**Boğaziçi University**

**Software Engineering MS Program**

**SWE 599 Project, Fall 2017**

**Supervisor: Suzan Üsküdarlı**

**Motive:**

**Community Information Processing Project**

**Final Report**

**18.12.2017**

**Revision 1.0**

**By: Yasemin Alpay**

**Student Id: 2015719012**

**Table of Contents**

[1. Introduction 2](#_Toc501310137)

[2. Software Requirements 3](#_Toc501310138)

[3. System Design 5](#_Toc501310139)

[**3.1. Technical Overview** 5](#_Toc501310140)

[**3.2. Software Technologies and Tools** 5](#_Toc501310141)

[**3.3. System Architecture** 6](#_Toc501310142)

[**3.4. API List** 7](#_Toc501310143)

[**3.4. Class Diagram** 8](#_Toc501310144)

[4. User Interface 9](#_Toc501310145)

[5. Continuous Integration and Deployment 12](#_Toc501310147)

[APPENDIX A: Postman Collection 14](#_Toc501310148)

[APPENDIX B: Deployment Manual 19](#_Toc501310149)

## Introduction

The purpose of this project to develop a community information processing web and mobile application. It aims to provide users to create and organize content and connect with each other in meaningful ways and to create information paths that can be utilized for more effective and enjoyable collective action.

After users are logged in to the system, they can create interest specific groups (interests) or subscribe existing ones. They can create content in these interests and share their knowledge about the content or interest in pre-defined ways. Content information is interest specific, so that users can create their own way of communication for a specific interest.

Features of the application as the following:

1. Creating interests that has custom fields
2. Searching interests
3. Viewing interest details
4. Creating content in the interests
5. Searching existing contents
6. Displaying contents
7. Commenting to contents
8. Having a profile page

## Software Requirements

**Functional Requirements**

User Account Operations

1. User shall be able to provide their username, email and password while creating a new account.
2. User shall be able to login to their account using username and password.
3. User shall be able to recover their password.

Creating Interests

1. User shall create interest specific groups (interests) providing the title, description, privacy of the interest (public or private), members. Titles shall be unique.
2. Users shall add custom fields to interests, so that each content created for specific interest would have its own custom fields.
3. User shall be the admin of the interest that they have created.
4. Admin of the interest shall be able to make other members admin.
5. Admin of the interest shall be able to add new members to the interest or remove members from the interest.
6. Admin of the interest shall be able to change the privacy setting of the interest, so they can set the privacy to public or private.
7. If the admin change the privacy of the interest to private, contents of the interest shall directly become private or vice versa.

Searching Interests

1. User shall search interests by title.

Viewing Interests

1. User shall be able to view interest details (title, description, privacy, and subscription status).

Creating contents in the interests

1. Users shall contents with interest specific fields.

Searching content

1. User shall search content by title or tags.

Displaying content

1. User shall view the details (title, description, tags, creator, and vote count) of the content.

Commenting the content

1. Users shall comment the content if they can see the content.

User profile

1. Users shall have profile page.
2. Profile page shall display the name of the user, interests, contents, annotations, and comments created by user.
3. User profile shall be private or public.

**Non-functional Requirements**

1. The system shall expose a Rest API allowing interaction with other systems.
2. System shall be able to provide Web client.
3. The system shall use open source libraries and tools.

## System Design

### **3.1. Technical Overview**

Motive is a client-server application so it consists of a client side which is web application and a server side exposing Restful API to be consumed by client side. Server (backend) side is on Spring Boot with MongoDB as NoSQL database. Client side is responsive using Bootstrap 4 and AngularJS.

### **3.2. Software Technologies and Tools**

Following software technologies and tools are used to develop "Motive" web application:

Backend:

* Java 1.8.0\_152
* Gradle 4.4
* MongoDB 3.6

Frontend:

* Angular JS 1.5.8
* Bootstrap 4
* Font Awesome 4
* jQuery
* Require JS

Unit and Integration Testing:

* Mockito
* JUnit

IDE and Tools:

* Robo 3T (Previously Robomongo) for MongoDB
* IntelliJ Idea 2017
* Eclipse Oxygen

Source Control:

* Git with GitHub (https://github.com/yaseminalpay/motive)

Continuous Integration:

