BACKEND DEVELOPMENT

Backend Development

A training report

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

(Computer Science and Engineering)

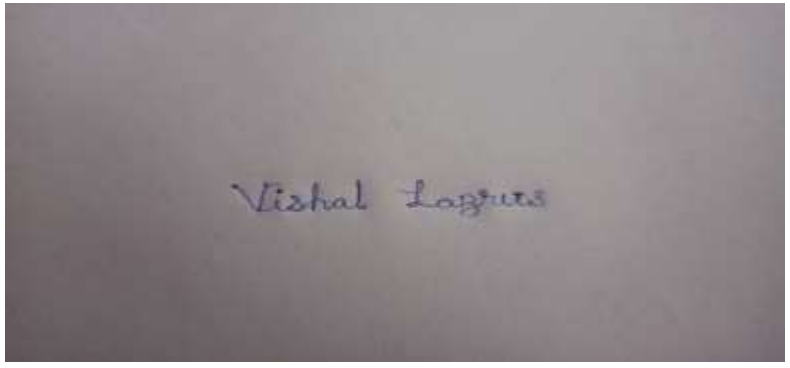
Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

Name of student: Vishal Lazrus
Registration Number: 12004235

Signature of students:



z

DECLARATION

To whom so ever it may concern

I hereby declare that I have completed my six weeks summer training at Backend

Development from the 1st of June, 2022 to 7th of July, 2022 under the guidance of Mr.Anik Acharjee. I hereby declare that I have worked with full dedication during these sixweeks of training and my learning outcomes fulfill the requirements of training for theaward of the degree of Bachelor of Technology, Lovely Professional University,

Phagwara.

ACKNOWLEDGEMENT

First of all, I would like to thank my university for giving me the golden opportunity to do ,this wonderful course name “Backend Development”.

Secondly, I would like to thank my mentor and teacher of this course Mr. Anik Acharjeewho, besides teaching in an easy-to-understand and friendly way, also solved each andevery query of mine.

Last, but not least, I would like to thank my friends and family members who always

encouraged me at each and every step in successfully completing the project.

TRAINING CERTIFICATE FROM THE ORGANIZATION

CERTIFICATE OF PARTICIPATION

THIS CERTIFICATE IS AWARDED TO

Vishal Lazrus

for successfully completing

Back end Development

7 July, 2022

ISSUED DATE

CEO, Board Infinity
Sumesh Nair

BI22LPM345425987

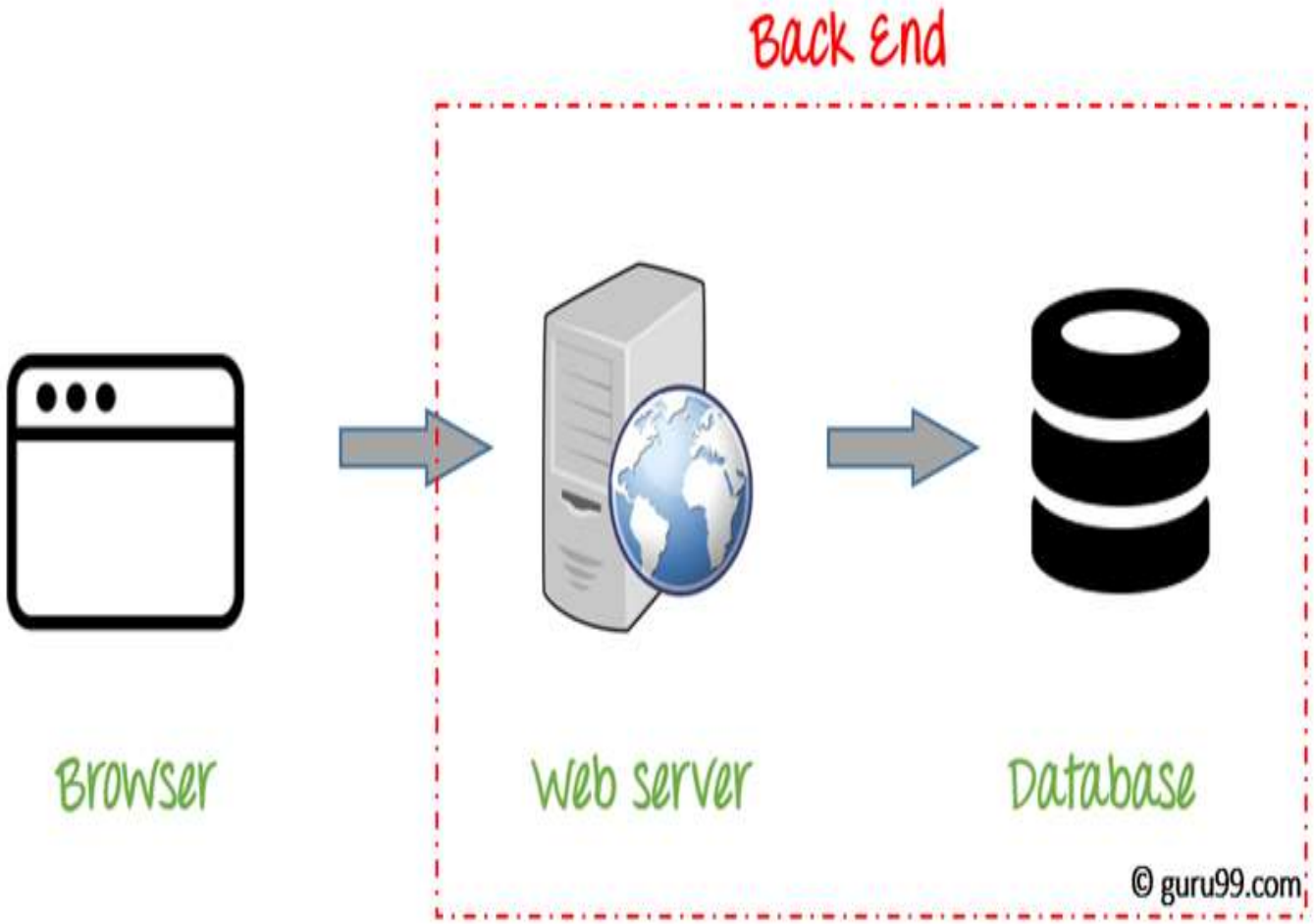
CERTIFICATE NO.

BOARD

CONTENS

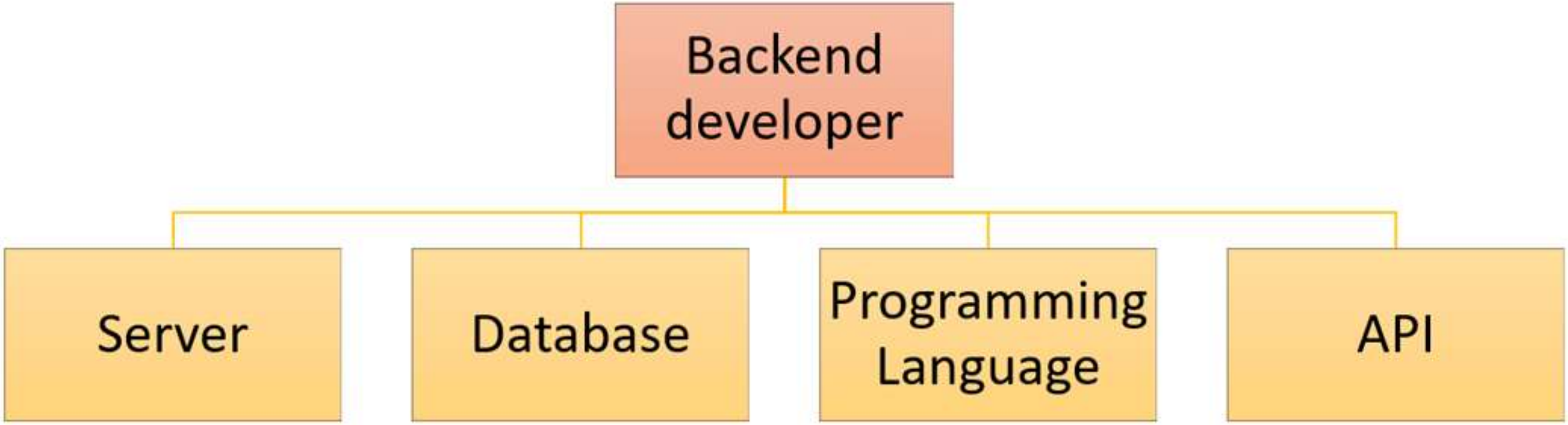
INTRODUCTION

- Back-end Development refers to the server-side development. It focuses on databases, scripting, website architecture. It contains behind-the-scene activities that occur when performing any action on a website. It can be an account login or making a purchase from an online store. Code written by back-end developers helps browsers to communicate with database information



Most common example of Backend programming is when you are reading an article on the blog. The fonts, colors, designs, etc. constitute the frontend of this page. While the content of the article is rendered from a server and fetched from a database. This is the backend part of the application.

