiveness in training policies for sequential decision-making tasks.

15. "Playing Atari with Deep Reinforcement Learning": Authored by Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al., this paper introduced the Deep Q-Network (DQN). DQN demonstrated significant advancements in playing Atari games by learning directly from

raw pixels, showcasing the power of deep reinforcement learning.

16. "Unsupervised Learning of Video Representations using LSTMs": This paper, by Srivastava, Mansimov, and Salakhudinov, presented a method for unsupervised learning of video representations using Long Short-Term Memory (LSTM) networks. This work contributed to the understanding of video processing and representation learning, enabling the extraction of meaningful features from video data.

17. "Reinforcement Learning with Unsupervised Auxiliary Tasks": Authored by Jaderberg, Mnih, Czarnecki, Schaul, Leibo, Silver, and Kavukcuoglu, this paper introduced the concept of auxiliary tasks in reinforcement learning. It demonstrated that auxiliary tasks, such as predicting future states or rewards, can greatly improve the learning efficiency and performance of RL agents.

CHAPTER 3

PROPOSED METHODOLOGY

3.1 DATASET

The dataset used for training a image generator consists of pairs of textual descriptions or prompts and their corresponding images. While the exact details of the dataset used by OpenAI itself are proprietary, the general approach involves collecting a large and diverse dataset from various sources. The dataset should cover a wide range of visual concepts and semantic descriptions to train the model effectively.

Typically, the dataset includes images sourced from the internet, image repositories, or curated collections. These images can encompass a broad array of subjects, objects, scenes, and visual attributes. It is important to ensure that the dataset is representative of the target domain and covers a wide spectrum of visual diversity.

For each image in the dataset, there needs to be one or more associated textual descriptions or prompts. These descriptions can be manually curated or collected from various sources, such as image captions, image annotations, or user-generated tags. The descriptions should provide meaningful information about the content, attributes, or context of the images.

To align the textual descriptions with their corresponding images, the dataset should organize the pairs in a structured manner. Each image should be linked or associated with the relevant textual description or prompt, forming input-output pairs for training the image generation model.

It's worth mentioning that constructing a high-quality and diverse dataset is crucial for training an effective image generation model like ImageGenerator. Proper data preprocessing, including data cleaning, normalization, and ensuring balanced representation of different concepts, can contribute to the overall performance and generalization of the model.

While the specific details of the dataset used by OpenAI are not publicly disclosed, the above considerations provide a general understanding of the type of dataset that is typically used for training an image generation model.

3.2 DATA PROCESSING

The data processing involved in a image generator system can be complex and multi-faceted. While It can provide a high-level overview.

1. Data Collection: The system gathers a diverse dataset of images from various sources. This dataset serves as the training data for the ImageGenerator model.

2. Preprocessing: The collected images undergo preprocessing to ensure consistent formats, resolutions, and quality. This step may involve resizing, cropping, normalization, or other image transformations.

3. Textual Data Collection: Alongside the image dataset, textual descriptions or prompts are collected or curated. These descriptions serve as inputs for the model during training and generation.

4. Tokenization: The textual descriptions are tokenized, splitting them into smaller units such as words or subwords. Tokenization allows the model to understand and process the text effectively.

5. Training Data Preparation: The dataset comprising the tokenized textual descriptions and corresponding images is formatted for training. This involves creating input-output pairs, aligning the descriptions with the corresponding images.

6. Model Training: The prepared training data is fed into the model, which undergoes a training process. The model learns to generate images based on textual descriptions through iterative optimization and adjustment of its internal parameters.

7. Inference: Once the model is trained, it can generate images from new textual prompts. During inference, the textual prompt is encoded and processed by the model, which generates an image as output.

8. Post-processing: The generated image may undergo post-processing steps, such as color adjustment, filtering, or resizing, to ensure desired quality and consistency.

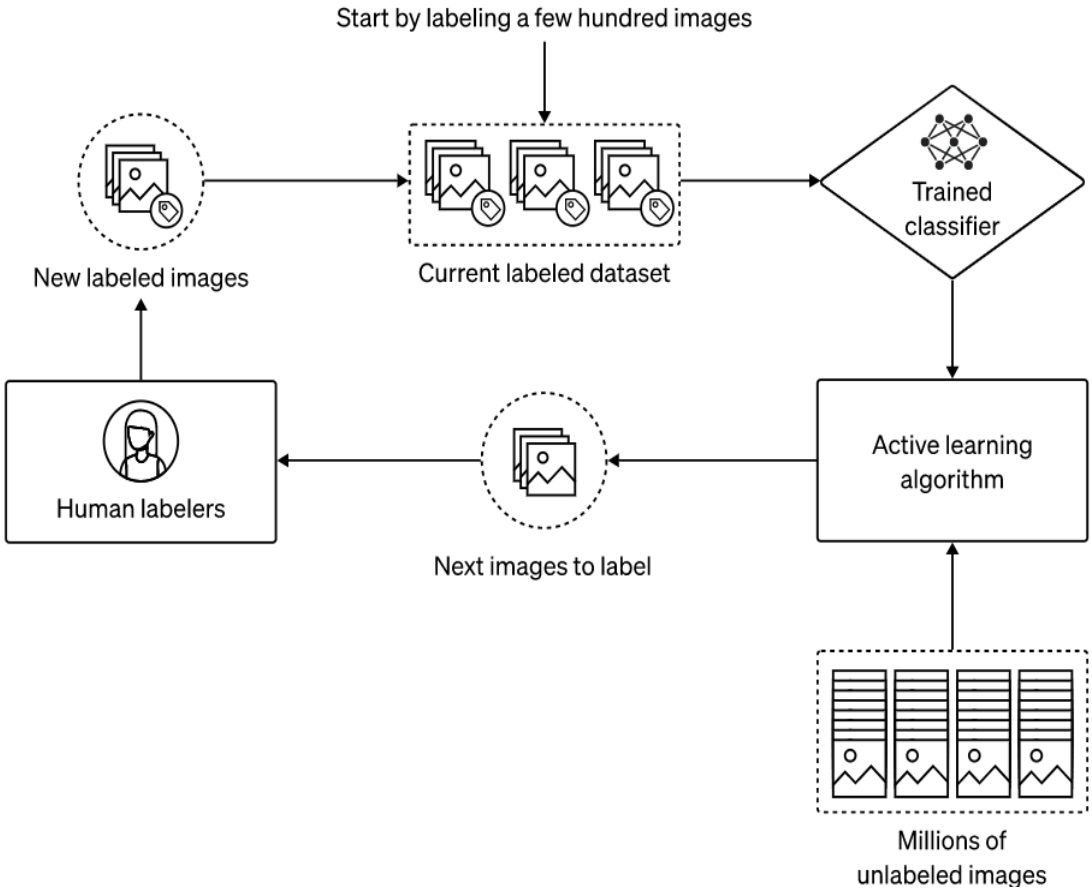
9. Output Delivery: The final processed image is delivered as the output of the image generation system, either for display to the user or for further integration into applications or systems.

