Installation Guide

# Version Control

| **Ver** | **Description of Amendment** | **Sections impacted** | **Date** | **Changed By** |
| --- | --- | --- | --- | --- |
| 1.0 | Created template | All | 12 July 2018 | RaviRaj |

Typographical rules

To improve readability, visual markers have been defined to give context as to what is expected around the different sections:

|  |  |
| --- | --- |
| picto_note_LH45 | **Note :**  Indicates useful and complementary information. Explanation, comment, or short   expansion of the text objects that is intended to catch your attention. |

|  |  |
| --- | --- |
| picto_tip_LH45 | **Tip :**  Indicates practical but nonessential information that makes the solution easier to use or operate (e.g. keyboard shortcut, alternative way to perform a step in a procedure, etc.) |

|  |  |
| --- | --- |
|  | **Warning :**  The Warning icon indicates a caution. Would this information be ignored, it could cause possible and irreversible damage to the equipment, data or software. |

***Table of Contents***

[1 Version Control 2](#_Toc519204859)

[2 INTRODUCTION 4](#_Toc519204860)

[2.1 Recommended environment 4](#_Toc519204861)

[2.2 DB Scripts 4](#_Toc519204862)

[2.3 Steps to follow to setup the NodeJs Server (W3Schools admin portal) 6](#_Toc519204863)

[2.4 Steps to follow to setup in Angular (W3Schools admin portal) 6](#_Toc519204864)

[2.5 How to test the API's are secured by JWT token 6](#_Toc519204865)

# INTRODUCTION

This document details the installation guide of assesement.

These Api's are fully secured by JWT tokens.

## Recommended environment

* NodeJs.
* PostgresSQL 9.3.
* Your favurote frontend IDE (We used VSCode).
* Express-js.
* Postman(REST Client).

## DB Scripts

* Create Database:-  
    
  CREATE DATABASE w3schools

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'English\_India.1252'

LC\_CTYPE = 'English\_India.1252'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1;

* Tables:-

CREATE TABLE public.programming\_language

(

id bigint NOT NULL DEFAULT nextval('programming\_language\_seq'::regclass),

date\_creation timestamp without time zone NOT NULL,

name character varying(20) COLLATE pg\_catalog."default",

title character varying(50) COLLATE pg\_catalog."default",

introduction text COLLATE pg\_catalog."default",

CONSTRAINT pl\_key\_id PRIMARY KEY (id),

CONSTRAINT pl\_unuque\_name UNIQUE (name)

)

WITH (OIDS = FALSE)

TABLESPACE pg\_default;

ALTER TABLE public.programming\_language

OWNER to postgres;

CREATE TABLE public.lesson

(

id bigint NOT NULL DEFAULT nextval('lesson\_seq'::regclass),

date\_creation timestamp without time zone NOT NULL,

lession\_name character varying(20) COLLATE pg\_catalog."default",

description character varying(20) COLLATE pg\_catalog."default",

details text COLLATE pg\_catalog."default",

pl\_id bigint NOT NULL,

CONSTRAINT lession\_key\_id PRIMARY KEY (id),

CONSTRAINT "FK\_LAN\_ID" FOREIGN KEY (pl\_id)

REFERENCES public.programming\_language (id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

WITH ( OIDS = FALSE)

TABLESPACE pg\_default;

ALTER TABLE public.lesson

OWNER to postgres;

CREATE TABLE public.example

(

id bigint NOT NULL DEFAULT nextval('example\_seq'::regclass),

date\_creation timestamp without time zone NOT NULL,

ex\_name text COLLATE pg\_catalog."default",

example text COLLATE pg\_catalog."default",

le\_id bigint NOT NULL,

CONSTRAINT ex\_key\_id PRIMARY KEY (id)

CONSTRAINT "FK\_LE\_ID" FOREIGN KEY (le\_id)

REFERENCES public.lesson (id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

WITH (OIDS = FALSE)

TABLESPACE pg\_default;

ALTER TABLE public.example

OWNER to postgres;

CREATE TABLE public.w3users

(

id bigint NOT NULL DEFAULT nextval('user\_seq'::regclass),

date\_creation timestamp without time zone NOT NULL,

name character varying(20) COLLATE pg\_catalog."default",

email character varying(50) COLLATE pg\_catalog."default",

password text COLLATE pg\_catalog."default",

CONSTRAINT u\_id PRIMARY KEY (id),

CONSTRAINT u\_email UNIQUE (email)

)

WITH (OIDS = FALSE)

TABLESPACE pg\_default;

ALTER TABLE public.w3users

OWNER to postgres;

SEQUENCES:

CREATE SEQUENCE public.example\_seq;

ALTER SEQUENCE public.example\_seq

OWNER TO postgres;

CREATE SEQUENCE public.programming\_language\_seq;

ALTER SEQUENCE public.programming\_language\_seq

OWNER TO postgres;

CREATE SEQUENCE public.lesson\_seq;

ALTER SEQUENCE public.lesson\_seq

OWNER TO postgres;

CREATE SEQUENCE public.user\_seq;

ALTER SEQUENCE public.user\_seq

OWNER TO postgres;

## Steps to follow to setup the NodeJs Server (W3Schools admin portal)

* Grab the code from git -> <https://github.com/raviGanapuram/task_server>
* Make sure the above listed tools and servers should be up and running
* Execute the above db scripts in pgAdmin(which is the client for postgresSQL)
* Open the server code and change the db details in constants.js file
* Open the internal terminal in IDE and install all the node modules by executing  
  npm install.
* Now start the server by executing the below command.  
  npm start.

## Steps to follow to setup in Angular (W3Schools admin portal)

* Grab the code from git -> <https://github.com/raviGanapuram/task_angular>
* Open the project and edit the server.js to change the server url
* Open the integrated terminal and execute the below command  
  npm install  
  npm start

## How to test the API's are secured by JWT token

In this exercise we have chosen to JWT tokens to secure our endpoints. The below steps are to demonstrate how an invalid JWT token will prevent the endpoint to be executed.

* First we need atleast one RestClient to test the REST services e.g. Postman.
* Register the user by using the postman  
  Request:-  
  POST /api/auth/register HTTP/1.1

Host: localhost:3000

Content-Type: application/json

Request-Body:-

{

"name":"rajesh",

"email":"rajesh@gmail.com",

"password":"rajesh"

}

Response:-

{

"auth": true,

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjI4IiwibmFtZSI6Ik11cyIsImVtYWlsIjoibXVzQGdtYWlsLmNvbSIsImlhdCI6MTUzMTQxNzM0OCwiZXhwIjoxNTMxNTAzNzQ4fQ.qnBYN\_gp6vOEQ1xeyu-qzdcoJs6DBBW\_CI0MQvU2JcM"

}

* Login with the registered details  
  Request:-

POST /api/auth/login HTTP/1.1

Host: localhost:3000

Content-Type: application/json

Cache-Control: no-cache

Postman-Token: 356cae4e-56b7-07cb-959a-0d0c22dedf45

{

"email":"mus@gmail.com",

"password":"password"

}

Response:-

{

"auth": true,

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjI4IiwibmFtZSI6Ik11cyIsImVtYWlsIjoibXVzQGdtYWlsLmNvbSIsImlhdCI6MTUzMTQxNzM0OCwiZXhwIjoxNTMxNTAzNzQ4fQ.qnBYN\_gp6vOEQ1xeyu-qzdcoJs6DBBW\_CI0MQvU2JcM"

}

* Lets check the RestApi's with and without jwt tokens
* The below api was created only for the testing of JWT.

Request:-

GET /api/auth/me HTTP/1.1

Host: localhost:3000

Pass the below token in headers;

access-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjIwIiwibmFtZSI6InJhdmlyYWoiLCJlbWFpbCI6InJhdmlAZ21haWwuY29tIiwiaWF0IjoxNTMxMzk5NjgzLCJleHAiOjE1MzE0ODYwODN9.6dFMWJ6jZU89Uh\_pXtSFCr\_jhcCTlE7iHEAqY9QUoHk

Response:-  
{

"id": "20",

"date\_creation": "2018-07-12T12:11:30.089Z",

"name": "raviraj",

"email": "ravi@gmail.com",

"password": "$2a$08$qAbgzXjpC7ZCXwMzlqN3U.ZnKePCRngsDs2BAqK9waXk9Lj5L44WO"

}

Note: In the above response we are returning the encrypted password to demonstrate this example. It is not recommened to display the password in any format.

Request with invalid token:-

GET /api/auth/me HTTP/1.1

Host: localhost:3000

access-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjIwIiwibmFtZSI6InJhdmlyYWoiLCJlbWFpbCI6InJhdmlAZ21haWwuY29tIiwiaWF0IjoxNTMxMzk5NjgzLCJleHAiOjE1MzE0ODYwODN9.6dFMWJ6jZU89Uh\_pXtSFCr\_

Cache-Control: no-cache

Postman-Token: 1fa8cfdc-e6e0-8a71-efb7-97f7aa8b1c39

Response:-  
{

"auth": false,

"message": "Failed to authenticate token."

}