A back-end developer builds and maintains the technology that powers those components which, together, enable the user-facing side of the website to even exist in the first place.



Following are the skills you require to become a back-end developer:

- Web Development Languages(
- Database and Cache
- Server
- API (REST & SOAP)

- Web Development Languages:

Backend engineer should know at least one server-side or Backend programming language like Java, Python, Ruby, . Net etc.

- Database and Cache:

Knowledge of various DBMS technology is one of the important Backend developer skills. MySQL, MongoDB, Oracle, SQLServer, Redis are widely used for this purpose. Knowledge of caching mechanisms like varnish, Memcached, Redis is a plus.

- Server:

Exposure to handling Apache, Nginx, IIS servers, Microsoft IIS

is desirable. A good background in Linux helps tremendously in administering servers.

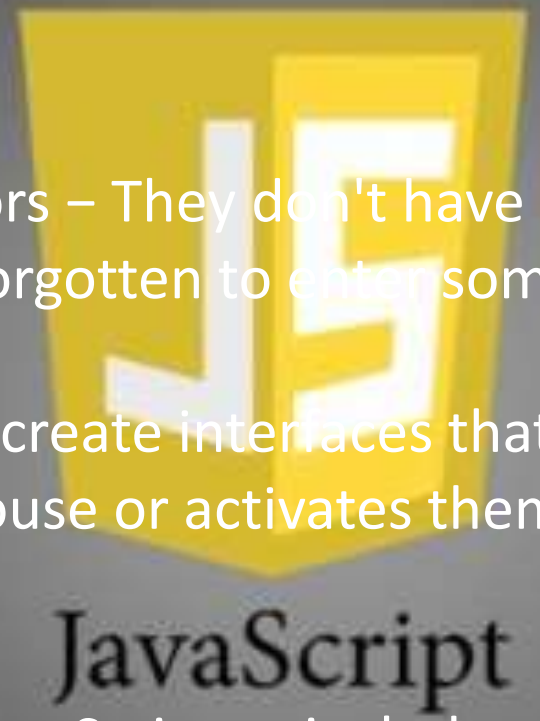
- API (REST & SOAP):

Knowledge of web services or API is also important for full stack developers. Knowledge of creations and consumption of REST and SOAP services is desirable

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.

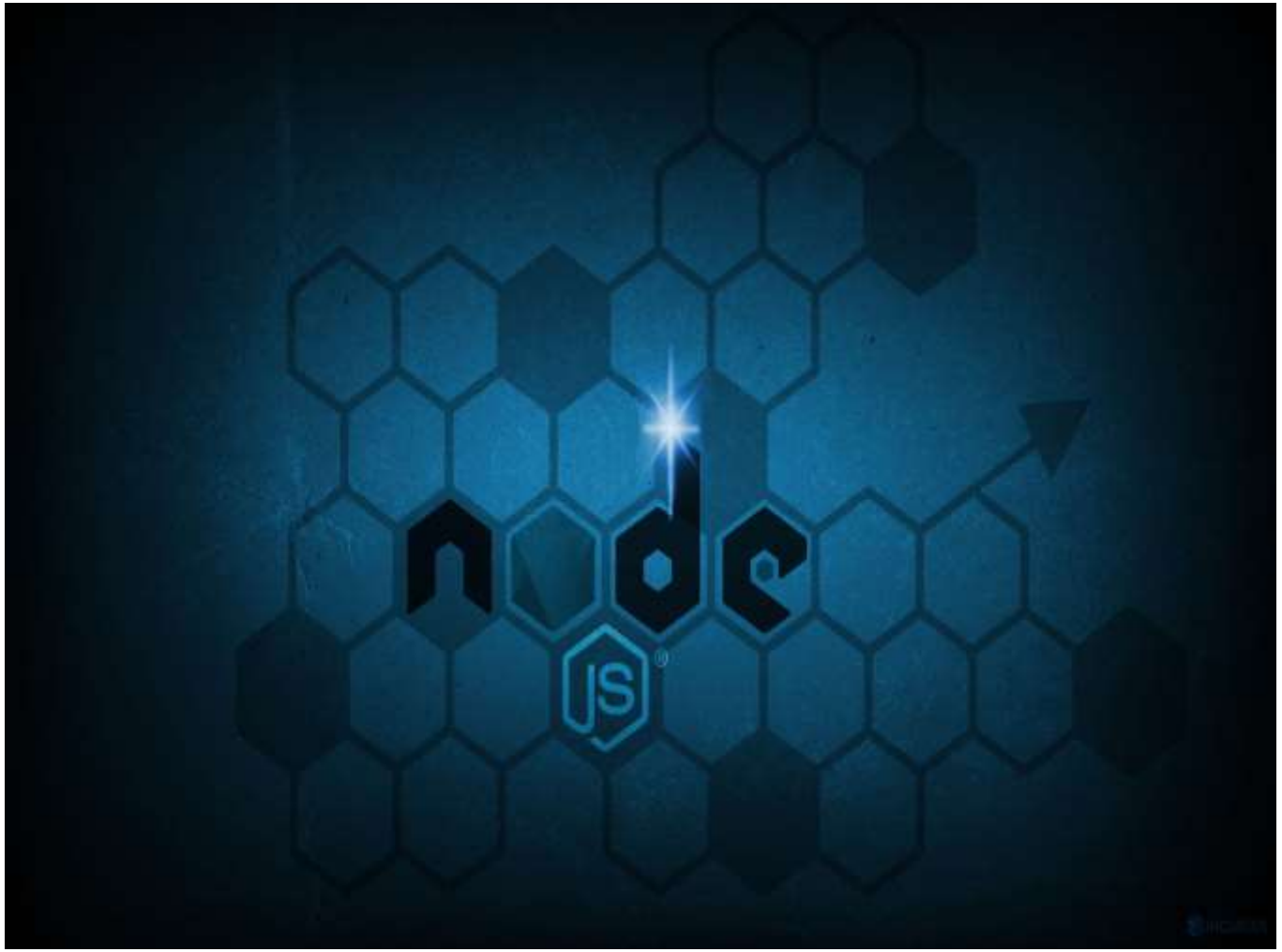
- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform

- Less server interaction − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- Immediate feedback to the visitors − They don't have to wait for a page reload to see if they have forgotten to enter something.
- Increased interactivity − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- Richer interfaces − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