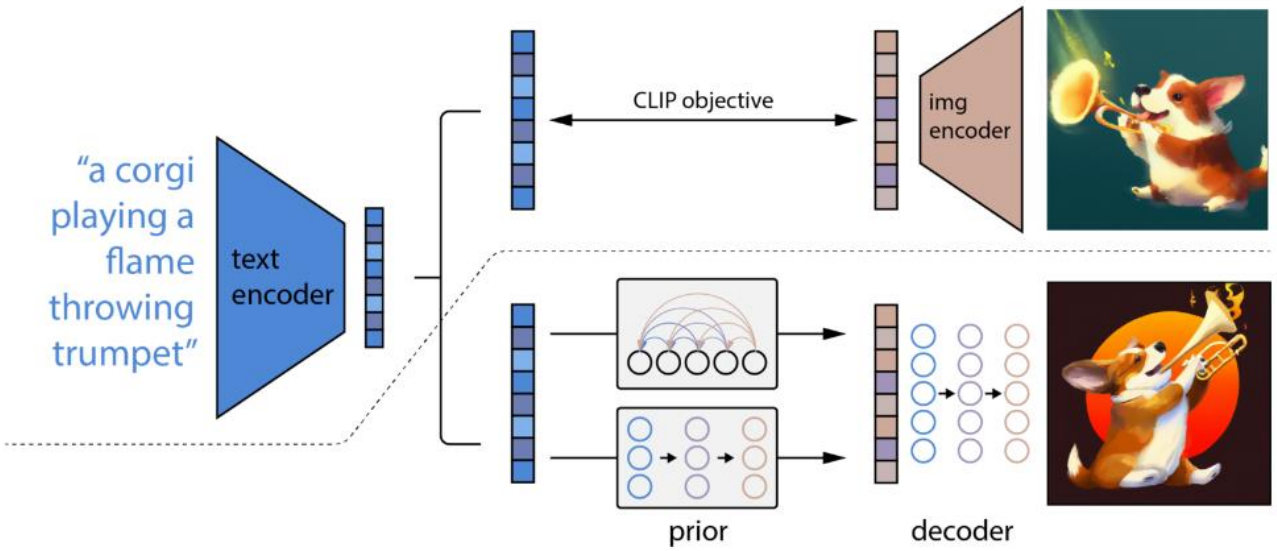
CHAPTER 4

4.1 SYSTEM ANALYSIS AND DESIGN

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

DATA FLOW DIAGRAM





A textual representation of a data flow diagram for OpenAI ImageGenerator. it is an image generation model, and a typical data flow diagram might not capture the intricacies of its internal processes. However, we can outline a simplified data flow diagram for this module:

1. User Input: The user provides a textual description or prompt to generate an image.

2. Input Processing: The system processes the user input, tokenizes it, and prepares it for further processing.

3. Text Encoding: The text is encoded into a numerical representation that the model can understand.

4. OpenAI Model: The encoded text is fed into the DALL·E model, which generates an image based on the provided prompt.

5. Image Generation: This model processes the encoded text and generates a corresponding image.

6. Image Rendering: The generated image is rendered and prepared for display or further processing.

7. Output Delivery: The system delivers the generated image to the user or the intended recipient.

It's important to note that the actual data flow and internal processes of OpenAI Module are more complex and involve sophisticated deep learning techniques. The above representation simplifies the data flow for illustrative purposes, highlighting the core steps involved in generating an image from a textual prompt.

