Developer Manual

Welcome to the Buzz Backend. This manual will guide you through setting up, testing, and the routes of the backend. It includes addition information that will be useful to you.

**Getting** **Started**

**1) Compilation and Setting up**

Make sure you are in the root directory of backend. To make sure you are in the correct directory, check if pom.xml is in the directory that you are in.

Once you are in the correct directory, please type:

mvn compile

Doing so will compile the backend.

**2) Testing the Buzz**

Once the source files are compiled, please type:

mvn test

This will run the tests in the test directory.

note: some of the tests were omitted such as createTable() to prevent the tests from removing the current database setup. Simply remove // or /\* \*/ from the database.java to run these tests as well.

**3) Deploying to local directory**

in the directory with pom.xml, please type:

mvn exec:java

This will host the program in the port that you have set to. You can check if the routes are working using curl command from the command line or you can also use the web browser of your choice to connect to the localhost.

**4) Deploying to Heroku**

From the root backend directory, please type:

mvn heroku:deploy

This will deploy the backend to the Heroku.

**Routes**

Buzz has multiple routes that can be used in the frontend to request or post the data.

GET Routes

|  |  |
| --- | --- |
| /buzz | Returns entire information of the posts stored in the database |
| /buzz/:id | Returns the entire information of the post specific to that ID |
| /buzz/:u\_id/profile | Returns all posts that the user with the user ID(primary key) has posted |
| /buzz/search\_title/:subject | Returns all posts that have the title that are similar to the (subject) |
| /buzz/search\_message/:content | Returns all posts that have the message that are similar to the content |
| /buzz/search\_author/:writer | Returns the information of that writer. |
| /buzz/search\_TM/:key | Returns all posts that have the message or title that are similar to the key without the duplicates. |
| /buzz/:id/comment | Returns all comments that are assigned to the message with the id |

POST Routes

|  |  |
| --- | --- |
| /buzz | Posts the post in the database/uploads any attachments |
| /buzz/:id/comment | Posts the comment to the post specific to that ID |
| /buzz/:u\_id/profile | Returns all posts that the user with the user ID(primary key) has posted |
| /auth | Google OAuth – Grants a session ID to the frontend. |

PUT Routes

|  |  |
| --- | --- |
| /buzz/:id/message | Edits the post in the database |
| /buzz/:c\_id/comment/edit | Edits the comment with that ID |
| /buzz/:id/upvote | Upvotes a message with the ID |
| /buzz/:id/downvote | Downvotes a message with the ID |

DELETE Routes

|  |  |
| --- | --- |
| /buzz/:id | Deletes the post in the database with that ID |

**JSON Format**

These are an example of JSON object that will be given by the backend to the frontend in the following get routes.

|  |
| --- |
| /buzz |
| /buzz/:id |
| /buzz/:u\_id/profile |
| /buzz/search\_title/:subject |
| /buzz/search\_message/:content |
| /buzz/search\_TM/:key |

{"mStatus":"ok","mData":[{"mId":31,"mTitle":"Karthick Android Test","mMessage":"testing testing","attachment":"this is for android","uId":1,"cId":0,"vote":0,"mVote":1,"date\_created":"Apr 14, 2020"}]}

These are an example of JSON object that will be given by the backend to the frontend in the following get routes.

|  |
| --- |
| /buzz/:id/comment |

{"mStatus":"ok","mData":[{"mId":0,"uId":1,"uBio":"jj","uName":"j","cId":0,"vote":0,"mVote":0]}

These are an example of JSON object that will be given by the backend to the frontend in the following get routes.

|  |
| --- |
| /buzz/search\_author/:writer |

{"mStatus":"ok","mData":[{"mId":0,"uId":1,"uBio":"jj","uName":"j","cId":0,"vote":0,"mVote":0}]}

**Curl Test**

Curl Tests are important to make sure Spark routes are working. You can test each routes by using the following command.

Link to the current backend that is deployed to Heroku:

(<https://bmw-dba.herokuapp.com/>)

Testing the GET Routes.

curl -XGET

Testing PUT Routes

curl -XPUT

Testing the POST Routes

curl -XPOST

Testing the DELETE Routes

curl -XDELETE

Useful resource for building curl commands: (<https://curlbuilder.com/>)

**Customization and Extension**

Source files that you can use to customize or improve the buzz are listed below:

pom.xml – Dependencies should be included here.

App.java – spark routes should be included here.

Database.java – SQL prepared statements and any methods that would use the database should be included in here.

Test files that you can use to customize or improve the buzz are listed below:

DatabaseTest.java – tests that checks if the database function are working correctly.

**Tech Debt**

Some tech debt exists in the backend. Here we list the tech debt that currently exists in the backend.

Because of the Tech debt in the frontend, all methods that limit the restriction to the users that are not logged in were commented out.

The files that can be posted are only empty txt files.