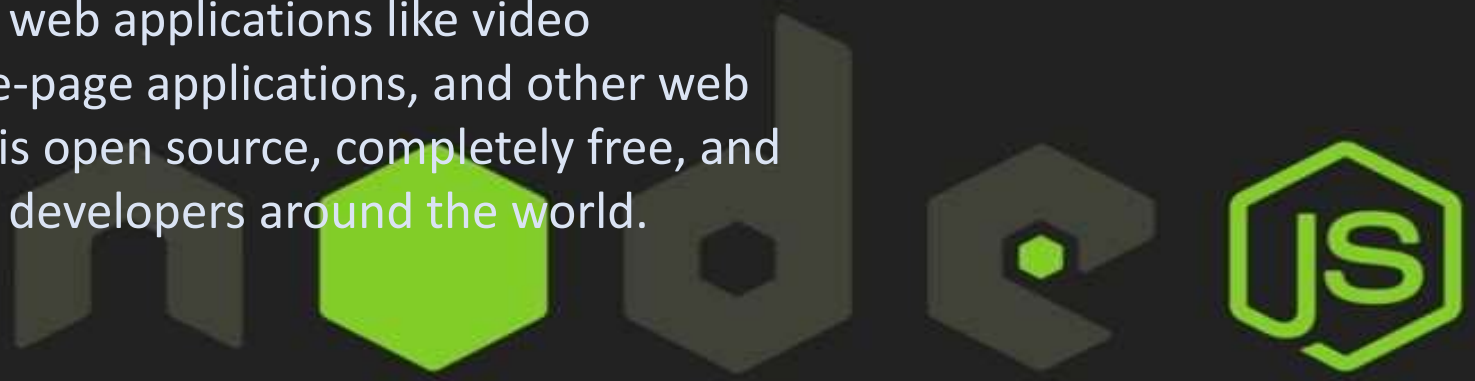


NODE.JS

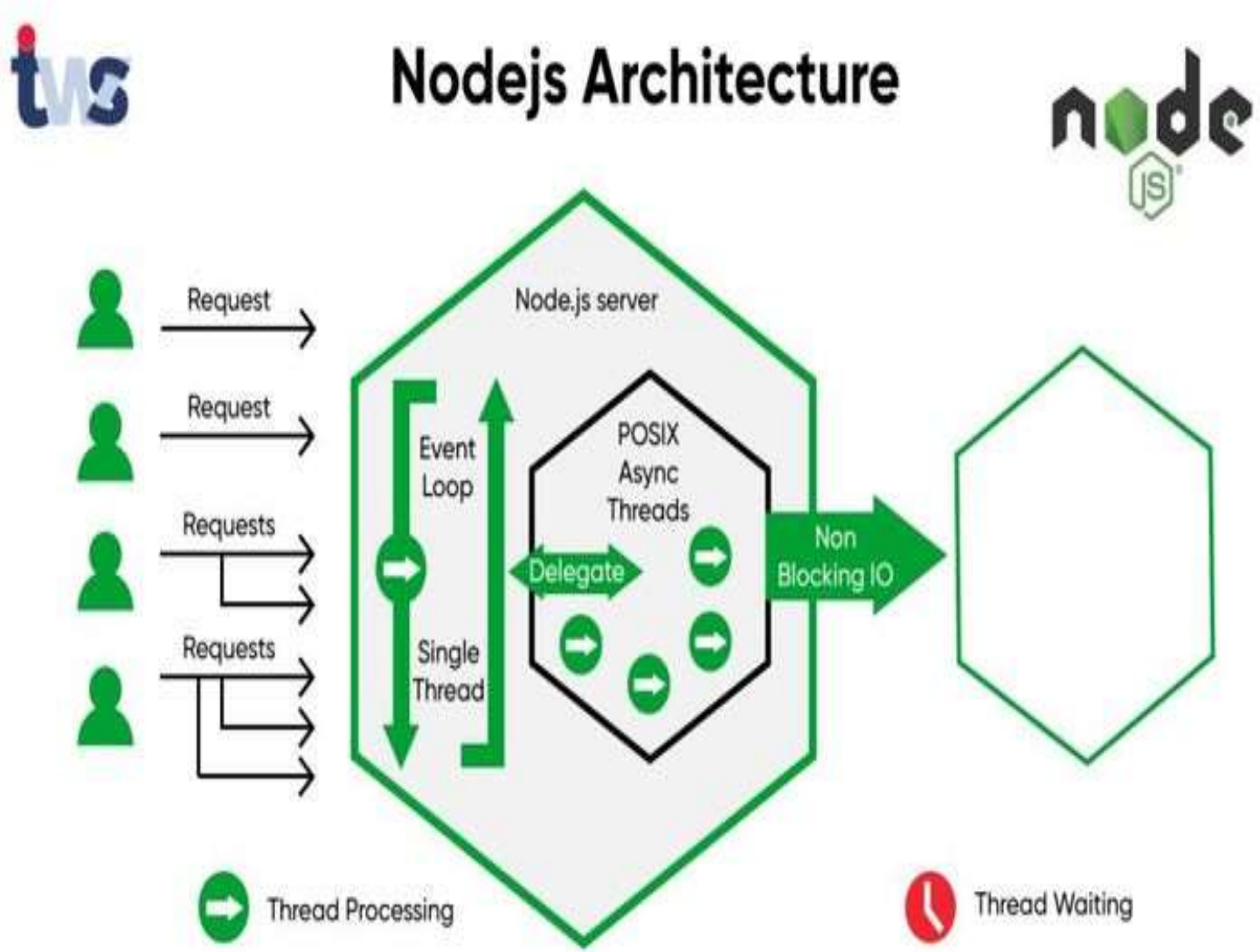
- Node.js is a very powerful JavaScript-based platform built on Google Chrome's JavaScript V8 Engine. It is used to develop I/O intensive web applications like video streaming sites, single-page applications, and other web applications. Node.js is open source, completely free, and used by thousands of developers around the world.



Node.js is a very powerful JavaScript-based platform built on Google Chrome's JavaScript V8 Engine. It is used to develop I/O intensive web applications like video streaming sites, single-page applications, and other web applications. Node.js is open source, completely free, and used by thousands of developers around the world.



- Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.



