SW Engineering CSC 648/848 - FALL 2020

Milestone 4

Team 5 - *Project String*

Team Lead & Github Master: Jainam Shah - jshah3@mail.sfsu.edu

Frontend lead: Alfredo Diaz

Backend lead: Warren Singh

Frontend Developer: Xuanjun (Eric) Chen

Frontend Developer: Leonid Novoselov

Backend Developer: Ritesh Panta

Milestone/Version	Date
Milestone 4 v2.0	December 10, 2020
Milestone 4 v1.0	December 4, 2020
Milestone 3 v2.0	December 1, 2020
Milestone 3 v1.0	November 19, 2020
Milestone 2 v2.0	November 7, 2020
Milestone 2 v1.0	October 29, 2020
Milestone 1 v2.0	October 14, 2020
Milestone 1 v1.0	October 1, 2020

December 10, 2020

Contents

1 Product Summary

1.1 Name of the Product

String

1.2 List of Functions

1.2.1 Guest

1. Guest shall be able to view all Bands.
2. Guest shall be able to view all Band Members of a Band.
3. Guest shall be able to view all Bands' Posts.
4. Guest shall be able to view all Bands' Event List.
5. Guest shall be able to search for any Band by name.
6. Guest shall be able to filter any Bands by genre.
7. Guest shall be able to filter Bands by location.
8. Guest shall be able to register an account with a valid email address.
9. Guest shall be able to view all Bands' Repertoire List.

1.2.2 Registered User

1. Registered User shall be a Guest.
2. Registered User shall have a String Account.
3. Registered User shall be able to view their String Account information.
4. Registered User shall be able to log in using their email and password.
5. Registered User shall be able to send an invitation to apply to join a Band.
6. Registered User shall be able to create Band(s).
7. Registered User shall be able to change their password in their String Account.
8. Registered User shall be able to edit the field indicating their name in their String Account.
9. Registered User shall be able to edit the field indicating their role.

1.2.3 Band

1. Band shall be created by a Registered User.
2. Band shall have only one Band Admin.
3. The Registered User who creates the Band is set as the Band Admin.
4. Band shall have a field indicating if they are accepting invitations from Registered Users.
5. Band shall have a field indicating their geographic location.
6. Band shall have Band Post(s).
7. Band shall have an Event List indicating upcoming events.
8. Band shall have a Repertoire List.

1.2.4 Band Member

1. Band Member shall be registered users.
2. Band Member shall be able to leave their Band at any time.
3. Band Member shall be able to add Band posts to their Band.
4. Band Member shall be able to delete Band posts to their Band.
5. Band Member shall be able to add an Event Entry in the Event List of their Band.
6. Band Member shall be able to delete an Event Entry in the Event List of their Band.
7. Band Member shall be able to add a Repertoire Entry to their Band.
8. Band Member shall be able to delete a Repertoire Entry to their Band.

1.2.5 Band Admin

1. Band Admin shall be a Band member.
2. Band Admin shall be able to view all Invitations sent to their Band.
3. Band Admin shall be able to accept an Invitation sent to their Band.
4. Band Admin shall be able to reject an Invitation sent to their Band.

1.2.6 Band Post

1. Band Post shall contain a link to an image.
2. Band Post shall contain a description.
3. Band Post shall contain a title.

1.2.7 Invitations

1. Invitation shall have a message field.
2. Invitation shall have a date indicating when it was sent.
3. Invitation shall have a field indicating which Registered user sent it.
4. Invitation shall have a field indicating which Band it was sent to.
5. Accepting an Invitation shall result in adding the Registered User to the Band as a Band member.

1.2.8 Event Entry

1. Event Entry shall have a title field.
2. Event Entry shall have a location field indicating where the event will take place.
3. Event Entry shall have a date field indicating which day the event will take place.
4. Event Entry shall have a time field indicating the start time of the event.
5. Event Entry shall have a time field indicating the end time of the event.
6. Event Entry shall have a description field.

1.2.9 Repertoire Entry

1. Repertoire Entry shall have a field indicating the song's name.
2. Repertoire Entry shall have a field indicating the song's genre.
3. Repertoire Entry shall have a field indicating the song's run time.

1.3 Product Summary (contains unique value proposition)

String allows small and independent bands & artists to create, grow, and communicate with their local fan base. String is unique because it enables anyone to search for bands & music events nearby, browse and listen to band portfolios directly on the page, bands can manage events and their music repertoire, String also serves as an informational medium for fans to access posts and keep a close look on their favorite band's media page, fans can access the upcoming and past events and see what songs the band will play. It's also a platform for musicians to find and join bands by directly sending an invitation to bands who are recruiting.

