

# Develop a Reliable Backend with Node and Express



# Getting Started With APIs



# A Day in the Life of a MERN Stack Developer

You are working as a MERN stack developer for an organization.

Your company is incorporating the OpenAI API into its products to enhance natural language processing capabilities.

To integrate the OpenAI API into the company's infrastructure, you need to understand the following:

- Integration of APIs into web applications
- The advantages and disadvantages of using APIs
- The prerequisites for using the Open AI API and its billing model



# A Day in the Life of a MERN Stack Developer

To implement these requirements, you must keep up with the best practices and stay informed about any updates or changes in the OpenAI API documentation to deliver an efficient solution for your organization.

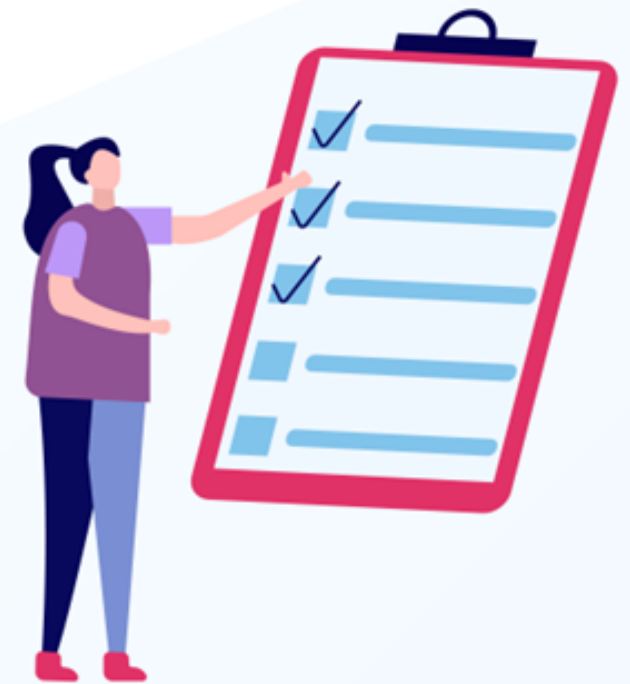
To achieve the above, you will learn a few concepts in this lesson that can help you find a solution for the scenario.



# Learning Objectives

By the end of this lesson, you will be able to:

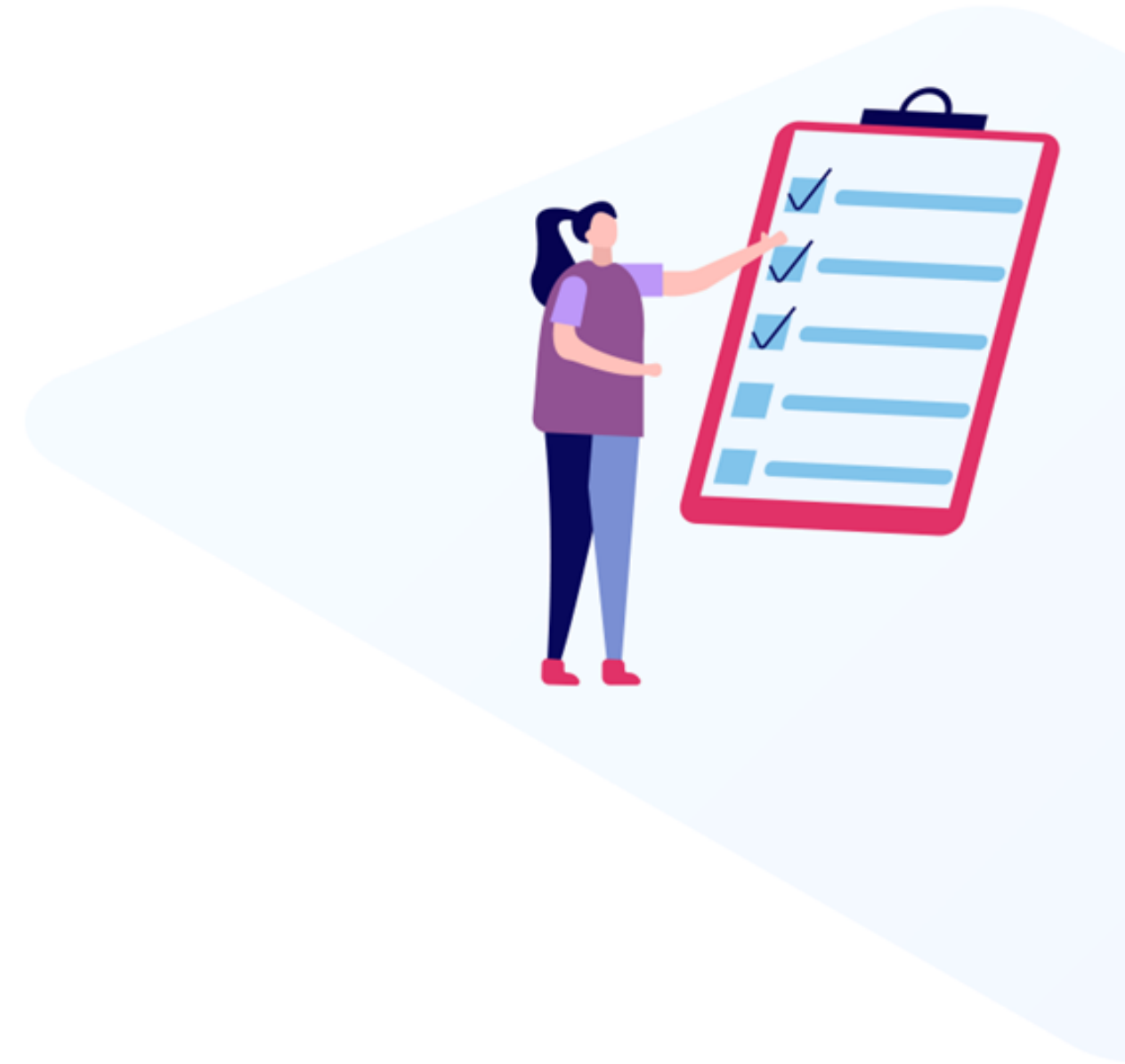
- Grasp the concepts of APIs and integrate them into the projects to facilitate communication between software components
- Differentiate between REST and SOAP APIs to make informed design based on the project's requirements
- Identify the use of API testing to validate the functionality and performance of APIs in software development
- Work with OpenAI API concepts to enhance content generation, user interactions, and overall functionality