FEATURES OF NODE.JS

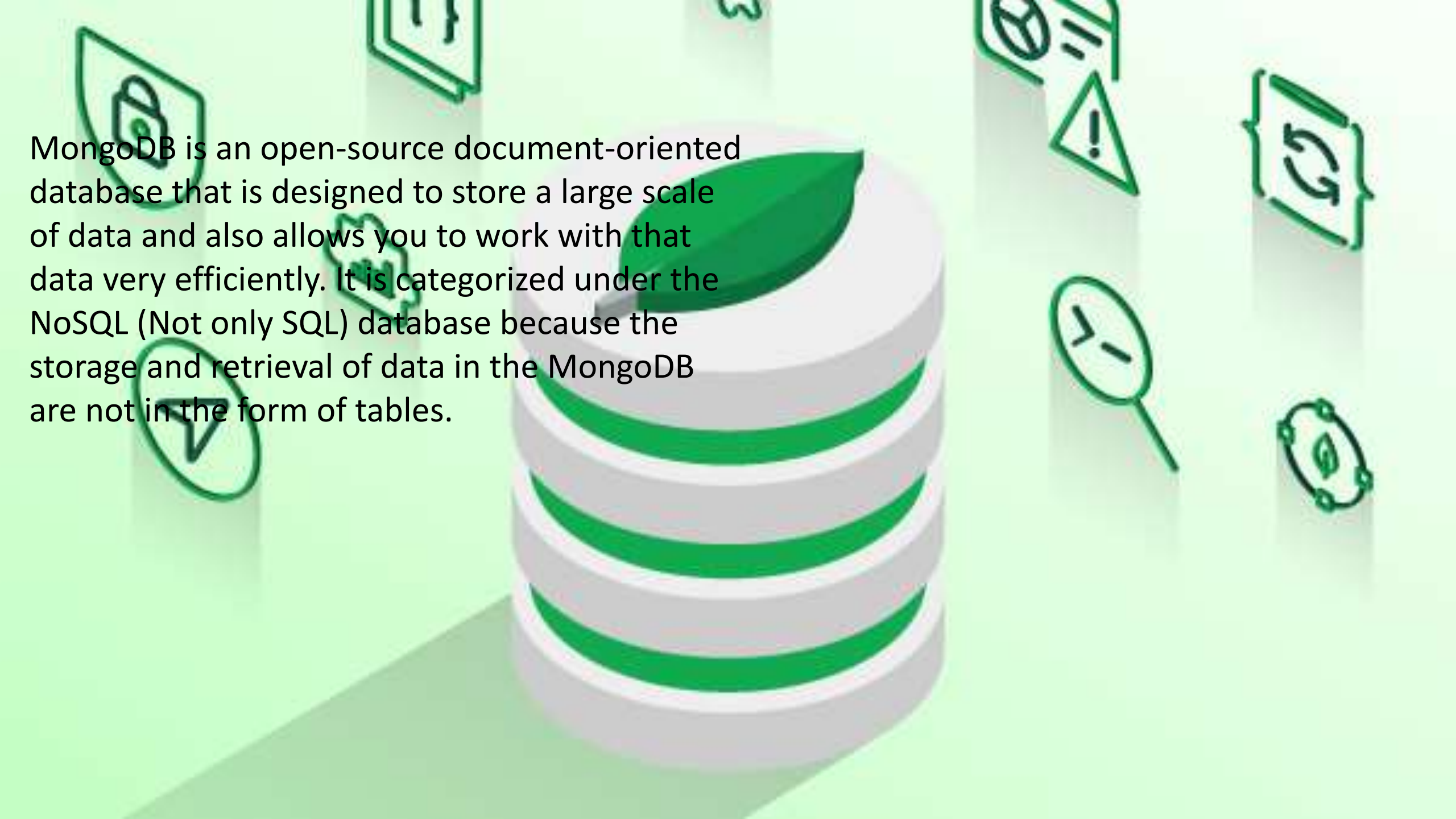
- Following are some of the important features that make Node.js the first choice of software architects.

- Asynchronous and Event Driven − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

- Very Fast − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.

- Single Threaded but Highly Scalable − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

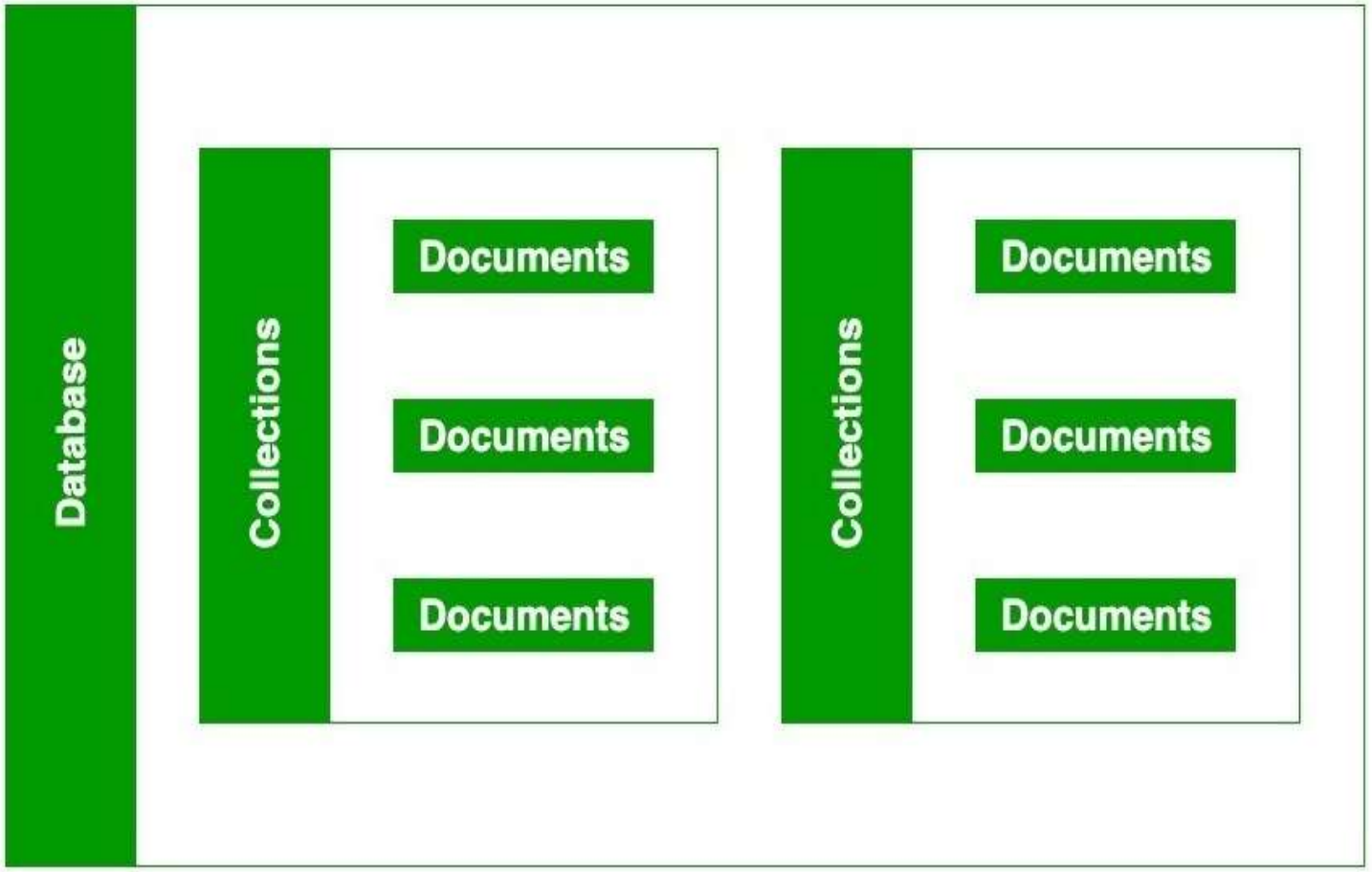
- No Buffering − Node.js applications never buffer any data. These applications simply output the data in chunks.



MongoDB is an open-source document-oriented database that is designed to store a large scale of data and also allows you to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL) database because the storage and retrieval of data in the MongoDB are not in the form of tables.

WORKING

- MongoDB is a database server and the data is stored in these databases. Or in other words, MongoDB environment gives you a server that you can start and then create multiple databases on it using MongoDB.
- Because of its NoSQL database, the data is stored in the collections and documents. Hence the database, collection, and documents are related to each other as shown:



- The MongoDB database contains collections just like the MYSQL database contains tables. You are allowed to create multiple databases and multiple collections.

- Now inside of the collection we have documents. These documents contain the data we want to store in the MongoDB database and a single collection can contain multiple documents and you are schema-less means it is not necessary that one document is similar to another.