* Travis CI with Github integration

Deployment

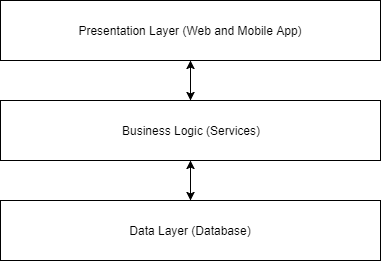
* Amazon Web Services (Elastic Beanstalk and EC2)

Project Process Management:

* GitHub

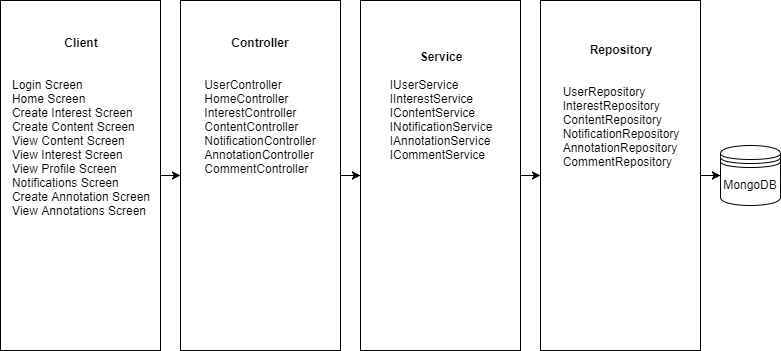
### **3.3. System Architecture**

For architectural design, multi-tier architecture is going to be followed and depicted as the following:



*Architectural Layer Diagram*

Also, in implementation level, client will communicate to the server through the API (controllers in Spring Boot application) and controllers will manipulate data in the database with repository and service layer. Service layer provides abstraction in case of any change in database-specific repository layer:

**

*Structure Diagram*

### **3.4. API List**

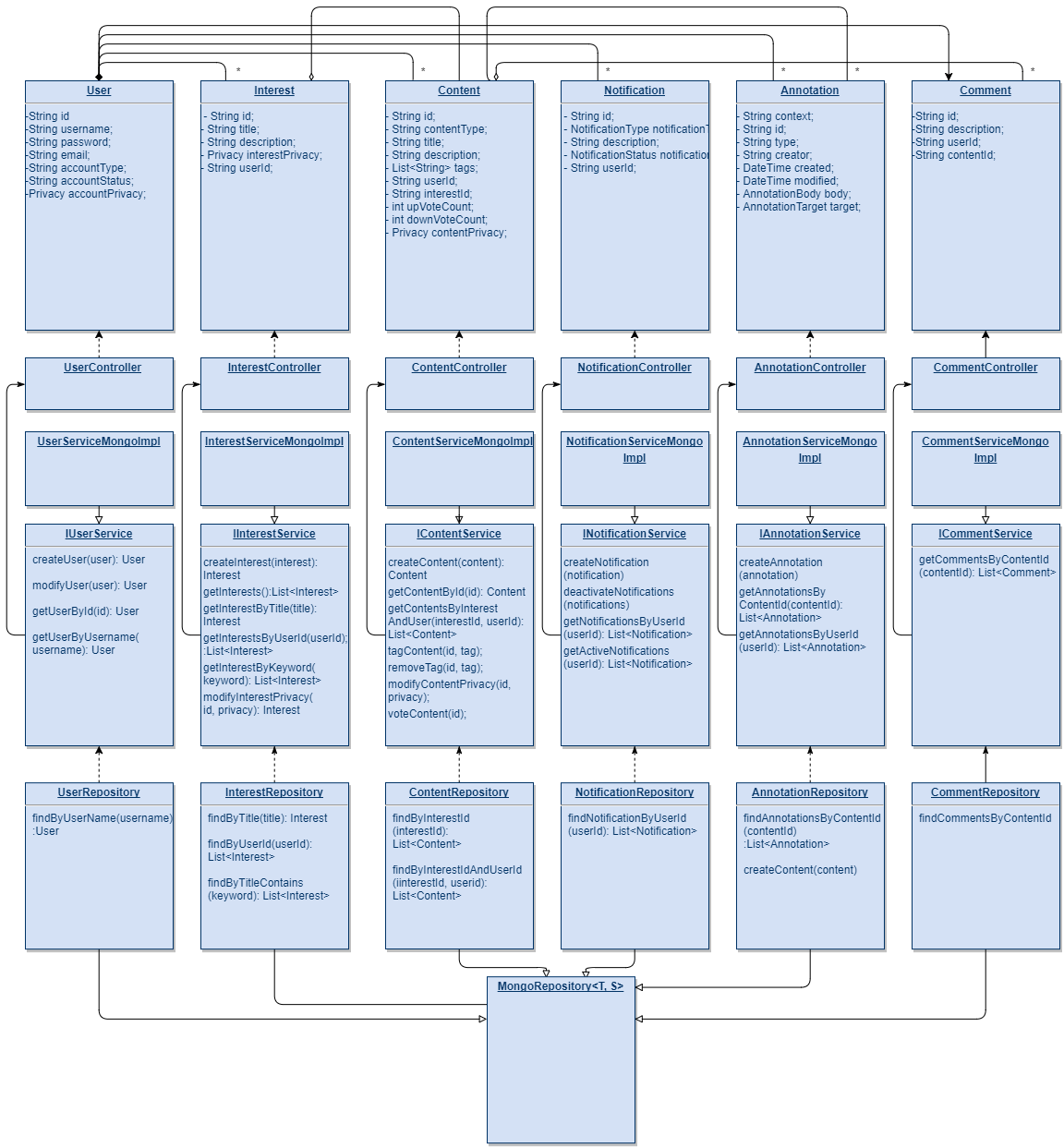
API endpoint list to be exposed for usage of client side as the following:

* Create content
* Get content by ID
* Get content by interest and user
* Tag content
* Remove tag from content
* Modify privacy of content
* Create interest
* Get all interests
* Search interest by title
* Modify privacy of interest
* Create notification
* Get notifications by user id
* Get active notifications by user id
* Deactivate notifications of user
* Create comment
* Get comments by content id

Further details is in Appendix A: Postman Collection. Any person can download Postman and import this collection to have clear idea.

### **3.4. Class Diagram**

Overall class diagram is depicted as the following:



*Class diagram*

## User Interface

Users do not have to install anything, but can reach the web application with the following link via any web browser:

http://motive-env.dyabhv3spp.us-east-1.elasticbeanstalk.com/index.html

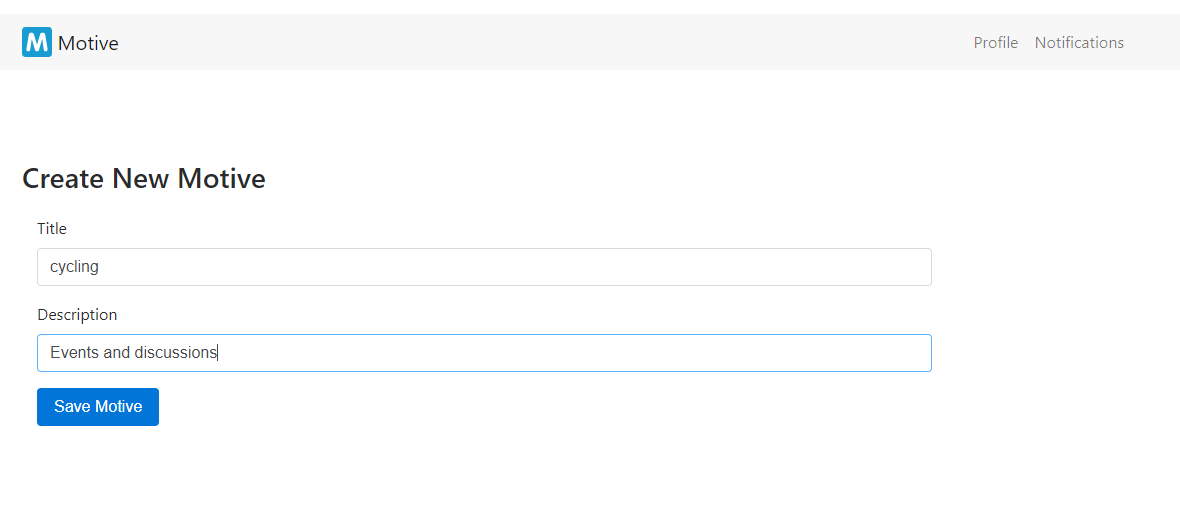
**Home Page**

In the home page, users can list motives (interest topics) and contents created for those motives. The purpose is to create content in the web application to share the ideas, events or any other content that motivates people to share.