# Learning Objectives

By the end of this lesson, you will be able to:

- 👁️ Identify the use of handling large text inputs for the practical application of natural language processing (NLP) models
- 👁️ Grasp the concept of prompt engineering for effective use of the NLP models
- 👁️ Define the use of obtaining an API key to integrate OpenAI's language models into applications



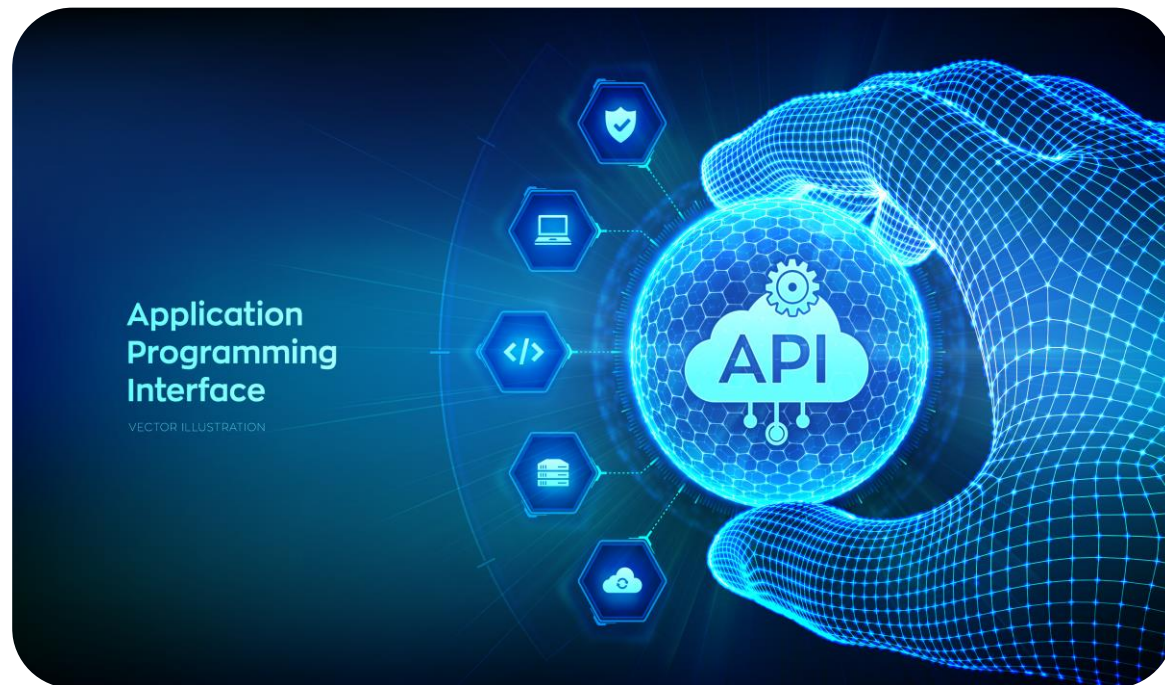


# Introduction to *API*



# API: Overview

An application programming interface serves as a set of defined rules and tools that allow different software applications to communicate and interact with each other.



- It facilitates communication between different software systems, allowing them to request and exchange data.
- It is integral to build scalable and modular software systems.



# How Do APIs Work?

Here is an overview of the working of APIs:

## Request

The client sends a request to the API specifying the desired operation or data.

## API endpoint

The client's request includes the endpoint that corresponds to the desired operation.

## HTTP methods

APIs often Use methods like GET, POST, and PUT to define the operation type the client is requesting.

# How Do APIs Work?

## Request processing

The server might perform actions or carry out other operations based on the client's request.

## Authentication

Clients need to provide authentication tokens with their requests to verify their identities.

## Authorization

After authentication, the API checks whether the authenticated user or client has the necessary permissions (authorization) to perform the requested operation.

# How Do APIs Work?

## Data transfer

The API sends a response to the client if the request is valid and authorized.

## Data format

APIs use standard data formats for communication, such as JSON or XML.

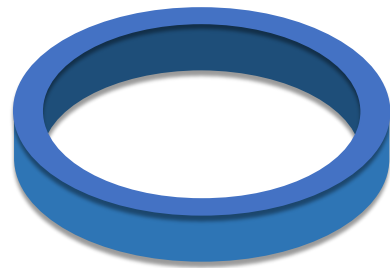
## Client integration

The client application processes the API response and integrates the received data into its user interface.

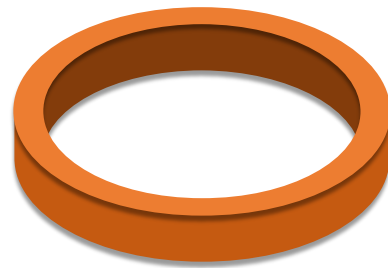
# APIs: Types

There are several types of APIs, each serving specific purposes and use cases.

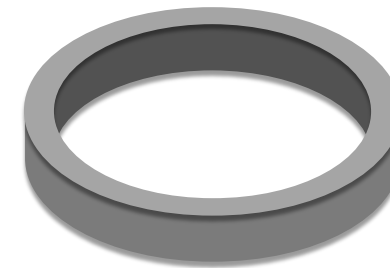
Here are some common types of APIs:



Open APIs



Internal APIs

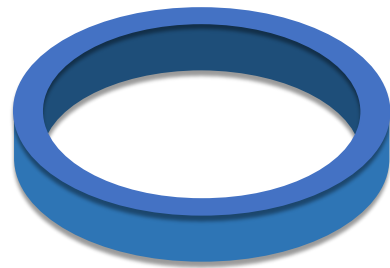


RESTful APIs

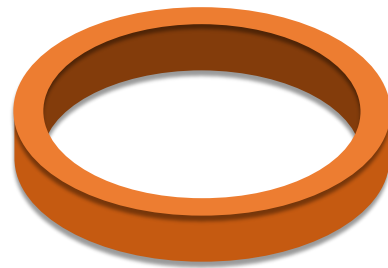
# APIs: Types

There are several types of APIs, each serving specific purposes and use cases.