CHAPTER 5

5.1 SOFTWARE REQUIREMENTS

REQUIREMENT	SOFTWARE ELEMENTS	SPECIFICATION
Software	**OS**	**WINDOWS, LINUX**
	React JS, **HTML** **CSS, JavaScript**	**Visual Studio code ,**

5.2 HARDWARE REQUIREMENT

REQUIREMENT	HARDWARE ELEMENTS	SPECIFICATION
Hardware	**Processor**	**i3**
	RAM	**2GB**
	Hard Disk	**250GB**

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1

1. **HTML** : (Hypertext Markup Language) is the standard markup language used to create and structure webpages. It uses a set of tags to define the elements and content within a webpage. These tags indicate headings, paragraphs, links, images, tables, forms, and more. HTML documents follow a hierarchical structure known as the Document Object Model (DOM), allowing for easy manipulation and interaction with webpage elements using scripting languages like JavaScript. HTML separates content from presentation, providing semantic meaning to different parts of a webpage. It supports multimedia elements and allows integration with stylesheets (CSS) and scripts (JavaScript). HTML is essential for web development, enabling the creation of accessible and interactive webpages.

2. **CSS**:(Cascading Style Sheets) is a language used to define the visual appearance and formatting of HTML documents. It works alongside HTML to separate content from presentation, allowing developers to control the layout, colors, fonts, and other visual aspects of a webpage. CSS achieves this by selecting HTML elements and applying styles to them using selectors and declarations. Selectors target specific elements or groups of elements, while declarations specify the desired styles, such as font size, background color, margins, and more. CSS provides flexibility and consistency across multiple webpages by enabling the use of external style sheets that can be shared among various HTML documents. It enhances the user experience, improves accessibility, and simplifies the maintenance of web designs.

3. **JavaScript:** JavaScript is a high-level programming language primarily used for adding interactivity and dynamic behavior to webpages. It enables developers to create responsive and interactive elements that can respond to user actions or events. JavaScript can manipulate and modify HTML and CSS elements on the fly, allowing for real-time updates and user-friendly experiences. It offers a wide range of functionalities, including form validation, animations, dynamic content loading, data manipulation, and more. JavaScript can be executed on both the client-side (in the web browser) and the server-side (with Node.js), making it versatile for various web development scenarios. It is a key component in modern web applications, powering dynamic web pages and enhancing user engagement.

4. **OpenAI:** The OpenAI API key is a unique identifier that grants access to the OpenAI GPT-3.5 language model and its capabilities. It acts as a secure authentication token that allows developers to make API requests and utilize the powerful natural language processing capabilities of GPT-3.5. The API key serves as a means to authenticate and track usage, ensuring that only authorized users can access and interact with the OpenAI model. With the API key, developers can send requests to the OpenAI API endpoint, passing in prompts or questions and receiving generated text or responses in return. It is an essential component for integrating the OpenAI language model into applications, chatbots, and other software systems, enabling developers to leverage the advanced language processing capabilities of GPT-3.5 in their own projects.

5. **React:** React is a popular JavaScript framework for building user interfaces (UI) in web applications. It allows developers to create reusable UI components that update efficiently and render dynamically based on changes in data. React follows a component-based architecture, where complex UIs are broken down into smaller, self-contained components, each managing its own state and behavior. It uses a virtual DOM (Document Object Model) to efficiently update only the necessary parts of the UI, resulting in improved performance. React promotes a declarative programming style, where developers describe the desired UI state and React takes care of updating the UI accordingly. It provides a rich ecosystem of libraries and tools, making it a versatile choice for building interactive and scalable web applications.

6.2 SOFTWARE TESTING

Software testing for the Image Generator project, developed by OpenAI, would involve a comprehensive and rigorous process to ensure the quality and reliability of the image generation model. Here is a brief explanation of software testing aspects that could be considered for this project:

1. Unit Testing: Test individual components and functions of the codebase to verify their correctness and ensure they behave as expected. This could include testing specific modules responsible for data processing, text encoding, image generation, and other core functionalities.

2. Integration Testing: Validate the interaction and integration between different modules or components of this system. This ensures that the system functions seamlessly as a whole, with proper data flow and communication between various parts of the model.

3. Functional Testing: Verify that the model performs its intended function correctly, which is to generate images based on textual prompts. Test different input scenarios and validate the generated images against expected outputs, ensuring they align with the provided prompts.

4. Performance Testing: Assess the performance of the model in terms of its speed, scalability, and resource utilization. This testing helps identify any bottlenecks, performance issues, or optimization opportunities that could enhance the efficiency of image generation.

5. Robustness Testing: Evaluate the DALL·E model's ability to handle unexpected or invalid inputs gracefully. Test edge cases, exceptional inputs, and boundary conditions to ensure the model doesn't crash or produce incorrect results, providing robustness and resilience.

6. Security Testing: Assess the DALL·E system for potential security vulnerabilities, ensuring that it doesn't expose sensitive data or have vulnerabilities that could be exploited by malicious actors. This includes testing for data privacy, access controls, and protection against potential attacks.

7. Compatibility Testing: Verify the compatibility of the system across different platforms, operating systems, and browsers, if applicable. This ensures that the model can be deployed and used effectively across a variety of environments.

8. User Acceptance Testing: Involve end users or stakeholders to test the system from a user's perspective. Collect feedback, assess user satisfaction, and ensure that the generated images meet their expectations and requirements.

9. Regression Testing: Conduct regular regression testing to ensure that new updates or modifications to the system do not introduce new bugs or regressions. This involves retesting previously validated functionalities to ensure they still work as expected.

10. Documentation Testing: Validate the accuracy and completeness of the documentation accompanying the project. This includes the user guide, API documentation, code comments, and any other relevant documentation artifacts.

Unit testing is a type of software testing that focuses on individual units of code. It is typically performed by developers, and it helps to ensure that individual units of code are working as expected. Integration testing is a type of software testing that focuses on how individual units of code interact with each other. It is typically performed by testers, and it helps to ensure that the different parts of a software system work together as expected. System testing is a type of software testing that focuses on the entire software system. It is typically performed by testers, and it helps to ensure that the software system meets all of its requirements. Acceptance testing is a type of software testing that is performed by the customer or end user. It is typically performed after the software system has been released, and it helps to ensure that the software system meets the customer's needs.