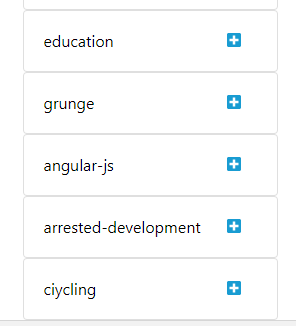
## 

**Creating New Motive**

Users can create new motive by clicking plus icon next to "Motives" header in the home page. After that, users need to fill a form.

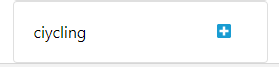


After saving, the new motive is listed in the home page.

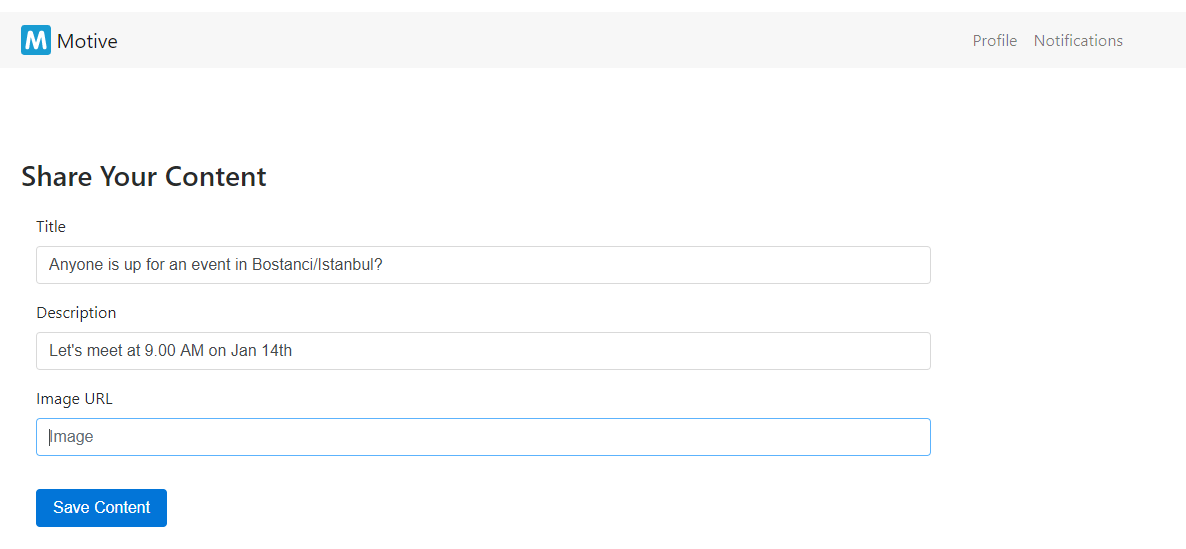


**Creating Content for the Motive**

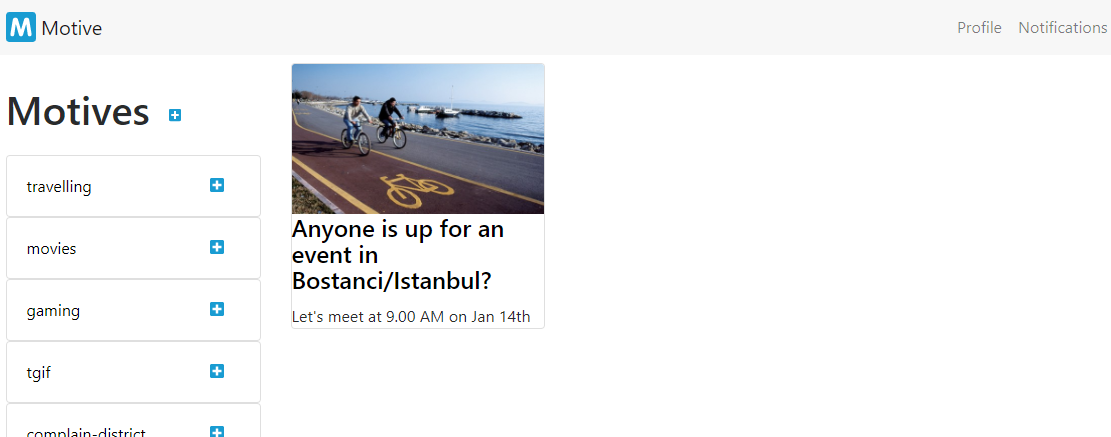
Users can add new content to motives by clicking plus icon next to motive name.



Users need to fill a form to create new content. Content fields are motive specific.