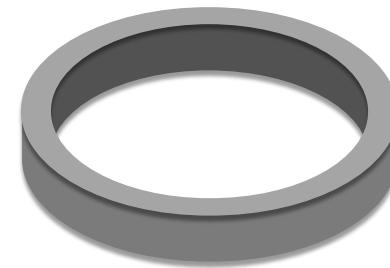
Here are some common types of APIs:



SOAP APIs



Web APIs



Database APIs

# What Are REST APIs?

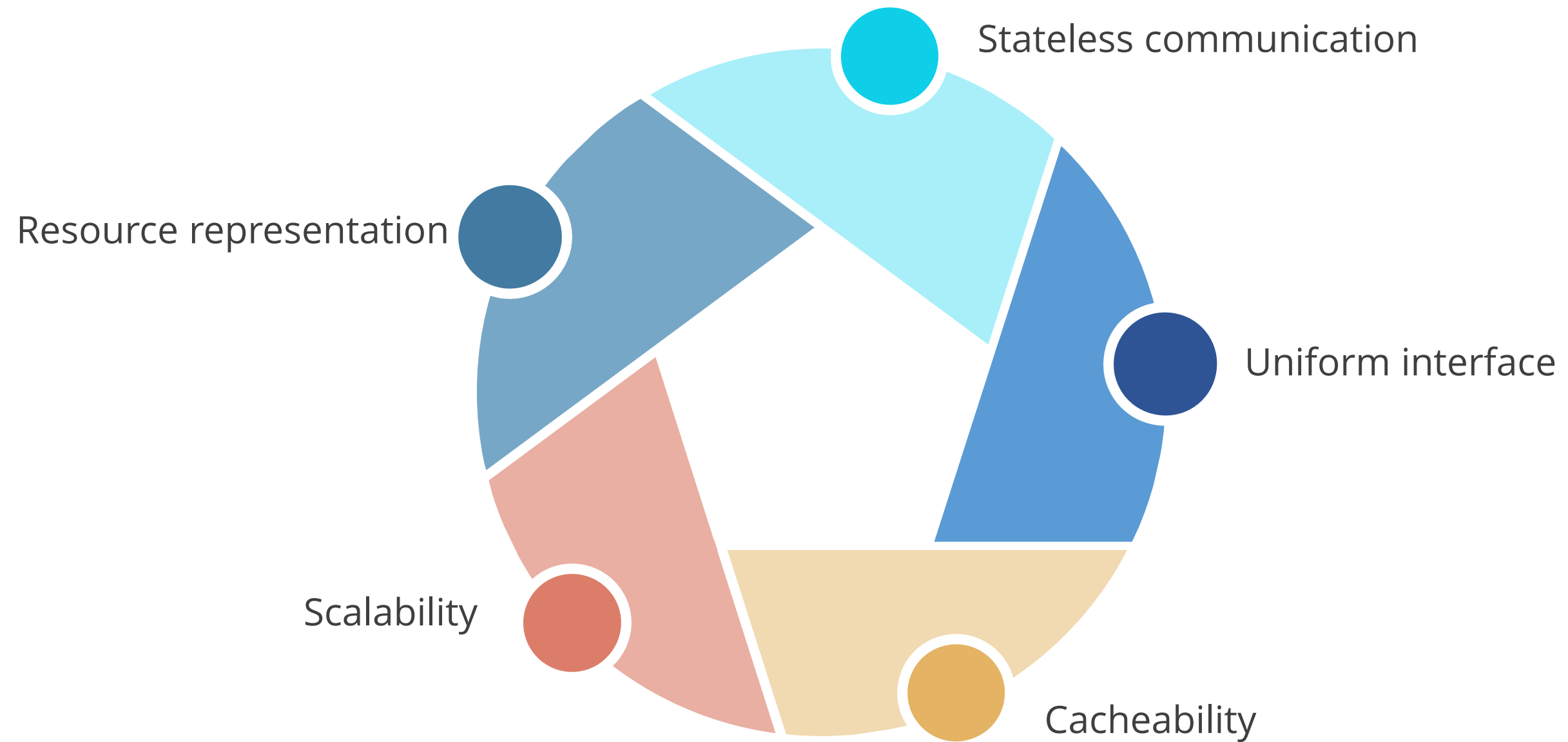
**Representational State Transfer** (REST) APIs follow the principles of REST, which is an architectural style for designing networked applications.



REST APIs use standard HTTP methods (GET, POST, PUT, and DELETE) for communication and are widely used in web development.

# REST API: Features

The REST APIs have several features, including:





# What Is a Web API?

A web API is a set of rules and protocols that allows one software application to interact with another over the web.



It defines the methods and data formats that applications can use to request and exchange information.

# Web API: Features

Web APIs have several features, including:



# REST API vs. SOAP API

Below are the differences between REST API and SOAP API:

Feature	REST API	SOAP API
Architecture	Stateless, follows a resource-based model	Stateful, follows a service-oriented model
Protocol	Uses HTTP primarily	Uses HTTP but can work with other protocols also
Communication	Client-server communication	Both synchronous and asynchronous communication
Performance	Faster due to lightweight messages	Slower due to XML overhead and additional processing

# What Is API Integration?

It is a fundamental aspect of modern software development, enabling organizations to create interconnected and interoperable systems.



- It streamlines processes, enhances automation, and improves overall system efficiency.
- It involves making HTTP requests to specific endpoints.

# API Integration: Benefits

01

API integration ensures that data is accurately and consistently shared between systems.

02

It supports scalability by allowing systems to grow without major disruptions.

03

API integration facilitates real-time communication and updates.

## API Integration: Benefits

04

API integration enables businesses to connect with a broader ecosystem of partners, suppliers, and third-party services.

05

It provides flexibility, allowing developers to choose the best tools and services for their specific needs.

06

It empowers organizations to create more powerful and connected solutions.

# What Is API Testing?

API testing is critical for ensuring the reliability and performance of applications, especially in the context of modern, interconnected software ecosystems.



It helps identify and address issues early in the development process, contributing to the overall quality and stability of the software.