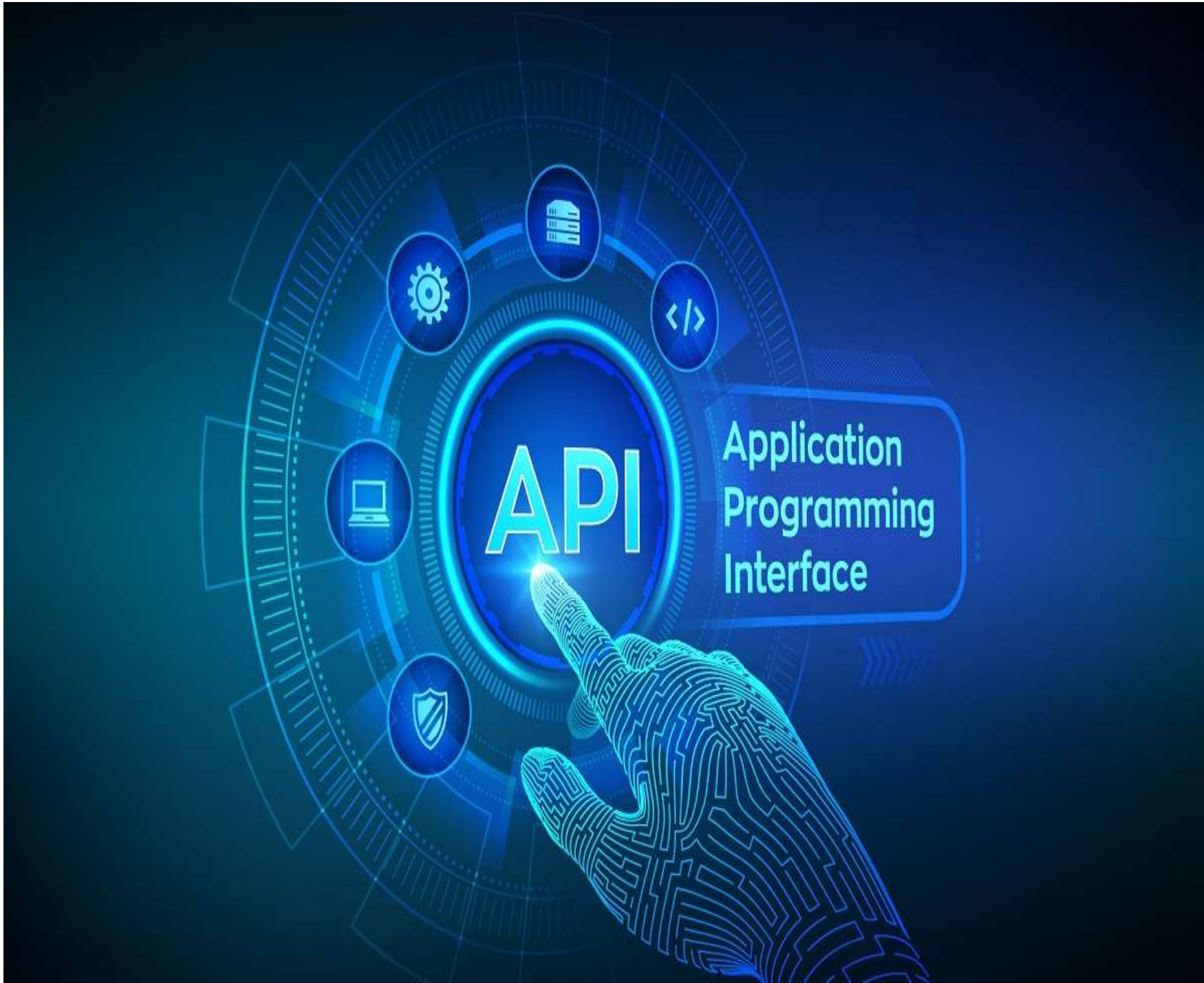
- The documents are created using the fields. Fields are key-value pairs in the documents, it is just like columns in the relation database. The value of the fields can be of any BSON data types like double, string, boolean, etc.

- The data stored in the MongoDB is in the format of BSON documents. Here, BSON stands for Binary representation of JSON documents. Or in other words, in the backend, the MongoDB server converts the JSON data into a binary form that is known as BSON and this BSON is stored and queried more efficiently.

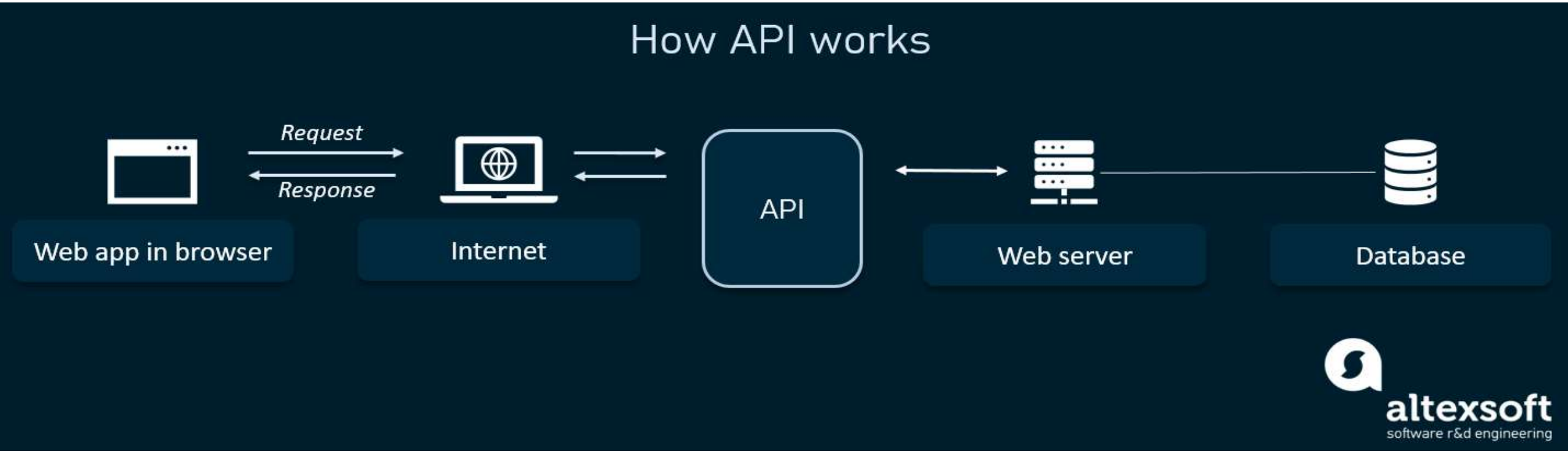
API

- API stands for Application Programming Interface. In basic terms, APIs are a set of functions and procedures that allow for the creation of applications. They access the data and features of other applications, services, or operating systems.

- Essentially, they’re a go-between for different software platforms. They allow two unrelated applications to “talk” to each other.



WORKING



- First of all, the API is not the database or even the server; it's the code that governs the server's access point(s). An API is like a common language, a communications mechanism between developers.

API TYPES BY RELEASE POLICIES

Private

APIs are solely used within an organization

Apps are mostly built for company employees.

Common use cases are the integration of company systems/apps or development of new systems using existing resources

Partner

Openly promoted but available for known business partners

End customers or business users are potential target audiences for such apps

Software integration between two organizations is the popular use case for these APIs

Public or external

Available to any third-party developers

Apps with public APIs are mostly designed for end customers

This API release policy allows for increasing brand awareness and fostering external innovation