API (Application Programming Interface) is one of the main components of any software systems today. APIs power most of the mobile, desktop and web applications. Software companies connect their SaaS products with other applications using APIs to offer more features. APIs became so popular that some companies started selling APIs as a product to customers. APIs use a concept called **API Key** to identify the customer who is using their API. In this post, I will demonstrate the purpose of API Keys and how to use them.

What is API Key? As discussed in the APIs are computer software programs without a GUI (Graphical User Interface). Most APIs are protected; to use them, one should authenticate (log in) first, similar to your online bank account that requires logging in before using the online banking service. There are multiple ways to do the authentication & authorisation in APIs; leveraging an approach known as API Key is one of the simplest yet efficient ways to do it. API Key is issued by API vendor and has the following characteristics:

Often API Keys are an encrypted/randomly generated sequence of characters. The number of characters varies depending on the vendor, but often API keys are not shorter than 20 characters.

It’s virtually impossible to guess an API Key due to being a long encrypted/random text.

Your API Key is unique to you. The API vendor does not issue your API key to anyone else.

API keys are encrypted/randomly generated sequence of characters generated by an API vendor. API Keys are used to authorizing requests to an API.

Some APIs provide a sandbox version of their service, a playground for developers to start using an API without impacting customer data. Sandbox API Key is an API key used to access a sandbox API.

What API key should be used for testing purposes? It's best practice to have dedicated API keys for testing and usage in production to minimise the risk of data loss, bugs or undesired behaviour. Most API vendors offer sandbox environments to test APIs, and we recommend leveraging sandbox APIs for testing purposes.
What disaster recovery plan means for an API Key? Disaster, as far as it's related to an API key, means your API Key is available to unauthorised people. It's essential to be able to quickly revoke an API key as soon as it's leaked. Moreover, you need to update your system ASAP with a new API key to minimise system interruption.

Every time we use a protected API (sending requests to the API via code or an API Client), you must pass your API Key in your request to the API. Documentation sites for APIs often have a section called "Authentication & Authorization", which explains how you should provide the API Key. In general, there are three main ways to pass on an API key to an API:
Include the API Key in the URL (as a query string parameter)
Embed as a Request Header
Embed in Request Body

Where to get API Key? API vendors are the authority to issue API Keys for users of an API. Depending on the vendor, you can use one of the following ways to get your API key:
When a developer portal is provided to you by the API vendor, use the Developer Portal (sometimes called developer site) to get your API key on demand.
Log in to your online dashboard with the vendor and look for the Settings section. Often there is an option to generate an API key on demand.
Some vendors issue API keys manually. Request your vendor to issue an API key for you.

API Key Best Practices API Keys are sensitive information, giving access to potentially sensitive data using an API, so it's essential to follow the below best practices when using them.
Always use the HTTPS protocol to use APIs and pass API Keys. Never integrate with an API that offers HTTP as it's not secure and puts you at security risks.
Don't add API Keys to your code repositories like Github, Git Lab or Bitbucket.

Use a back-end service to integrate with an external API that requires API Key. Mobile & Web

developers should not use APIs that require an API Key directly but via an internal back-end service.
Make API Key available to your code using one of the known solutions for config management.
Make sure the log files of your servers do not have the API Keys.
Check with your team about the time it takes to update an API key in your running system. You should be able to easily update the API key to a new one in case of API Key leak emergencies. (More in disaster recovery plan)
Practice generating a new API key regularly and using the new key for integration.
Exchanging API keys via email and messaging apps like Slack is not safe. Consider using a tool that is built for transferring sensitive information.

POSTMAN API TESTING

Postman is one of the most popular software testing tools which is used for API testing. With the help of this tool, developers can easily create, test, share, and document APIs.

This tutorial will help in understanding why Postman is so famous and what makes it unique when compared to other API testing tools. All the examples in this tutorial are tested and can be imported in Postman.

Postman is a standalone software testing API (Application Programming Interface) platform to build, test, design, modify, and document APIs. It is a simple Graphic User Interface for sending and viewing HTTP requests and responses.
While using Postman, for testing purposes, one doesn't need to write any HTTP client network code. Instead, we build test suites called collections and let Postman interact with the API.
In this tool, nearly any functionality that any developer may need is embedded. This tool has the ability to make various types of HTTP requests like GET, POST, PUT, PATCH, and convert the API to code for languages like JavaScript and Python.

API
Application Programming Interface (API) is software that acts as an intermediary for two apps to communicate with each other. We use APIs whenever we use an application like Twitter, Facebook, sending text messages, or checking the weather over the phone.

HTTP
HTTP (Hypertext Transfer Protocol) is the collection of rules for the transmission of data on the World Wide Web, like graphic images, text, video, sound, and other multimedia data. The Web users implicitly make use of HTTP as soon as they open their Web browser.

Example: A user or browser enters the HTTP request to the server; the server then returns the user response. This response includes the request status information and may consist of the requested material as well.

The most commonly used HTTP methods are GET, POST, PUT, PATCH, HEAD, DELETE, and OPTIONS.