# How to Create APIs?

Users can follow a few steps to create APIs, which are:

- 1 Define the purpose and scope
- 2 Choose a programming language
- 3 Design the API
- 4 Implement the API logic
- 5 Implement error handling

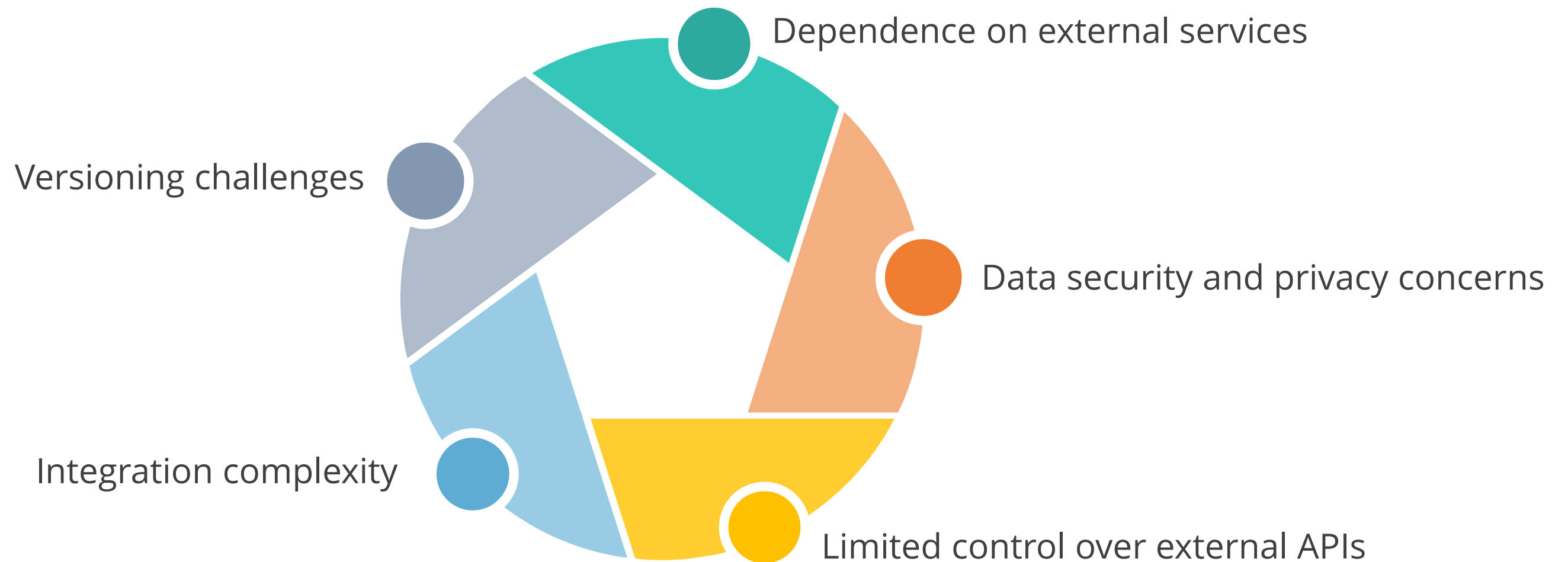
# How to Create APIs?

Users can follow a few steps to create APIs, which are:

- 6 Test the API
- 7 Document the API
- 8 Secure the API
- 9 Deploy the API
- 10 Monitor the API performance

# Restrictions of Using APIs

While APIs provide numerous advantages, they also come with specific limitations and challenges, which are:



# APIs: Advantages

APIs offer several advantages, including:



- Rapid development
- Flexibility
- Enhanced user experience
- Scalability

# APIs: Disadvantages

APIs has various disadvantages, including:



- Security concerns
- Documentation challenges
- Limited customization options
- Versioning issues



# Introduction to OpenAI API

# OpenAI API: Overview

The OpenAI API refers to the application programming interface provided by OpenAI, a leading artificial intelligence research laboratory.

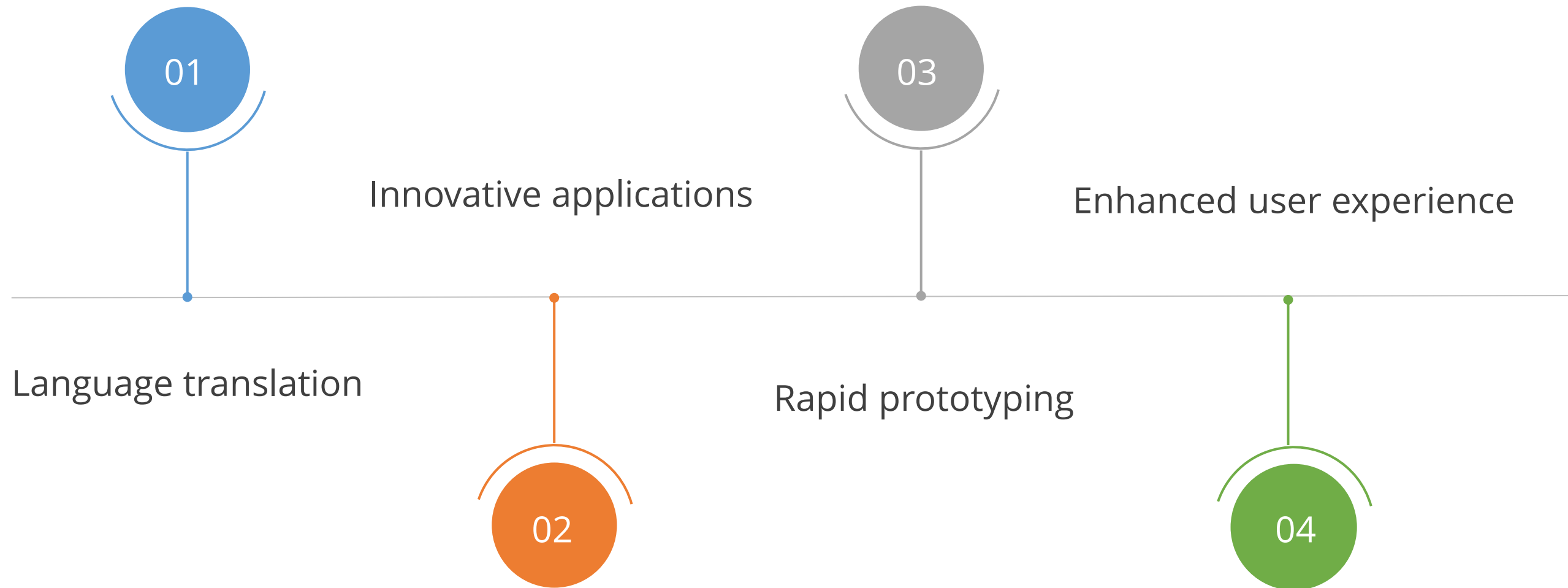


- It is designed to perform various natural language processing tasks, including text completion, language translation, and more.
- Developers can interact with the OpenAI API by sending HTTP requests with appropriate parameters and prompts.