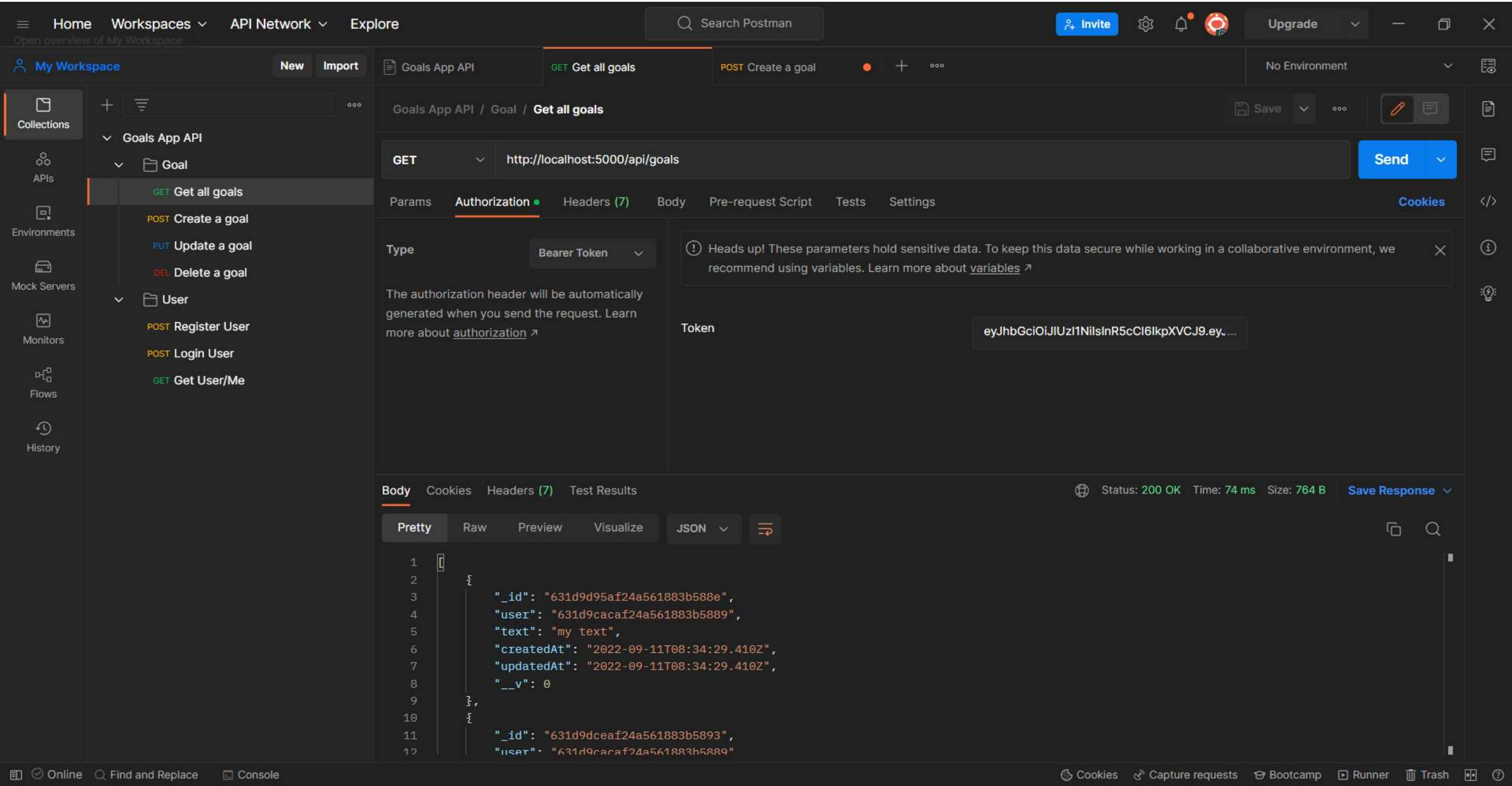
Public APIs can be open and commercial

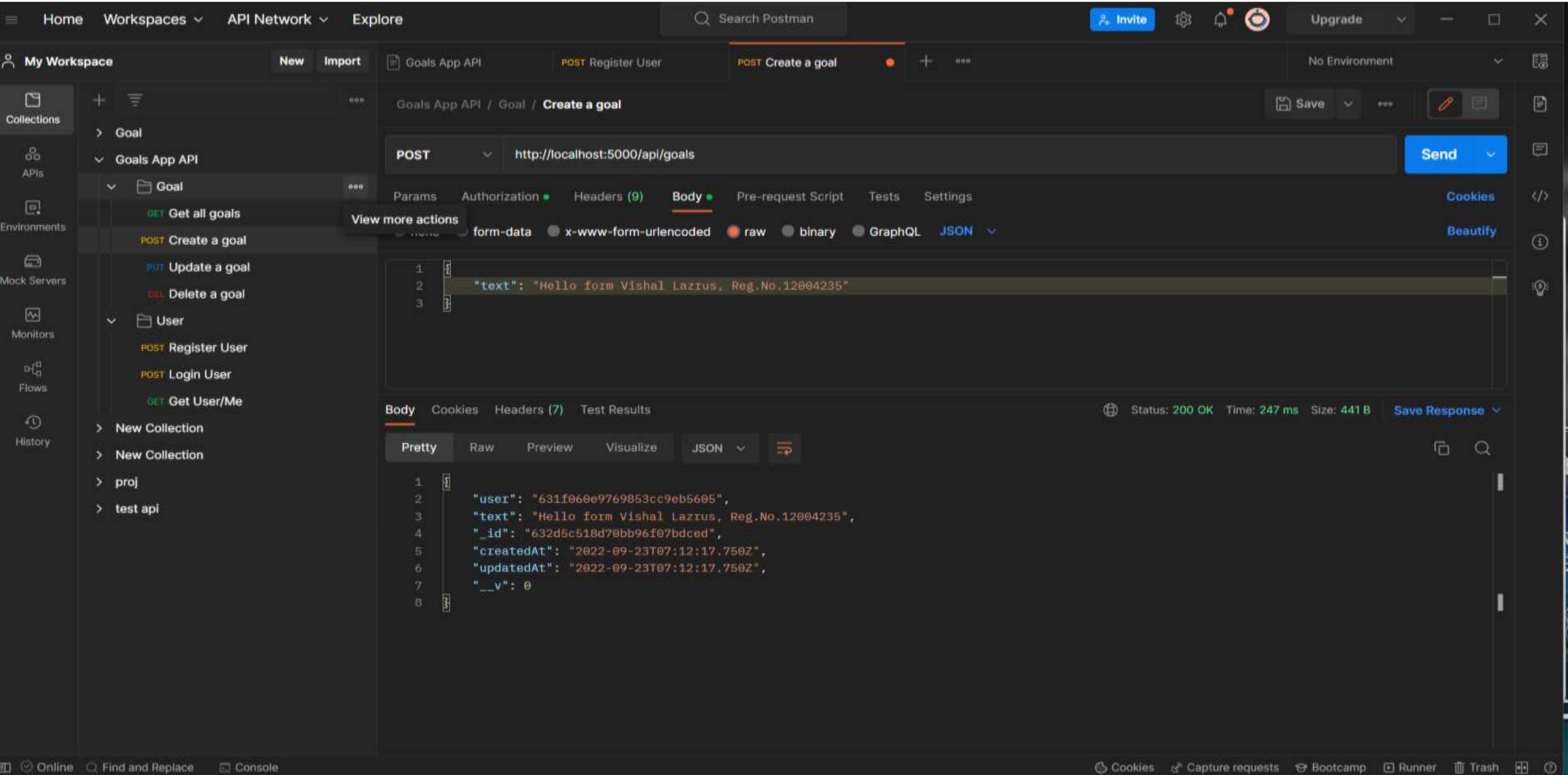
POSTMAN

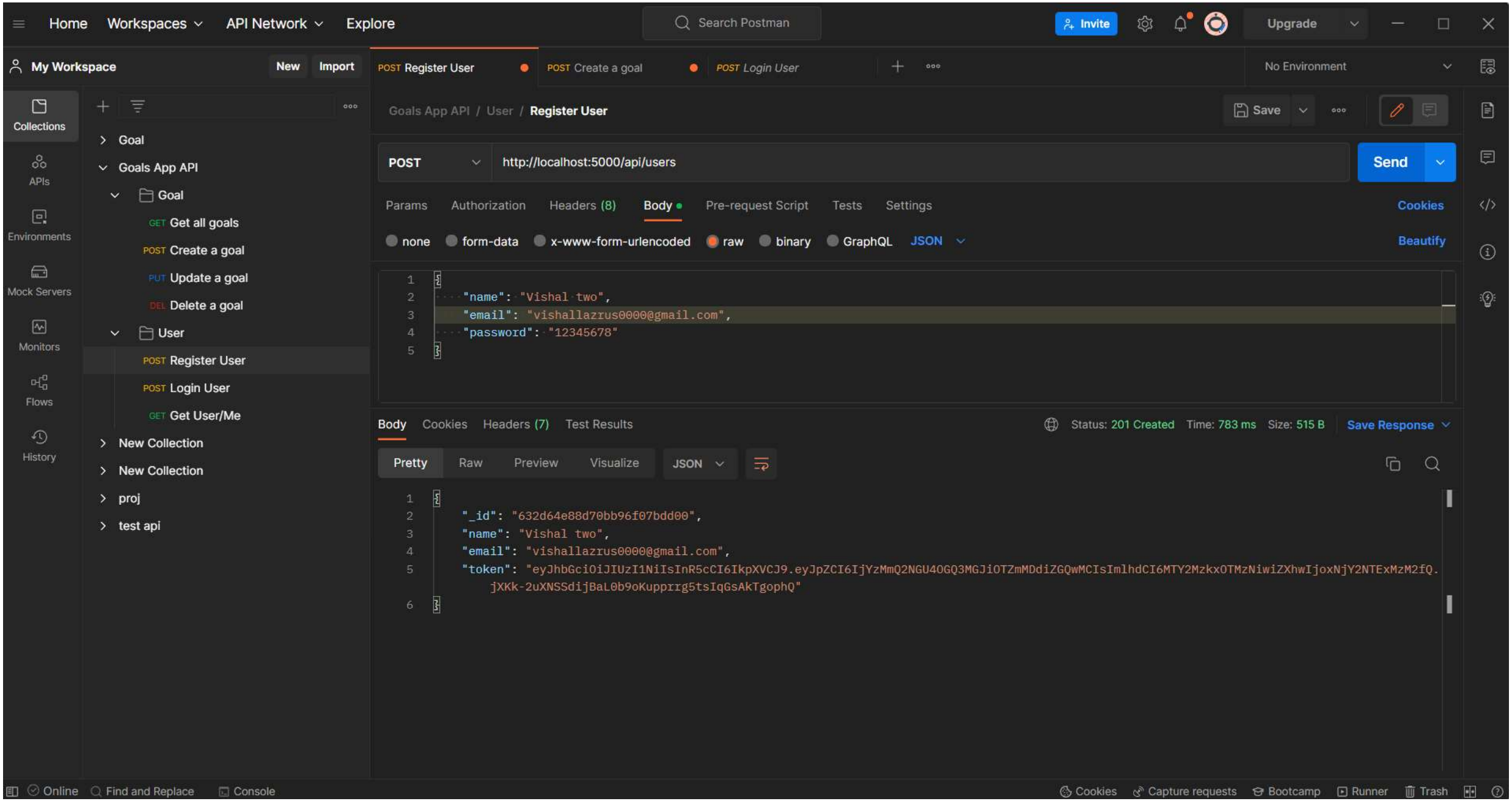
- Postman: Postman is an API(application programming interface) development tool which helps to build, test and modify APIs. Almost any functionality that could be needed by any developer is encapsulated in this tool. It is used by over 5 million developers every month to make their API development easy and simple. It has the ability to make various types of HTTP requests(GET, POST, PUT, PATCH), saving environments for later use, converting the API to code for various languages(like JavaScript, Python).

- In this post, I will use the Postman software to send and receive requests, POST data to the server and I will try to demo some other popular maneuvers. You can treat this article as your first contact with the Postman

