# Technical Test

Article API

Bikin Maharjan

[bikinmaharjan10@gmail.com](mailto:bikinmaharjan10@gmail.com)

# Setup and Installation guide

* Make sure have Nodejs Installed on your system. You can check it in your terminal by entering node –version. If nothing shows up, you can install it via the link below:

<https://nodejs.org/en/>

* Clone the repository from: <https://github.com/bikinmaharjan/nineProject>
* Run **‘npm install’** to install all the packages required for the api
* Run **‘npm start’** to run the api on production environment or **‘npm run dev’** to run on development mod.
* Open your choice of api development tool like Postman to query the api
* The api runs on port 3000 by default.
* The Postman documentation for the Api can be found at:  
  <https://documenter.getpostman.com/view/7877219/UVsFz8y2>

# About the Solution

* I have used ExpressJs for this test as I am familiar with it. For the database I have used MongoDB.
* The packages that I have used are
* Cross-env
* Dotenv
* Express
* Moment
* Mongoose
* Nodemon
* The main entry point for the api is index.js
* I have made specific folders for
* the model of the Article so that it can hold the information about what the field will accept. It uses mongoose schema to create a database model for article.
* The routes that handle the specific request for the article. Since it is a small project all the functions are in one file. If the project was big, I would have chosen to have different file for different end points.
* The configs where all the configs for the app that are private should be stored. For the purpose of this test I have not added the folder in ‘.gitignore’. It currently holds the port at which the app should run. If port 3000 is being used in your system you can use any other port that are free to run the api. It also has the MongoURL with the full username and password. This would of course be hidden if it was a real project.

# Assumptions made

* For the id of article, I have used the default \_id provided by MongoDB as it is safe and easy to work with. Mongoose has a lot of functions that can take use of it.
* For the date I have used the date and time the article is posted.
* Both the \_id and date are set by mongoose, so the user doesn’t have to enter it while posting an article.
* The title and body accept String value and the tags accepts an array.
* Since its a NoSQL database it can accept any tags.
* Tags to always use lowercase as tags
* The title has a limit of 100 characters as the title of an article shouldn’t be too long.
* For the article to be posted the title and body shouldn’t be empty while as the tags can be left empty.
* While any of the function is successful the status code is 200 whereas if anything goes wrong the status code is 500.
* The error is handled by the function provided by mongoose.
* I have also added GET /articles to get all the articles so you can view all the articles and make queries for it accordingly.
* All of the queries are async functions.
* I have used momentjs to convert the string date from the request to the appropriate date to get the results.