# Why OpenAI API?

OpenAI API is crucial for many reasons, including:



# OpenAI API: Prerequisites

Below are some prerequisites that are associated with using the OpenAI API:

-  OpenAI account
-  API key
-  Development environment
-  API documentation understanding

# OpenAI API: Prerequisites

Below are some prerequisites that are associated with using the OpenAI API:



Understanding of API requests



Compliance with OpenAI policies



Security considerations



Payment information

# OpenAI API: Billing Considerations

## API key and usage tracking

To use the OpenAI API, users need an API key, which is associated with their OpenAI account.

## Pricing model

Users should review the pricing details provided by OpenAI to understand how costs are calculated.

## Cost calculation

Users should be familiar with how the usage metrics contribute to the overall cost.

## Billing cycle

OpenAI may operate on a billing cycle, charging users at regular intervals.

# OpenAI API: Billing Considerations

## Payment information

Users may need to provide payment information to cover charges associated with API usage.

## Billing dashboard

Users get a billing dashboard where they can monitor their API usage and manage their account.

## Tiered plans and features

OpenAI might offer tiered plans with different features and usage limits at varying price points.

## Terms of service

Users should carefully review OpenAI's terms of service and billing policies.

# Consuming GitHub API Using Node.js



**Duration: 15 Min.**

## **Problem Statement:**

You have been assigned a task to fetch GitHub user data in Node.js application using GitHub API

ASSISTED PRACTICE

# Assisted Practice: Guidelines



Steps to be followed:

1. Set up a Node.js project
2. Install required packages
3. Create a JavaScript script to fetch user data from GitHub





## Obtaining an OpenAI API Key

# Generating an API Key

Generating an API key for OpenAI typically involves some specific steps within your OpenAI account dashboard, which are:

- Login to your OpenAI account
- Access your account dashboard
- Locate the API key section
- Generate a new API key
- Follow the prompts

# Generating an API Key

Generating an API key for OpenAI typically involves some specific steps within your OpenAI account dashboard, which are:

- Review usage guidelines
- Confirm and generate the API key
- Copy or download the API key
- Integrate the API key into your application
- Monitor the API usage

# API Key Security

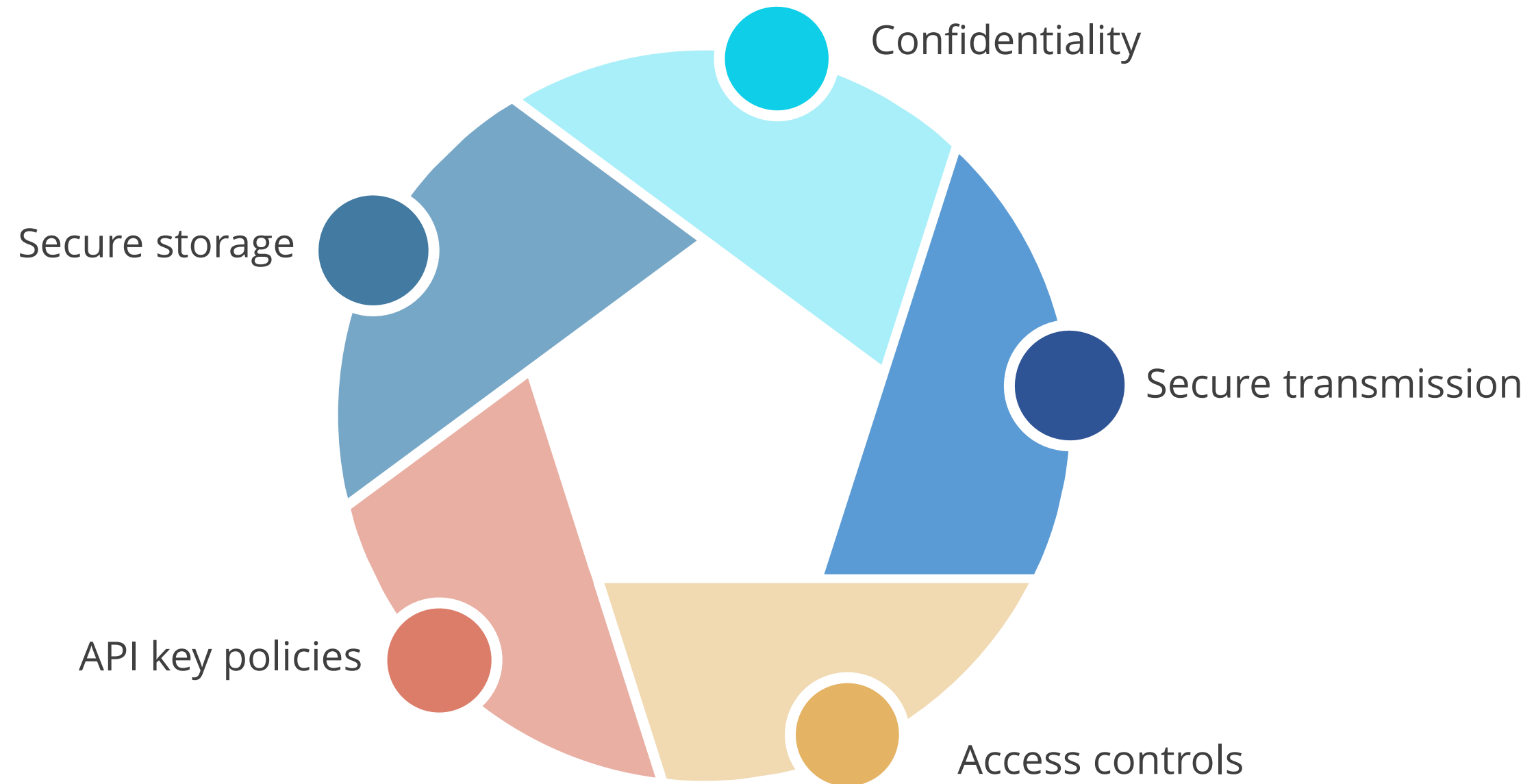
API key security is crucial to prevent unauthorized access and misuse of your application programming interface.



