# Introduction

## 1.1 Purpose

This document holds the low level design specification for the Server component of the project “Anotode”. It is supposed to be followed during the coding phase and will be used as a reference for using the API services provided by the server.

## 1.2 Document Conventions

Server means “Services provided by API”. The API provides services to retrieve data/send data for the client.

## 1.3 Intended Audience and Reading Suggestions

The audience of this document is assumed to be the development team and whosoever who is part of development of the server and in some cases, the entire project. It is recommended that anyone reading this document must have prior knowledge of how an API work. An experience in developing some REST APIs would come really handy.

## 1.4 References

Various references were used in preparing this document. A non-exhaustive list of them can be listed as follows -

* Rest API tutorial <http://adrianmejia.com/blog/2014/10/01/creating-a-restful-api-tutorial-with-nodejs-and-mongodb>/
* Creating RESTAPI in mongoDB and nodeJS <http://adrianmejia.com/blog/2014/10/01/creating-a-restful-api-tutorial-with-nodejs-and-mongodb/>
* MongoDB object modeling <https://github.com/Automattic/mongoose>
* JWT library <https://github.com/auth0/node-jsonwebtoken>

# 2. Design Overview

A simple overview of the application with an optional flowchart diagram.



# 3. Design Description

A server-side web API is a programmatic interface consisting of one or more publicly exposed endpoints to a defined request–response message system.In our case, the response is emmited in JSON format by means of an HTTP-based web server. This API serves as an endpoint for other application to interact with the API.

This same API will be used by the other client apps to get the highlights of a particular user.

## 3.1. Codebase structure

The API was made using NodeJS programming Language. For database we have decided to use MongoDB and Mongoose for Object Document Modelling(ODM)

## 3.2. User Interface

We have 4 main interfaces.

1. Creating a User
2. login
3. saving data.
4. Viewing our data

## 3.3. Model Diagrams

Data modeling is a conceptual model of how data items relate to each other. To manage the highlight of the user we have to make a 2 entities- Highlights, User.

The below picture shows the Data base design to manage the highlights of the user.



All database changes are made in parallel to required code changes

## 3.4 Event/Message flow

. **Creating a user:**

send a POST request to http://anotode.herokuapp.com/api/users/

with parameters

* email:
* password:

### **Login:**

Send a POST request to http://anotode.herokuapp.com/api/login/ with your login credentials

then you will receive a token. Copy that token.

### **Sending your highlights to DataBase:**

send a POST request to http://anotode.herokuapp.com/api/highlights?token=<yourtoken>

and body with your text and title and tags as parametres.

### **To get list of all your data:**

send a GET request to http://anotode.herokuapp.com/api/highlights?token=<yourtoken>

## 3.5. Error/Exception Handling

Errors are the most commonly occurring problems any systems. To eliminate any problems that occur due to platform, we use Travis CI continuous integration platform for GitHub projects.

# 4. Configuration

This section lists the configurations that need to be done in order to get the system running.

## 4.1 Adapter / Connector Configuration

### **Starting**

* Make sure you have mongodb installed on your system. apt-get install mongodb
* Then install the application dependencies.

npm install

* Finally, start the server.

npm start

* Visit http://localhost:3000 to view the server.

## 4.2 Application Configuration

The system will be functional as Web App and will be functional in all latest versions of chrome.

## 4.3 Third Party tool Configuration

**1.Travis CI:**

Third party tools like travis CI.Travis CI is configured by adding a file named .*travis.yml*. This file specifies the programming language used, the desired building and testing environment (including dependencies which must be installed before the software can be built and tested), and various other parameters.

**2.Heroku:**

Heroku is a web application deployment model.The main content of the development are the source code, related dependencies if they exist, and a Procfile for the command.The application is sent to Heroku using Git. Heroku’s HTTP routers distribute incoming requests for the application across the running web

# 5. Assumptions

Assumptions made when writing this document.

* The user will have a basic knowledge of API s
* The user knows basic of the deploying to server
* The user knows basic knowledge of various CI integration tools.
* The user know how heroku Deployment works

# 6. Dependencies

Dependencies of the component. Like ODMs like mongoose.Mongoose is a Node.js library that provides MongoDB object mapping similar to ORM with a familiar interface within Node.js. If you aren't familiar with Object Relational Mapping (ORM) or, Object Data Mapping (ODM) in the case of Mongoose, this means is that Mongoose translates data in the database to JavaScript objects for use in application.