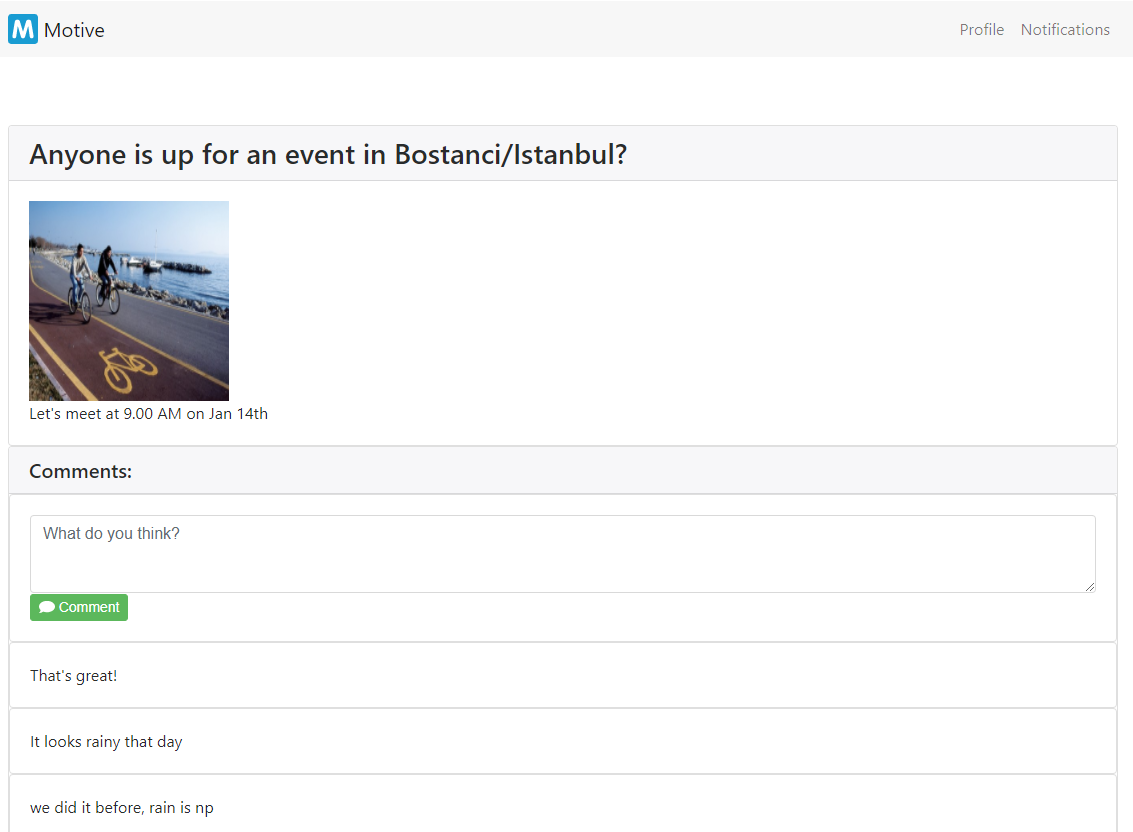


Created content is listed when the user clicks the motive.



**Viewing Content Details and Commenting**

Users can view the content details by clicking content.



Users can also comment the contents by clicking comment button under the content detail.

## Continuous Integration and Deployment

During the development of the project, continuous integration methodologies are also applied.

For the continuous integration, continuous testing is needed. To achieve this, automated unit and integration tests are also implemented besides development code.

For testing the modules and component of software, Mockito and JUnit libraries are used.

To make all the process automated, additional tool "Travis CI" is used that is integrated with the source repository, GitHub.

In GitHub repository, there are two main branches: dev and master. "master" branch is for the code that is stable and ready to be deployed and "dev" branch is for continuous development environment.

After every push to dev branch, "Travis CI" downloads the source code from the dev branch, compiles it and run the unit and integration tests. Then it notices the GitHub whether the test run is successful or not. If the process is successful, the code is mergeable to master. This provides the developer to have healthy and deployable code in any time.

Ready-to-be-deployed .jar file is build from master branch, then the source code can be manually (automated in the future) deployed to AWS Elastic Beanstalk. Further details is in Appendix B: Deployment Manual.