Why use Postman?
Postman is based on a wide range of extremely user-friendly power tools. For more than 8 million users,

Postman has become a tool of convenience. Following are the reasons why Postman is used:

Accessibility- One can use it anywhere after installing Postman into the device by simply logging in to the account.
Use Collections-Postman allows users to build collections for their API-calls. Every set can create multiple requests and subfolders. It will help to organize the test suites.
Test development- To test checkpoints, verification of successful HTTP response status shall be added to every API- calls.
Automation Testing-Tests can be performed in several repetitions or iterations by using the Collection Runner or Newman, which saves time for repeated tests.
Creating Environments- The design of multiple environments results in less replication of tests as one can use the same collection but for a different setting.
Debugging- To effectively debug the tests, the postman console helps to track what data is being retrieved.
Collaboration- You can import or export collections and environments to enhance the sharing of files. You may also use a direct connection to share the collections.
Continuous integration-It can support continuous integration.

Set up Postman. wr can download Postman for free from the Postman website.
Import the API definition. If you have an API definition, such as a Swagger file, you can import it into Postman. This will create a collection of requests that you can use to test your API.
Create requests. If you don't have an API definition, you can create requests manually. To do this, click on the + button in the top left corner of the Postman window and select New Request.
Configure the requests. For each request, you need to specify the following:
The HTTP method (GET, POST, PUT, DELETE, etc.)
The URL
The request body (if any)
The headers
Add tests. Once you have created the requests, you can add tests to them. Tests are JavaScript snippets that are executed after the request is sent. You can use tests to verify that the response is correct.
Run the tests. To run the tests, click on the Send button. Postman will send the requests and execute the tests.
View the results. The results of the tests will be displayed in the Test Results tab.
Here are some additional tips for API testing using Postman:

Use variables to make your tests more reusable.

Use the Pre-request Script and Post-request Script to automate common tasks.
Use the Collection Runner to run multiple requests and tests in a single batch.
Use the Monitors to track the performance of your API over time.
API testing is an essential part of the software development process. By using Postman, you can easily create and run tests to ensure that your APIs are working as expected.

To run a test collection for an API, do the following:

Select APIs in the sidebar and select an API.
Select Test and Automation.
Under Collections, select Runner icon Run next to a test collection.
Select any configuration options for the collection run, then select Run API Tests. Learn more about using the Collection Runner.
To view detailed test results, expand the collection and select View Report next to a test run.

6.3 SOURCE CODE

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
  <head>

    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="css/style.css" />
    <link rel="stylesheet" href="css/spinner.css" />

    <script src="js/main.js" defer></script>
    <title>Image Genrator Using OPENAI</title>
  </head>
  <body>
    <header>
      <div class="navbar">
        <div class="logo">
          <h2>AI Image Genrator</h2>
        </div>
        <div class="nav-links">
          <ul>
            <li>
              <a href="https://beta.openai.com/docs" target="_blank"
                >OpenAI API Docs</a
              >
            </li>
          </ul>
        </div>
      </div>
    </header>

    <main>
      <section class="showcase">
```

```
      <form id="image-form">

        <h1>Describe An Image</h1>
        <div class="form-control">
          <input type="text" id="prompt" placeholder="Enter Text" />
        </div>
        <!-- size -->
        <div class="form-control">

          <select name="size" id="size">
            <option value="small">Small</option>
            <option value="medium" selected>Medium</option>
            <option value="large">Large</option>
          </select>
        </div>
        <button type="submit" class="btn">Generate</button>
      </form>
    </section>

    <section class="image">
      <div class="image-container">
        <h2 class="msg"></h2>
        <img src="" alt="" id="image" />
      </div>
    </section>
  </main>

  <div class="spinner"></div>
 </body>
</html>
```

JavaScript CODE:

```
function onSubmit(e) {
e.preventDefault();

document.querySelector('.msg').textContent = '';
document.querySelector('#image').src = '';

const prompt = document.querySelector('#prompt').value;
const size = document.querySelector('#size').value;

if (prompt === '') {
  alert('Please add some text');
  return;
}

generateImageRequest(prompt, size);
}

async function generateImageRequest(prompt, size) {
try {
  showSpinner();

  const response = await fetch('/openai/generateimage', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      prompt,
      size,
    }),
  });

  if (!response.ok) {
    removeSpinner();
    throw new Error('That image could not be generated');
```

```
    }

    const data = await response.json();
    // console.log(data);

    const imageUrl = data.data;



    document.querySelector('#image').src = imageUrl;



    removeSpinner();
  } catch (error) {
    document.querySelector('.msg').textContent = error;

  }
}

function showSpinner() {
  document.querySelector('.spinner').classList.add('show');
}

function removeSpinner() {
  document.querySelector('.spinner').classList.remove('show');
}

document.querySelector('#image-form').addEventListener('submit', onSubmit);
```

OpenAIController:

```
const { Configuration, OpenAIApi } = require('openai');

const configuration = new Configuration({
   apiKey: process.env.OPENAI_API_KEY,
 });
 const openai = new OpenAIApi(configuration);

const generateImage = async (req, res) => {
const { prompt,size } = req.body;

const imageSize = size === 'small' ? '256x256' : size === 'medium' ? '512x512' : '1024x1024';



try{
   const response = await openai.createImage({

      prompt,

      n: 1,

      size: imageSize,
   });
   const imageUrl = response.data.data[0].url

   res.status(200).json({
      success: true,
      data: imageUrl
   });
} catch (error) {
   if (error.response) {
      console.log(error.response.status);
      console.log(error.response.data);
```

```
    } else {
      console.log(error.message);
    }
   res.status(400).json({
      success: false,
      error: 'the image could not be generated'
});
}
};

module.exports = { generateImage };
```

CSS CODE:

```
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;700&display=swap');

*,
*:before,
*:after {
  box-sizing: border-box;

  margin: 0;

  padding: 0;
}

:root {
  --primary-color: #eb2188;
}

body {
  font-family: 'Poppins', sans-serif;
  font-size: 16px;
  line-height: 1.5;
  color: #333;
  background: #080a52;
}

/* Navbar */
.navbar {
  background: #000000;
  color: #fff;
  padding: 1rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.navbar ul {
  list-style: none;
```

```
  display: flex;
}


.navbar ul li {
  margin-left: 1rem;
}



.navbar a {
  color: #fff;

  text-decoration: none;

  font-size: 1.2rem;
}

.navbar a:hover {
  color: var(--primary-color);
}

/* Showcase */
.showcase {
  height: 450px;
  width: 100%;
  background: var(--primary-color);
}

.showcase h1 {
  font-size: 2.5rem;
  color: #333;
  text-align: center;
  padding: 1rem 0;
}
```

```
.showcase form {
  display: flex;
  align-items: center;

  justify-content: center;
  flex-direction: column;
  height: 100%;
  width: 100%;

}

.showcase form input,
form select {

  font-size: 20px;
  width: 500px;
  max-width: 500px;

  padding: 1rem;

  border: 1px solid #ccc;

  border-radius: 5px;
  margin-bottom: 1rem;
}

form button {
  font-size: 20px;
  padding: 1rem 2rem;
  border: none;
  border-radius: 5px;
  background: #333;
  color: #fff;
  cursor: pointer;
  margin-top: 20px;
}
```

```
form button:hover {
  transform: scale(1.1);
}

/* Image */

.image {
  width: 100%;
  margin-top: 2rem;
}

img {
  display: block;
  margin: auto;
}

.msg {
  font-size: 1.6rem;
  text-align: center;
  margin-top: 1rem;
}
```

CSS SPINNER

```
/* Spinner From: https://codepen.io/tbrownvisuals/pen/edGYvx */
.spinner {
    position: fixed;
    z-index: 999;
    height: 2em;
    width: 2em;
    overflow: show;
    margin: auto;
    top: 0;
    left: 0;
    bottom: 0;
    right: 0;
    display: none;
  }

  .show {
    display: block;
  }

  /* Transparent Overlay */
  .spinner:before {
    content: '';
    display: block;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: radial-gradient(rgba(20, 20, 20, 0.8), rgba(0, 0, 0, 0.8));

    background: -webkit-radial-gradient(
      rgba(20, 20, 20, 0.8),
      rgba(0, 0, 0, 0.8)
    );
  }
```

```
/* :not(:required) hides these rules from IE9 and below */
.spinner:not(:required) {
  /* hide "loading..." text */
  font: 0/0 a;
  color: transparent;
  text-shadow: none;
  background-color: transparent;
  border: 0;
}

.spinner:not(:required):after {
  content: '';
  display: block;
  font-size: 10px;
  width: 1em;
  height: 1em;
  margin-top: -0.5em;
  -webkit-animation: spinner 150ms infinite linear;
  -moz-animation: spinner 150ms infinite linear;
  -ms-animation: spinner 150ms infinite linear;
  -o-animation: spinner 150ms infinite linear;
  animation: spinner 150ms infinite linear;
  border-radius: 0.5em;
  -webkit-box-shadow: rgba(255, 255, 255, 0.75) 1.5em 0 0 0,
    rgba(255, 255, 255, 0.75) 1.1em 1.1em 0 0,
    rgba(255, 255, 255, 0.75) 0 1.5em 0 0,
    rgba(255, 255, 255, 0.75) -1.1em 1.1em 0 0,
    rgba(255, 255, 255, 0.75) -1.5em 0 0 0,
    rgba(255, 255, 255, 0.75) -1.1em -1.1em 0 0,
    rgba(255, 255, 255, 0.75) 0 -1.5em 0 0,
    rgba(255, 255, 255, 0.75) 1.1em -1.1em 0 0;
  box-shadow: rgba(255, 255, 255, 0.75) 1.5em 0 0 0,
    rgba(255, 255, 255, 0.75) 1.1em 1.1em 0 0,
    rgba(255, 255, 255, 0.75) 0 1.5em 0 0,
    rgba(255, 255, 255, 0.75) -1.1em 1.1em 0 0,
    rgba(255, 255, 255, 0.75) -1.5em 0 0 0,
```

```
    rgba(255, 255, 255, 0.75) -1.1em -1.1em 0 0,
    rgba(255, 255, 255, 0.75) 0 -1.5em 0 0,
    rgba(255, 255, 255, 0.75) 1.1em -1.1em 0 0;
}

/* Animation */

@-webkit-keyframes spinner {
  0% {
    -webkit-transform: rotate(0deg);
    -moz-transform: rotate(0deg);
    -ms-transform: rotate(0deg);
    -o-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(360deg);
    -moz-transform: rotate(360deg);
    -ms-transform: rotate(360deg);
    -o-transform: rotate(360deg);
    transform: rotate(360deg);
  }
}
@-moz-keyframes spinner {
  0% {
    -webkit-transform: rotate(0deg);
    -moz-transform: rotate(0deg);
    -ms-transform: rotate(0deg);
    -o-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(360deg);
    -moz-transform: rotate(360deg);
    -ms-transform: rotate(360deg);
    -o-transform: rotate(360deg);
    transform: rotate(360deg);
```