An API key serves as a token that authenticates and authorizes requests to your API, so protecting it is essential for maintaining the security of your system.

# API Key Security Considerations

Here are some crucial factors for securing your API keys:



# Generate API Key and Setting up OpenAPI in Node Project



**Duration: 15 Min.**

## **Problem Statement:**

You have been assigned a task to demonstrate the integration of ChatGPT-3 with Node.js, highlighting its ease and versatility for various web and data applications

# Assisted Practice: Guidelines



Steps to be followed:

1. Install the OpenAI Node.js library
2. Set up the API key
3. Send API request



# Prompt Engineering



# Customizing Prompt Necessity

The quality of the AI's output is directly influenced by the prompt it receives. Effective prompts lead to more useful and coherent responses.



# What Is Prompt Engineering?

Prompt engineering is the process of carefully crafting input prompts to elicit the desired response from an AI.



**Example:** Compare responses from the AI when given a vague versus specific prompts.

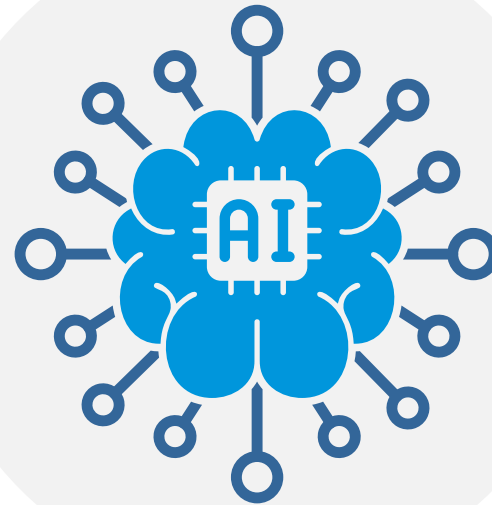
# Components of a Good Prompt

**Clarity:** Clear prompts eliminate ambiguity and clearly define the request.

**Relevance:** Relevant prompts ensure the response is directly related to the task.

**Output Format:** This aspect directs how the AI should structure its response.

**Specificity:** Specific prompts provide detailed guidance on what the response should include.



# Prompt Strategies: Iterative Refinement

**Iterative Refinement:** A process of progressively enhancing and fine-tuning prompts by incorporating insights from previous AI responses.



**Initial Prompt:** Write about space.  
**Refined Prompt after AI Response:**  
Write a detailed article about the latest Mars mission.

# Prompt Strategies: Precision and Brevity

## **Precision and Brevity:**

The art of crafting prompts that are detailed enough to guide the AI accurately



## **Example Prompt:**

Summarize the plot of **1984** by George Orwell in 50 words.

# Prompt Strategies: Feedback Loops

**Feedback Loops:** A method of continuously improving prompts by using the AI's responses as a basis for making subsequent adjustments and refinements.



**Initial AI Response:** [In response to a vague prompt about healthy eating] Eating a balanced diet is important.

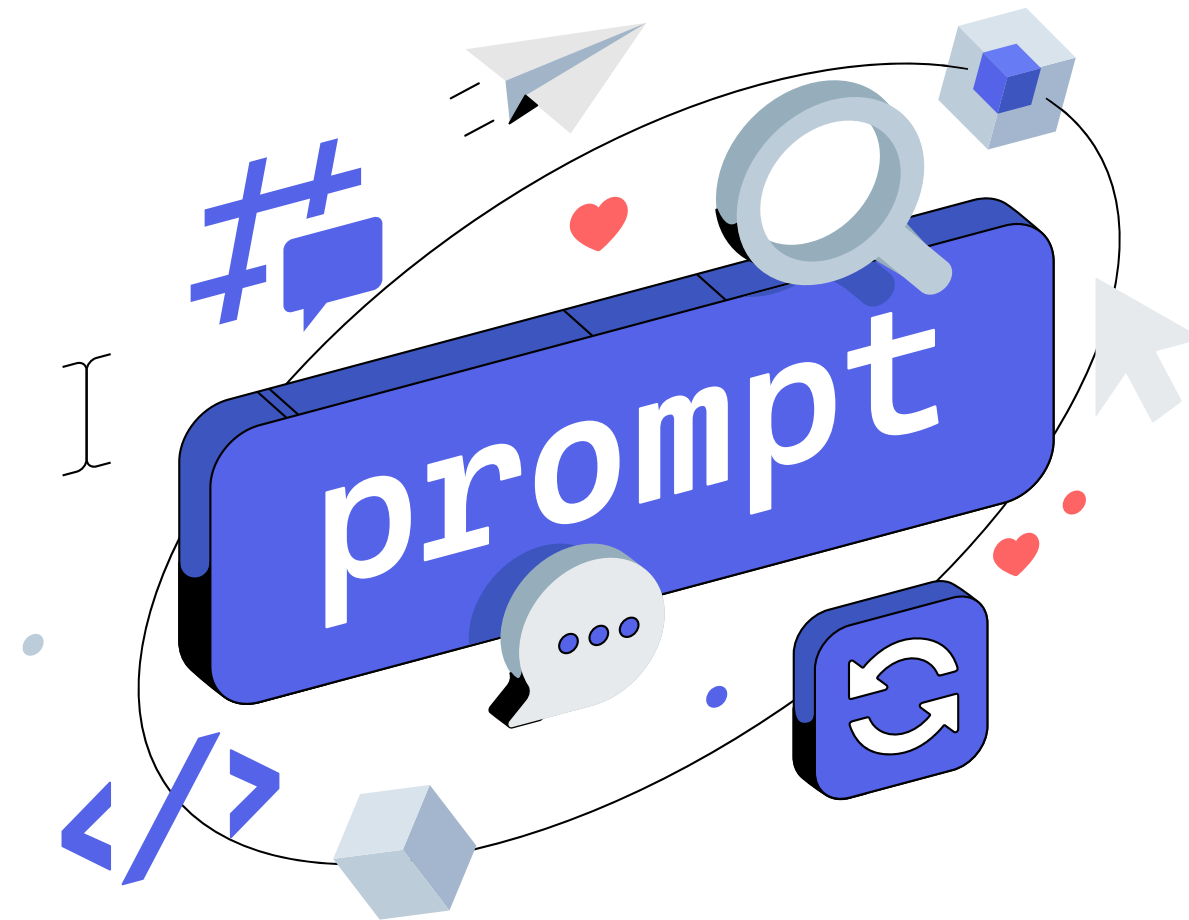
**Adjusted Prompt:** What are the five specific benefits of a diet rich in fruits and vegetables?



