ANOTODE

The Web Annotator .

LLD - Server

Revision 3.0

CS Group 1

Authors:
Manohar (201451024)

Reviewed by:
Avi (201451070)

Revision Table

Revision	Author	Reviewer	Revision Date	Revision Tracking Notes
1	Saurabh	Manohar	26-9-16	Added intended audience
2	Avi	Manohar	27-9-16	Added requirements and other software attributes
3	Avi	Mukesh	28-9-16	Added use case diagram

Contents

1 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 developing the extension part of the project.

1.2 Document Convention

This document uses the same standards as our all other documents and the specifications for the same has been defined in the Document Convention document. Apart from those defined there, in this document we are using some conventions which are as follows -

- Server means Services provided by API.

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 backend and in some cases, the entire project. It is recommended that anyone reading this document must have prior knowledge of how APIs 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

Browser extension consists of 3 parts .

- Content scripts
- Background pages
- Popup

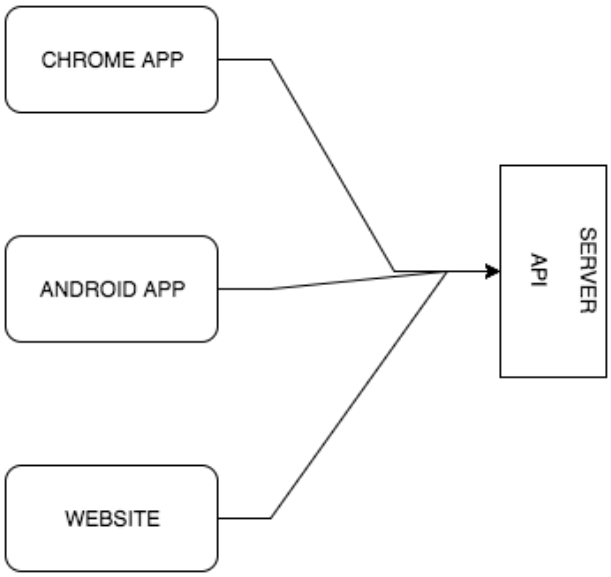
Content scripts They are responsible for highlighting the webpage in case of highlight or re-highlighting in case of re-highlight.

Background pages They are responsible for loading data from the server and saving it back on the server.

Popup It is responsible for showing the GUI that allows user to login/signup and change settings.

3 Design Description

3.1 Codebase Structure



Codebase should be organized as the stated below.

- DB Models should be placed in models folder.
- API routes should be saved in routes folder.
- Test scripts will be saved in test folder.

- all the modules that are being used are saved in node$_modules.README.mdshouldbecreatedtoincludebasicinf$

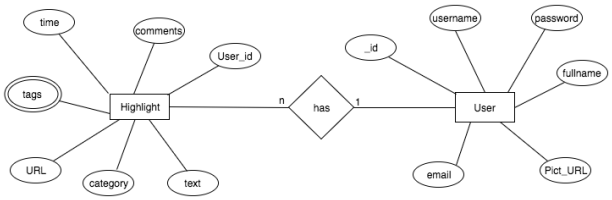
3.2 User Interface

We have 4 main interfaces.

- Creating a User
- login
- saving data.
- Viewing our data

3.3 Model Diagram

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 diagrams

3.4.1 Creating a user

send a POST request to http://anotode.herokuapp.com/api/users/
with parameters
email:
password:

3.4.2 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.

3.4.3 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.

3.4.4 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.

3.6 Configuration

3.6.1 Starting

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

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

  Finally, start the server.

3.7 Application Configuration

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

3.8 Third Party tool Configuration

Using third party tools like some external database server and so.. List here.

3.8.1 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.

3.8.2 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. Herokus HTTP routers distribute incoming requests for the application across the running web

4 Customization

If it is possible to customize application, write it here

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.