```
  }
}
@-o-keyframes spinner {
  0% {
    -webkit-transform: rotate(0deg);
    -moz-transform: rotate(0deg);
    -ms-transform: rotate(0deg);
    -o-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(360deg);
    -moz-transform: rotate(360deg);
    -ms-transform: rotate(360deg);
    -o-transform: rotate(360deg);
    transform: rotate(360deg);
  }
}
@keyframes spinner {
  0% {
    -webkit-transform: rotate(0deg);
    -moz-transform: rotate(0deg);
    -ms-transform: rotate(0deg);
    -o-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(360deg);
    -moz-transform: rotate(360deg);
    -ms-transform: rotate(360deg);
    -o-transform: rotate(360deg);
    transform: rotate(360deg);
  }
}
```

CHAPTER 7

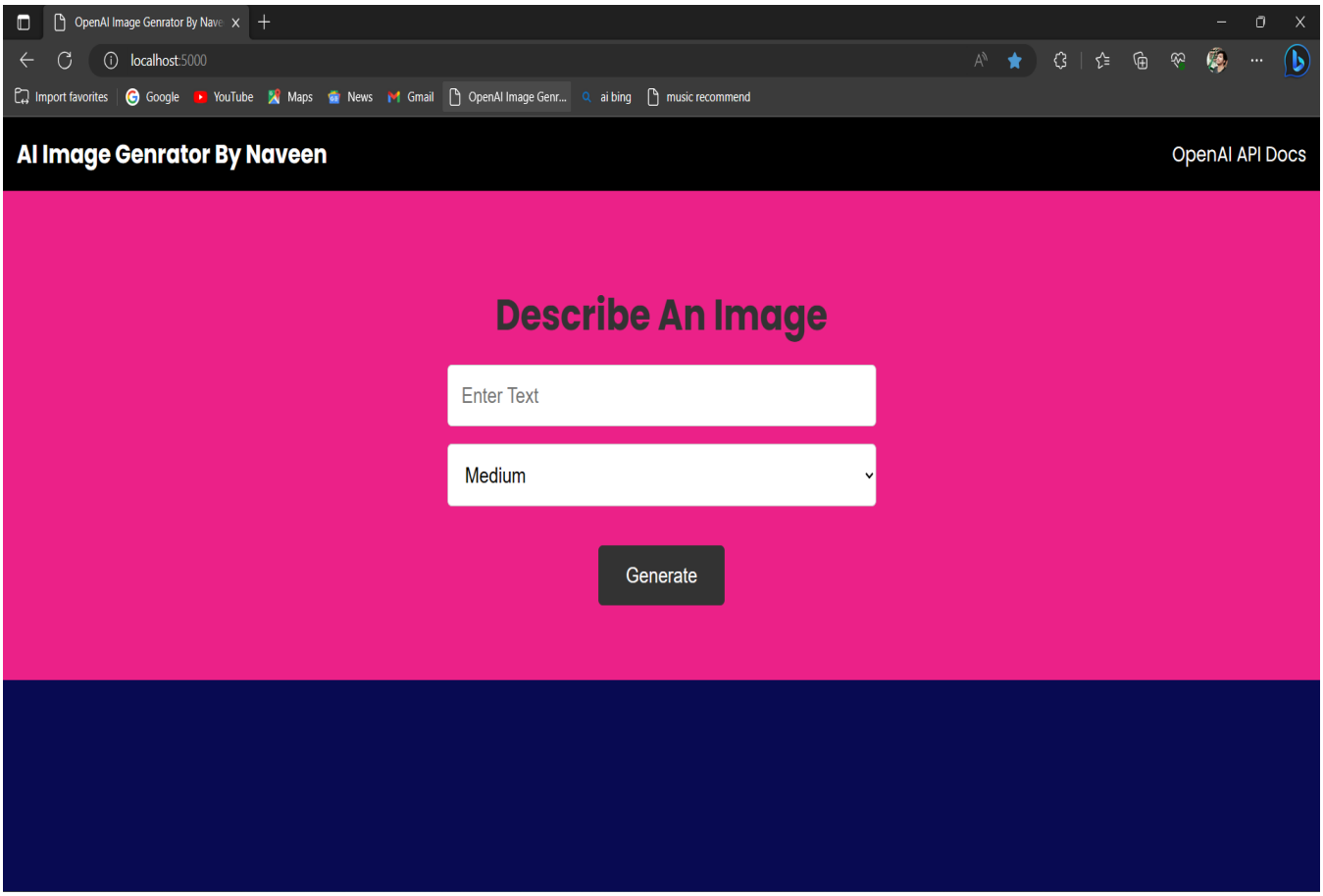
TESTING

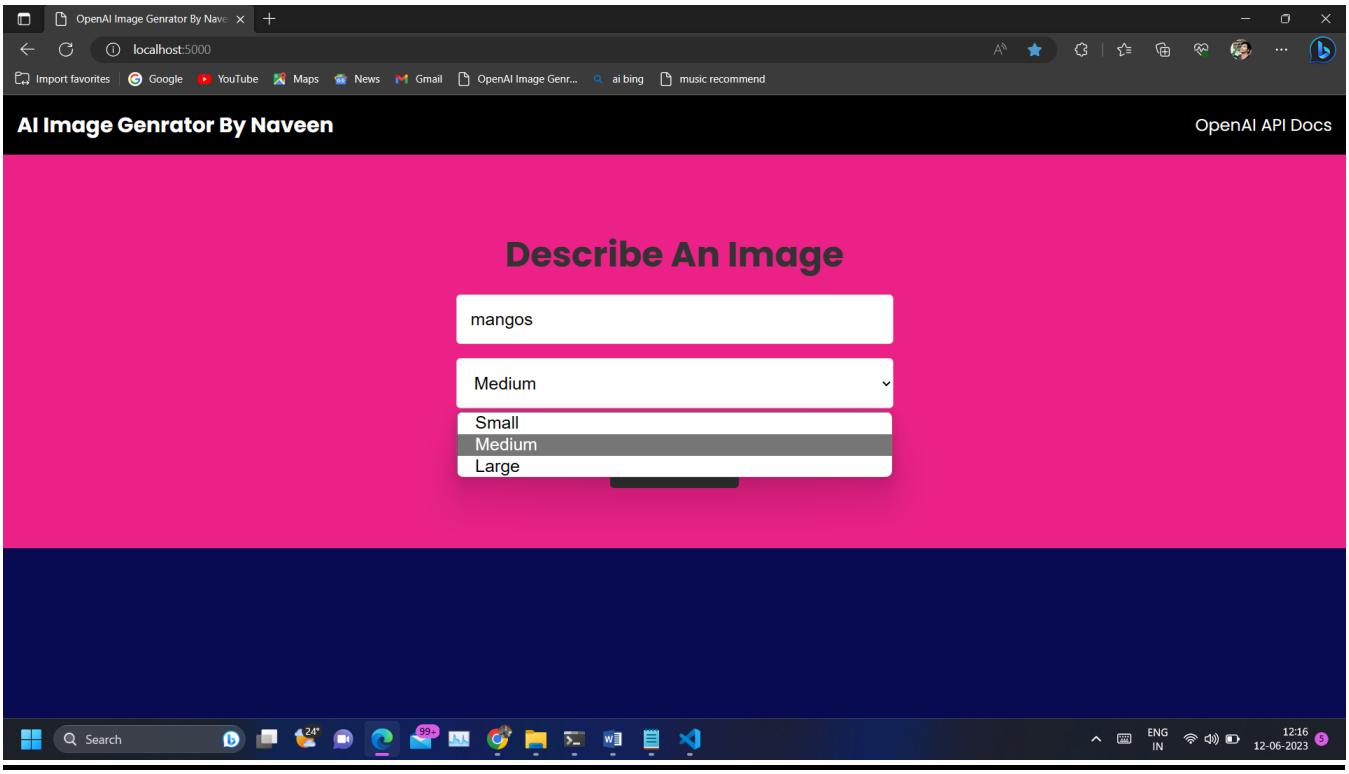
Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements. Some prefer saying Software testing definition as a White Box and Black Box Testing. In simple terms, Software Testing means the Verification of Application Under Test (AUT). This Software Testing course introduces testing software to the audience and justifies the importance of software testing.

Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

Image Generated by OpenAI

Page interface




OpenAI Image Genrator By Nave
localhost:5000
Import favorites
Google
YouTube
Maps
News
Gmail
OpenAI Image Genr...
ai bing
music recommend
AI Image Genrator By Naveen
OpenAI API Docs
Describe An Image
mangos
Medium
Small
Medium
Large
Search
ENG
IN
12:16
12-06-2023

CHAPTER 8
CONCLUSION

In conclusion, Image Generator project has revolutionized the fields of computer vision and natural language processing, showcasing the incredible potential of AI in generating visual content from textual descriptions. Through its groundbreaking capabilities, OpenAI has transformed the creative process, empowering artists, designers, and content creators to bring their ideas to life with ease and efficiency.

Moreover, it has extended its impact beyond the realm of art and design. Its applications in e-commerce have the potential to reshape online shopping experiences by providing realistic product visualizations, bridging the gap between physical and digital retail environments. In the medical field, it has ability to generate medical images and aid in diagnostics has the potential to enhance healthcare practices and improve patient outcomes.