## APPENDIX A: Postman Collection

{

"id": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"name": "motive",

"description": "",

"order": [

"873301f2-1b1a-6d70-d63c-0900e1786dbf",

"ae85ecf2-5872-f2a6-3649-78e7e4a608ad",

"cf01ba09-35e7-a42a-544d-ec7272d111fe",

"093c72df-0815-6b68-8cf0-d95026a12edb",

"7a140fc5-2235-2e2d-db31-8a3c9b007978",

"a81e7279-8544-d750-b930-2e442061bcfe",

"a6a45886-bec2-0603-4425-4f5342c89356",

"071503be-20b0-74b8-6c5b-6376d5df1dd8",

"479d6cba-8228-0919-b072-32e603914564",

"fd7b3741-760a-bdce-c3bc-d0fab47944aa",

"7e2120fe-3346-d2bb-778f-ce010924e3b0",

"7cdb7ba6-d257-0efb-763c-3dd5fbab605f"

],

"folders": [],

"folders\_order": [],

"timestamp": 1513014226131,

"owner": 0,

"public": false,

"requests": [

{

"id": "071503be-20b0-74b8-6c5b-6376d5df1dd8",

"headers": "Content-Type: application/json\n",

"headerData": [

{

"key": "Content-Type",

"value": "application/json",

"description": "",

"enabled": true

}

],

"url": "{{base-url}}/api/v1/contents/{{contentid}}/tag/{{tag}}",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "DELETE",

"data": [],

"dataMode": "raw",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513430066521,

"name": "remove tag from content",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": [],

"rawModeData": ""

}, {

"id": "093c72df-0815-6b68-8cf0-d95026a12edb",

"headers": "",

"headerData": [],

"url": "{{base-url}}/api/v1/interests/",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "DELETE",

"data": null,

"dataMode": "params",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513429705428,

"name": "delete interest by id",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": []

},

{

"id": "479d6cba-8228-0919-b072-32e603914564",

"headers": "Content-Type: application/json\n",

"headerData": [

{

"key": "Content-Type",

"value": "application/json",

"description": "",

"enabled": true

}

],

"url": "{{base-url}}/api/v1/contents/",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "GET",

"data": [],

"dataMode": "raw",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513429737468,

"name": "get all contents",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": [],

"rawModeData": "{\n\t\"title\": \"\",\n\t\"imgPath\": \"\",\n\t\"description\": \"\",\n\t\"interestId\": \"\",\n\t\"comments\": [\"\", \"\"]\n}"

}, {

"id": "7a140fc5-2235-2e2d-db31-8a3c9b007978",

"headers": "Content-Type: application/json\n",

"headerData": [

{

"key": "Content-Type",

"value": "application/json",

"description": "",

"enabled": true

}

],

"url": "{{base-url}}/api/v1/contents/",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "POST",

"data": [],

"dataMode": "raw",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513429592990,

"name": "post content",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": [],

"rawModeData": "{\n\t\"title\": \"\",\n\t\"imgPath\": \"\",\n\t\"description\": \"\",\n\t\"interestId\": \"\",\n\t\"comments\": [\"\", \"\"]\n}"

},

{

"id": "7cdb7ba6-d257-0efb-763c-3dd5fbab605f",

"headers": "",

"headerData": [],

"url": "{{base-url}}/api/v1/contents/interest/{{interestid}}",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "DELETE",

"data": null,

"dataMode": "params",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513429876716,

"name": "delete content by id",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": []

},

{

"id": "7e2120fe-3346-d2bb-778f-ce010924e3b0",

"headers": "",

"headerData": [],

"url": "{{base-url}}/api/v1/contents/interest/{{interestid}}",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "GET",

"data": null,

"dataMode": "params",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513429765519,

"name": "get content by id",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": [],

"isFromCollection": true,

"collectionRequestId": "fd7b3741-760a-bdce-c3bc-d0fab47944aa"

},

{

"id": "873301f2-1b1a-6d70-d63c-0900e1786dbf",

"headers": "Content-Type: application/json\n",

"headerData": [

{

"key": "Content-Type",

"value": "application/json",

"description": "",

"enabled": true

}

],

"url": "{{base-url}}/api/v1/interests/",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "POST",

"data": [],

"dataMode": "raw",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513429606358,

"name": "post interest",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": [],

"rawModeData": "{\n\t\"title\": \"\",\n\t\"description\": \"\",\n\t\"properties\": []\n}"

},

{

"id": "a6a45886-bec2-0603-4425-4f5342c89356",

"headers": "Content-Type: application/json\n",

"headerData": [

{

"key": "Content-Type",

"value": "application/json",

"description": "",

"enabled": true

}

],

"url": "{{base-url}}/api/v1/contents/{{contentid}}/tag/{{tag}}",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "PUT",