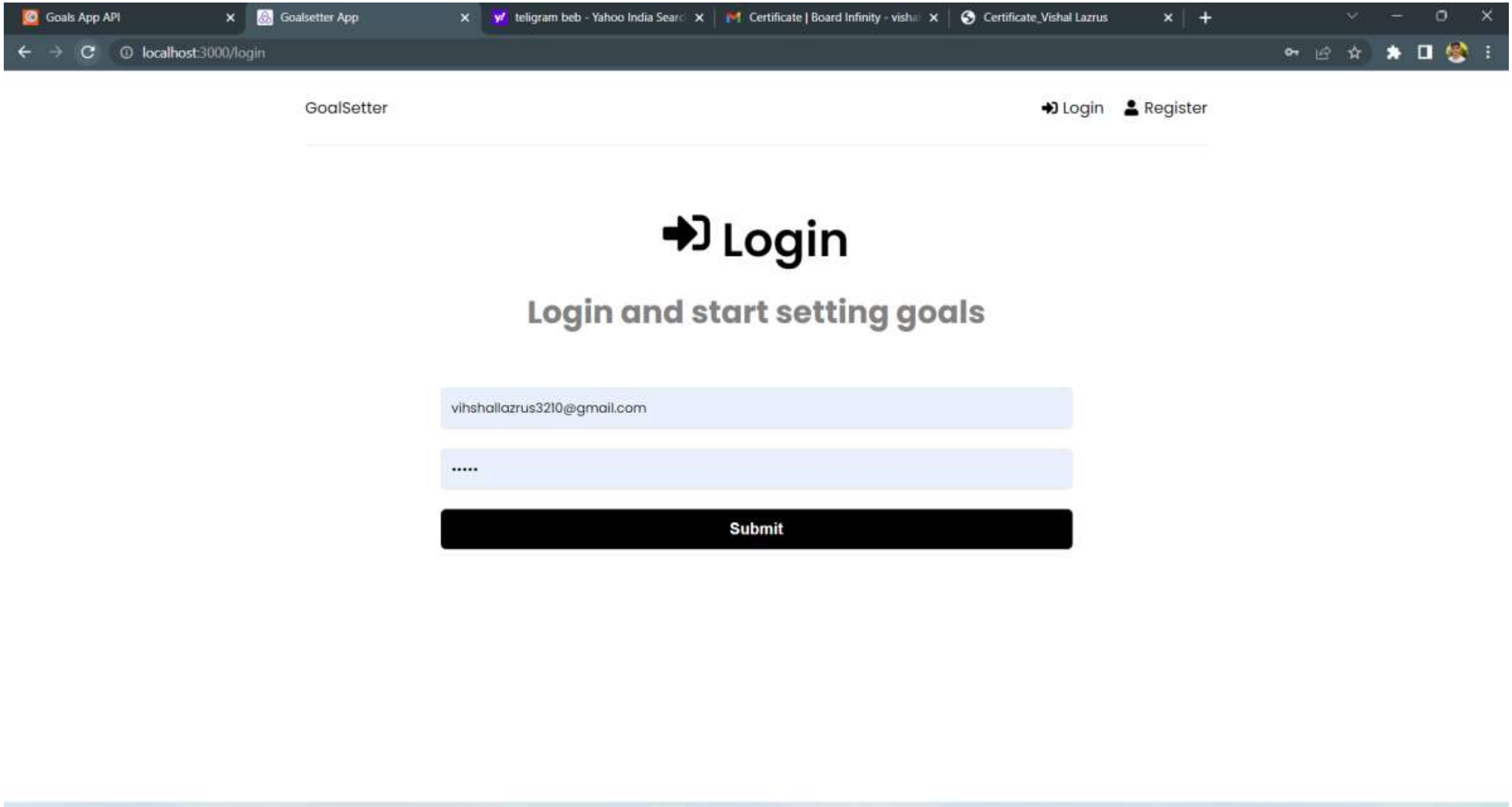
POSTMAN

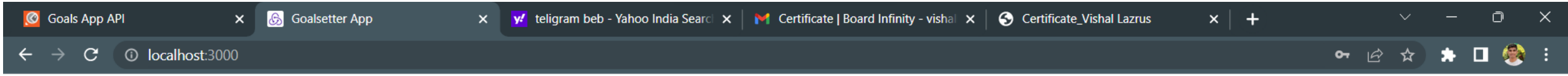

Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Goals App API
GET Get all goals
POST Create a goal
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
Goals App API
Goal
GET Get all goals
POST Create a goal
PUT Update a goal
DEL Delete a goal
User
POST Register User
POST Login User
GET Get User/Me
Goals App API / Goal / Get all goals
Save
GET
http://localhost:5000/api/goals
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Type
Bearer Token
The authorization header will be automatically generated when you send the request. Learn more about authorization
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables
Token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey...
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 74 ms
Size: 764 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"_id": "631d9d95af24a561883b588e",
"user": "631d9cacaf24a561883b5889",
"text": "my text",
"createdAt": "2022-09-11T08:34:29.410Z",
"updatedAt": "2022-09-11T08:34:29.410Z",
"__v": 0
},
{
"_id": "631d9dceaf24a561883b5893",
Online
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Goals App API
POST Register User
POST Create a goal
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
Goal
Goals App API
Goal
GET Get all goals
POST Create a goal
PUT Update a goal
DEL Delete a goal
User
POST Register User
POST Login User
GET Get User/Me
New Collection
New Collection
proj
test api
View more actions
Goals App API / Goal / Create a goal
Save
POST
http://localhost:5000/api/goals
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"text": "Hello form Vishal Lazrus, Reg.No.12004235"
}
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 247 ms
Size: 441 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"user": "631f060e9769853cc9eb5605",
"text": "Hello form Vishal Lazrus, Reg.No.12004235",
"_id": "632d5c518d70bb96f07bdced",
"createdAt": "2022-09-23T07:12:17.750Z",
"updatedAt": "2022-09-23T07:12:17.750Z",
"__v": 0
}
Online
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST Register User
POST Create a goal
POST Login User
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
Goal
Goals App API
Goal
GET Get all goals
POST Create a goal
PUT Update a goal
DEL Delete a goal
User
POST Register User
POST Login User
GET Get User/Me
New Collection
New Collection
proj
test api
Goals App API / User / Register User
Save
POST
http://localhost:5000/api/users
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "Vishal two",
"email": "vishallazrus0000@gmail.com",
"password": "12345678"
}
Body
Cookies
Headers (7)
Test Results
Status: 201 Created
Time: 783 ms
Size: 515 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"_id": "632d64e88d70bb96f07bdd00",
"name": "Vishal two",
"email": "vishallazrus0000@gmail.com",
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzMmQ2NGU4OGQ3MGJiOTZmMDdiZGQwMCIsImlhdCI6MTY2MzkxOTMzNiwiZXhwIjoxNjY2NTExMzM2fQ.
jXKk-2uXNSSdijBaL0b9oKupprrg5tsIqGsAkTgophQ"
}
Online
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash

FRONTEND


Goals App API
Goalsetter App
teligram beb - Yahoo India Sear
Certificate | Board Infinity - vishal
Certificate_Vishal Lazrus
localhost:3000/login
GoalSetter
Login
Register
Login
Login and start setting goals
vihshallazrus3210@gmail.com
.....
Submit



GoalSetter

Logout

Welcome vishal lazrus

Goals Dashboard

Goal

Add Goal

8/23/2022, 12:47:56 PM X

complete remining work

8 8/23/2022, 12:48:19 PM X

ready for class

DOCUMENTATION

ENVIRONMENT No Environment LAYOUT Double Column LANGUAGE cURL

GOALS APP API

Introduction

- Goal
- User

Goals App API

A basic RestAPI where a particular user can create an store as many goals as possible and track his/her progress daily.

Goal

This section contains all the routes related to ***Goals***

GET Get all goals

http://localhost:5000/api/goals

The work of this route is to fetch all the goals created by a particular user. In order to use this route, the user must be **logged in.**

AUTHORIZATION Bearer Token

Token	<token>

Example Request — Get all goals

```
curl --location --request GET 'http://localhost:5000/api/goals'
```

ENVIRONMENT No Environment LAYOUT Double Column LANGUAGE cURL

GOALS APP API

Introduction

- Goal
- User

POST Create a goal

http://localhost:5000/api/goals

StartFragment

The work of this route is to let user create a goal. In order to use this route, the user must be **logged in.**