While the benefits of this module are undeniable, ethical considerations must be at the forefront of its development and deployment. Guarding against copyright infringement and addressing the risks associated with deepfakes and deceptive content is crucial. Regulations and responsible use guidelines must be established to ensure the technology is employed in a manner that aligns with societal values and safeguards against misuse.

Challenges lie ahead for module, such as refining the model's prompt understanding and addressing biases in generated outputs. Ongoing research and collaboration will be necessary to overcome these obstacles and further enhance the model's capabilities. Additionally, efforts to diversify the training data and improve the realism and contextual understanding of the generated images will be key to advancing the technology.

Looking to the future, it holds tremendous promise. Continued research and development can lead to even more advanced iterations of the model, enabling it to generate images with unparalleled detail, realism, and context sensitivity. The collaboration between researchers, artists, and industry professionals will be instrumental in pushing the boundaries of OpenAI applications and exploring new frontiers.

Open-source initiatives and community involvement will play a vital role in the evolution of OpenAI. Embracing a collective approach to its development will ensure a wide range of perspectives are considered, fostering innovation and inclusivity in the technology's use. By combining human creativity

with AI capabilities, OpenAI has the potential to unlock groundbreaking solutions and drive transformative change across various domains.

In summary, the ImageGenerator project has introduced a remarkable breakthrough in AI-driven image generation. Its impact spans creative industries, e-commerce, healthcare, and beyond. With responsible development and deployment, it can enhance human creativity, improve user experiences, and revolutionize how we interact with visual content. As we venture into a future shaped by AI, the journey of it represents an inspiring fusion of human imagination and machine intelligence, paving the way for a new era of visual expression and innovation.

REFERENCES

https://labs.openai.com/