## Handling Large Text Inputs

# Understanding Token Limits

Tokens in the OpenAI API represent pieces of words, punctuation, or spaces, essentially the building blocks of text.

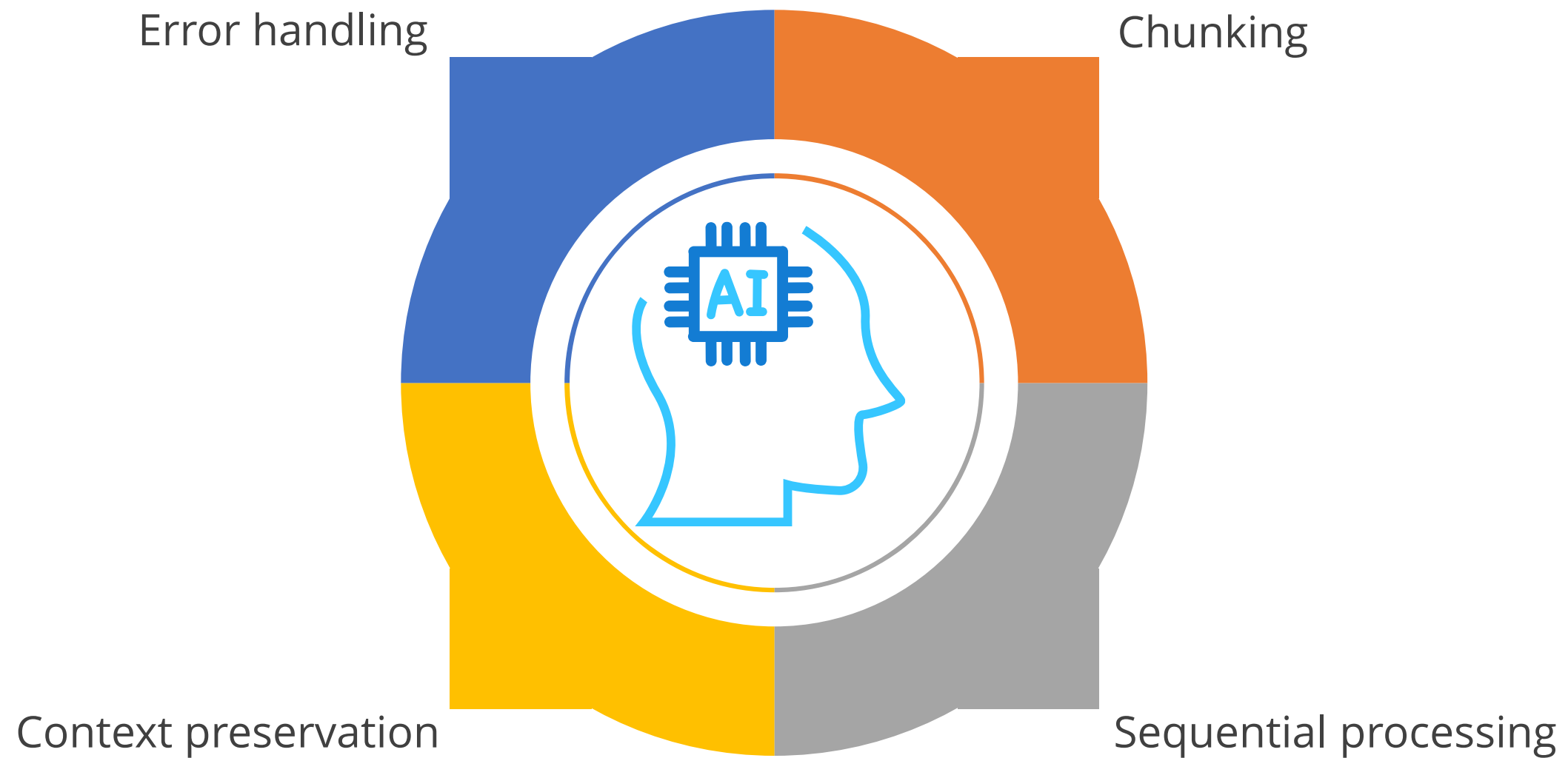


The OpenAI API limits the number of tokens per request, such as **2048** tokens, requiring large texts to be divided into smaller segments for processing.



# Strategies for Handling Large Texts

The following strategies can be used to handle large texts:



# Strategies for Handling Large Texts

**Chunking:** Breaking down large texts into smaller, manageable pieces that fit within processing limits.

```
const text = "Your very long text goes here...";
const MAX_LENGTH = 1000; // Set according to the
                           token limit of the API

function chunkText(text, maxLength) {
  let chunks = [];
  while (text.length) {
    let chunk = text.slice(0, maxLength);
    let nextIndex = chunk.lastIndexOf(' ');
    // Ensures breaking at a space
    chunk = text.slice(0, nextIndex);
    chunks.push(chunk);
    text = text.slice(nextIndex).trim();
  }
  return chunks;
}

const textChunks = chunkText(text, MAX_LENGTH);
console.log(textChunks);
```

# Strategies for Handling Large Texts

**Sequential Processing:** Handling text chunks in a specific order to maintain the narrative or logical flow.

```
// Assuming 'textChunks' is an array of text
chunks from the previous method
async function processChunksSequentially(chunks)
{
    let results = [];
    for (const chunk of chunks) {
        const result = await processChunk(chunk);
// Replace with actual API call
        results.push(result);}
    return results;}
async function processChunk(chunk) {
    // Simulating an asynchronous API call
    return new Promise(resolve => {
        setTimeout(() => resolve(`Processed:
${chunk}`), 100);
    });
}
processChunksSequentially(textChunks).then(result
s => console.log(results));
```