EndFragment

AUTHORIZATION Bearer Token

Token <token>

BODY raw

```
{
    "text": "my egrg text"
}
```

Example Request — Create a goal

```
curl --location --request POST 'http://localhost:5000/api/goals' \
--data-raw '{
    "text": "my egrg text"
}'
```

Example Request — Update a goal

PUT Update a goal

http://localhost:5000/api/goals/goal_id

StartFragment

The work of this route is to let user update a particular goal. In order to use this route, the user must be **logged in.**

EndFragment

AUTHORIZATION Bearer Token

Token	<token>

BODY urlencoded

text	my second goal completed

Example Request — Update a goal

```
curl --location --request PUT 'http://localhost:5000/api/goals/goal_id' \
--data-urlencode 'text=my second goal completed'
```

DEL Delete a goal

http://localhost:5000/api/goals/goal_id

StartFragment

The work of this route is to let user delete a particular goal. In order to use this route, the user must be **logged in.**

EndFragment

AUTHORIZATION Bearer Token

Token	<token>

Example Request — Delete a goal

```
curl --location --request DELETE 'http://localhost:5000/api/goals/goal_id'
```

User

StartFragment

This section contains all the routes related to ***Users***

EndFragment

POST Register User

```
http://localhost:5000/api/users
```

The work of this route is to let user create an account in Notes App.

BODY raw

```
{
    "name": "test user",
    "email": "testuser@gmail.com",
    "password": "12345678"
}
```

Example Request — Register User

```
curl --location --request POST 'http://localhost:5000/api/users' \
--data-raw '{
    "name": "test user",
    "email": "testuser@gmail.com",
    "password": "12345678"
}'
```

POST Login User

http://localhost:5000/api/users/login

StartFragment

The work of this route is to let user login into his/her an account in Notes App.

EndFragment

BODY urlencoded

email	somtwo@gmail.com
password	123456

Example Request — Login User

```
curl --location --request POST 'http://localhost:5000/api/users/login' \
--data-urlencode 'email=somtwo@gmail.com' \
--data-urlencode 'password=123456'
```

GET Get User/Me

http://localhost:5000/api/users/me

StartFragment

The work of this route is to let user get the info he/she has filled while creating the account in Notes App.

EndFragment

AUTHORIZATION Bearer Token

Token	<token>

Example Request — Get User/Me

```
curl --location --request GET 'http://localhost:5000/api/users/me'
```

LEARNING OUTCOME

- After doing this course and creating the mini-project, I came across several important and vital concepts in backend development. Those concepts are:
- 1. What is backend development and what role does it play in full stack web development.
- 2. Setting and running an express server.
- 3. Connecting the backend API with the MongoDB database.
- 4. Performing CRUD (Create, Read, Update and Delete) operations using ExpressJS and MongoDB.
- 5. Handling various errors such as validation errors and mongoose errors effectively.
- 6. Breaking the whole backend project into MVC (Model, View, and Controller) structure with the help of Routes and Controllers.
- 7. Fundamentals of JsonWebToken (JWT).
- 8. Setting up authentication features like Signup and Login Feature. 9. Creating a particular job with respect to a user.

10. Sending emails using ExpressJS and SendGrid.

11. Testing API using Postman.

12. Generating Swagger Documentation of the API.

13. Deployment of the backend API in production with the help of GIT and Heroku so that the whole world can access it

BIBLIOGRAPHY

The resources that I used while creating this report are:

1. Board Infinity Video Lectures.
2. GeeksForGeeks
3. Stack Overflow
4. YouTube
5. NodeJS Official Documentation
6. ExpressJS Official Documentation
7. MongoDB Official Documentation

CONCLUSION

Backend Development is the backbone of any website present across the Globe. It is the part of the program that stores data but never interacts with users directly. Backend accounts for about 50 to 70 percent of a website's functionality. A good backend API can help create high-performance, fast and secure products with scalable architecture. The backend is responsible for maintaining the application state and managing the load on the server. The backend has a responsibility of what to display in the front-end and what not, with authentication that it can provide sensitive data. In recent years, we have a whole lot of trustworthy websites and mobile applications


Thank You!