"data": [],

"dataMode": "raw",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513430042606,

"name": "tag content",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": [],

"rawModeData": ""

},

{

"id": "a81e7279-8544-d750-b930-2e442061bcfe",

"headers": "Content-Type: application/json\n",

"headerData": [

{

"key": "Content-Type",

"value": "application/json",

"description": "",

"enabled": true

}

],

"url": "{{base-url}}/api/v1/contents/{{contentid}}/comment",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "PUT",

"data": [],

"dataMode": "raw",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513429930498,

"name": "comment content",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": [],

"rawModeData": "comment"

},

{

"id": "ae85ecf2-5872-f2a6-3649-78e7e4a608ad",

"headers": "",

"headerData": [],

"url": "{{base-url}}/api/v1/interests/",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "GET",

"data": null,

"dataMode": "params",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513425297887,

"name": "get all interests",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": []

},

{

"id": "cf01ba09-35e7-a42a-544d-ec7272d111fe",

"headers": "",

"headerData": [],

"url": "{{base-url}}/api/v1/interests/",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "GET",

"data": null,

"dataMode": "params",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513425297887,

"name": "get interest by id",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": [],

"isFromCollection": true,

"collectionRequestId": "ae85ecf2-5872-f2a6-3649-78e7e4a608ad"

},

{

"id": "fd7b3741-760a-bdce-c3bc-d0fab47944aa",

"headers": "",

"headerData": [],

"url": "{{base-url}}/api/v1/contents/interest/{{interestid}}",

"queryParams": [],

"preRequestScript": null,

"pathVariables": {},

"pathVariableData": [],

"method": "GET",

"data": null,

"dataMode": "params",

"tests": null,

"currentHelper": "normal",

"helperAttributes": {},

"time": 1513429765519,

"name": "get content by interest",

"description": "",

"collectionId": "f7d0d50d-5a15-3a73-4eed-e89e305e0104",

"responses": []

}

]

}

## APPENDIX B: Deployment Manual

**Prerequisite:**

PuTTy and PuTTyGen is installed (Windows)

**Steps:**

1. In project source folder, run command: gradlew bootRepackage. This will create motive-0.0.1-SNAPSHOT.jar file under <project\_root>/build/libs
2. In AWS, select Elastic Beanstalk and Create Application.
3. Select platform as Java and upload motive-0.0.1-SNAPSHOT.jar file. (application.properties having mongoDB host=localhost and port=27017)
4. Click Configure More Options -> Software and add SERVER\_PORT with value 5000 to environment variables.
5. Finish.
6. After the environment is created, you can view the home page at this point by clicking the URL in application environment page. e.g.: Elastic Beanstalk -> motive -> Motive-env

Now we need to configure MongoDB in EC2 instance:

1. First we need to SSH to EC2 instance of Elastic Beanstalk. In AWS, go to EC2 -> Network & Security
2. EC2 -> Network & Security -> Security Groups -> Select Environment's security group -> Edit inbound rules -> add custom TCP with port 22
3. Create new Key Pair. This wil download .pem file.
4. Go back to Elastic Beanstalk and select environment. Click Configuration -> Instances. Add existing key pair that has just been created.

Save private key to local:

1. Open PuTTyGen
2. Load public key pair (.pem file)
3. Save private key with extension .ppk.

SSH to EC2 instance:

1. Open PuTTy
2. Go to EC2 instance and copy IPv4 Public IP address.
3. Connect to IP address. Before connecting, select Connection -> SSH-> Auth from left panel and import .ppk file.
4. user name is ec2-user.
5. Install MongoDB to EC2. EC2 have Amazon Linux AMI running on it.
6. Add MongoDB repo: Create /etc/yum.repos.d/mongodb-org-3.0.repo file with nano /etc/yum.repos.d/mongodb-org-3.0.repo
7. Paste the following and save:

*[mongodb-org-3.0]*

*name=MongoDB Repository*

*baseurl=https://repo.mongodb.org/yum/amazon/2013.03/mongodb-org/3.0/x86\_64/*

*gpgcheck=0*

*enabled=1*

1. Install MongoDB with command *sudo yum install -y mongodb-org*
2. Start MongoDB with *sudo service mongod start*
3. Check status with *sudo service mongod status*