# Strategies for Handling Large Texts

**Context Preservation:** Ensuring that each segmented part of the text retains the overall context, which is crucial for coherent understanding and processing.

```
function preserveContext(chunks) {  
  return chunks.map((chunk, index) => {  
    if (index === 0) return chunk; // First  
    chunk remains unchanged  
    return "Recap: " + chunks[index -  
1].slice(-50) + "...\\n" + chunk;  
  });  
}  
  
const contextPreservedChunks =  
  preserveContext(textChunks);  
console.log(contextPreservedChunks);
```

# Strategies for Handling Large Texts

**Error Handling:** Implementing robust error handling strategies for scenarios where text segmentation might lead to loss of meaning or other processing issues.

```
async function processChunkWithHandling(chunk) {
  try {
    const result = await processChunk(chunk);
    // Replace with actual API call
    return result;
  } catch (error) {
    console.error("Error processing chunk:",
error);
    // Implement fallback or retry logic here
  }
}

// Use this function in the sequential processing
loop
```

# Best Practices for Handling Large Texts

- **Concise Language Usage:** Use the most succinct phrasing possible to minimize token consumption while maintaining message clarity.
- **Caching Common Responses:** Implement caching for frequently used responses to reduce redundant API calls and save resources.
- **Robust Testing Across Scenarios:** Conduct extensive testing with texts of various lengths and complexities to ensure consistent and reliable handling.
- **Continuous Monitoring and Optimization:** Regularly monitor and refine the processing system to handle diverse and evolving text input scenarios effectively.



## **Node.js and OpenAI**

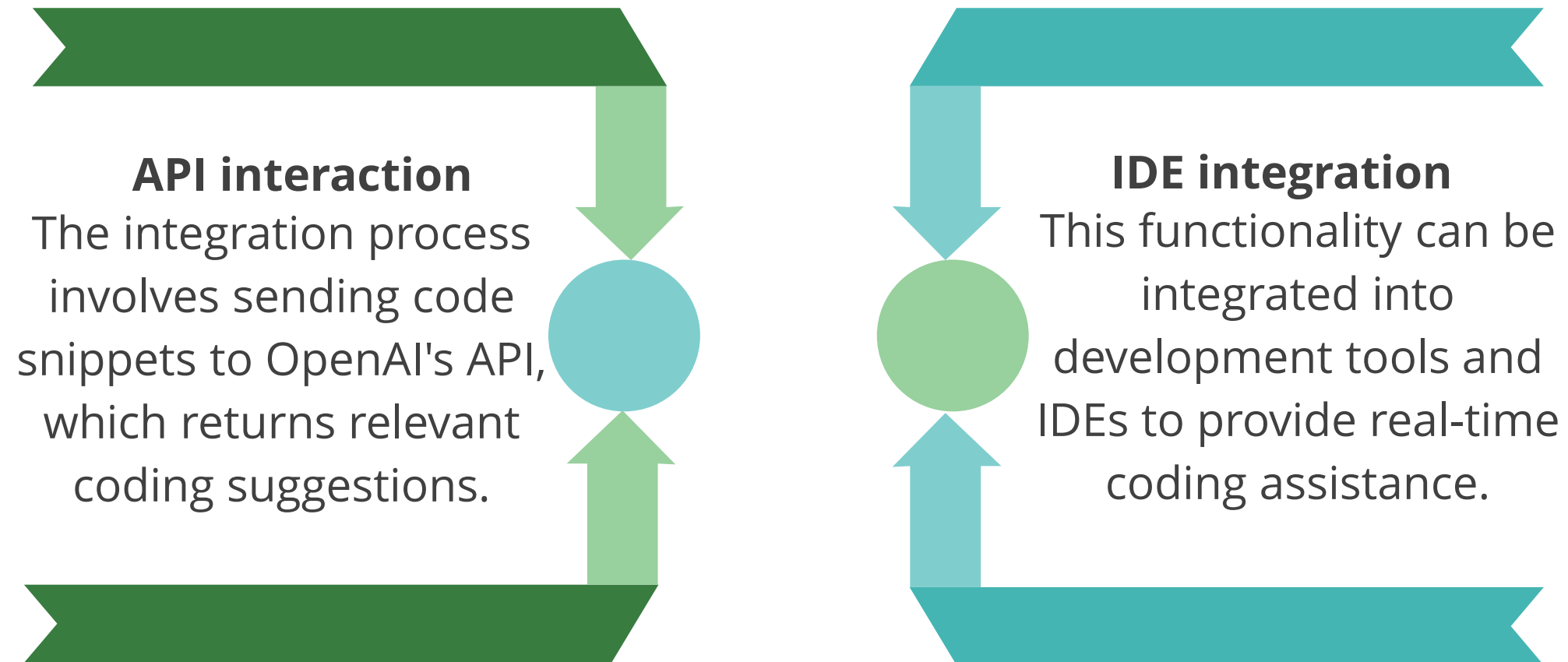
# OpenAI and Node.js Code Completion Significance

Code completion with OpenAI utilizes its AI language models to assist in writing and completing code, and this is gaining popularity due to the following reasons:

- **Efficiency:** AI accelerates development by suggesting completions for partially written code.
- **Accuracy:** The AI offers precise code suggestions by understanding context and syntax.
- **Learning aid:** AI-assisted code completion serves as an educational resource for new programmers.
- **Context-aware suggestions:** AI provides smarter suggestions by comprehending the broader context of the code, beyond just syntax.



# Ways to Implement Code Completion



# Node.js GPT Prompt Evaluator



**Duration: 15 Min.**

## **Problem Statement:**

You have been assigned a task to showcase Node.js's capability in evaluating ChatGPT-3 responses to diverse prompts

ASSISTED PRACTICE

# Assisted Practice: Guidelines



Steps to be followed:

1. List some benefits of Agile
2. Generate a React code for a homepage of a website
3. Create an Express.js web service

# Key Takeaways

- 🕒 An API serves as a set of defined rules and tools that allow different software applications to communicate and interact with each other.
- 🕒 API testing is critical for ensuring the reliability and performance of applications, especially in the context of modern, interconnected software ecosystems.
- 🕒 The OpenAI API refers to the application programming interface provided by OpenAI, a leading artificial intelligence research laboratory.
- 🕒 API integration is a fundamental aspect of modern software development, enabling organizations to create interconnected and interoperable systems.
- 🕒 API key security is crucial to prevent unauthorized access and misuse of your application programming interface.





**Thank You**