1.4 Product URL

https://code-404.xyz

2 Usability Test Plan

2.1 List of functions

1. Band Search
2. Band Filter
3. Creating a Band
4. Uploading pictures
5. Request to join a band

2.2 Test Objectives

To determine whether the website is intuitive, and easy to use. Test tasks were constructed so that they correlate with the main Tested Functions. The Test contained following types of tasks:

First impression questions - Whether the website landing page was self explanatory, and authentication flow was easy enough for regular users to use.

Exploratory tasks - Whether it was easy to search for bands and events. Whether search filters were sufficient for regular users' needs.

Directed tasks -Whether creating a band is easy enough. Whether band-event search, and related features work properly. These two tasks require users to follow directions set by our team.

2.3 Test Setup

System Setup

Link to the application was given to String team's friends. Each of the participants completed the tasks on their own device.

Starting Point

Participants were given the list of tasks to complete. After the tasks were completed, participants were asked to complete a questionnaire. Note, total number of participants 7, who used PC and 3 who used mobile.

Intended Users

Participants of the tests have the same age group, and location as intended users of String platform.

URLs to be Tested

Navigation intuitiveness: `https://code-404.xyz`

2.4 Tasks Description

Following the provided link, navigate to the website in the browser on your device. Authenticate on the website, and try to filter bands by genre and search for a band you might be interested in. Once found, check out bands information. Find how you can join this band. Then go to your profile and try to create a band and upload band profile picture.

2.5 Usability Test Table

Test/use case	% completed	Errors	Comments
Band Search	100%	None	Suggest names while typing for faster search
Band Filter	90%	Doesn't reset when new search is performed	Only small variety of genres
Creating a Band	50%	Does not identify which field is incorrect	Button to create hard to find (unobvious)
Uploading Picture	100%	Some browsers fail	Some formats/sizes rejected
Request to join a band	95%	None	Very easy to access

2.6 Questionnaire

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Navigation around the website was easy and intuitive	0	1	2	3	4
Filtering was sufficient for my needs	0	1	2	3	4
Search worked as expected	0	1	2	3	4
It was easy to find how to join a band	0	1	2	3	4
It was easy to create a band	0	1	2	3	4
Band profile picture uploading worked as intended	0	1	2	3	4
Both mobile and desktop version looked pleasant	0	1	2	3	4
Equal performance of website on desktop and mobile	0	1	2	3	4

2.7 Responses

Total number of respondents: 10

Category	**Result (marks out of 40)**
Navigation around the website was easy and intuitive	32
Filtering was sufficient for my needs	25
Search worked as expected	40
It was easy to find how to join a band	36
It was easy to create a band	23
Band profile picture uploading worked as intended	19
Both mobile and desktop version looked pleasant	31
Equal performance of website on desktop and mobile	29

3 QA Test Plan

3.1 QA Test (1)

3.1.1 Category and Non-functional Requirement Being Tested

Compatibility

Website must be compatible with Safari 12.0+, Chrome 81+, Firefox 70+

3.1.2 Test Objectives

Compatibility non-functional requirement (1): To ensure website renders correctly on major browsers, while specifying versions.

With more time, this would be automated via a Selenium IDE or Selenium WebDriver scripted test suite. Due to the exigencies of time and lack of team experience, manual testing was used.

3.1.3 Hardware Setup

Consumer level computer setup (no loss of generality).

3.1.4 Software Setup

Safari 14.0.1 (16610.2.11.51.8), Chrome v87.0.4280.88 (Official Build) (64-bit), Firefox v83.0 (64-bit)

URL: `https://code-404.xyz/`

3.1.5 Feature to be Tested

Website compatibility with major browsers, version floor specified.

3.1.6 QA Test Results

#	Test Title	Test Description	Test Input	Expected Correct Output	Result
1	Safari	Homepage renders, can sign in	Browser directed to URL, sign in details provided	Renders and Log-in resolves	PASS
2	Chrome	Homepage renders, can sign in	Browser directed to URL, sign in details provided	Renders and Log-in resolves	PASS
3	Firefox	Homepage renders, can sign in	Browser directed to URL, sign in details provided	Renders and Log-in resolves	PASS

3.2 QA Test (2)

3.2.1 Category and Non-functional Requirement Being Tested

Storage and Data Integrity

Images uploaded by users should have file size less than 8MB.

3.2.2 Test Objectives

Images which fit the size requirement successfully upload; images which violate them do not.

With more time, this would be automated via a Selenium IDE or Selenium WebDriver scripted test suite. Due to the exigencies of time and lack of team experience, manual testing was used.

3.2.3 Hardware Setup

Consumer level computer setup (no loss of generality).

3.2.4 Software Setup

Safari 14.0.1 (16610.2.11.51.8), Chrome v87.0.4280.88 (Official Build) (64-bit), Firefox v83.0 (64-bit)

URL: `https://code-404.xyz/profile`

Once there and signed in, plus sign is clicked in order to create a band.

3.2.5 Feature to be Tested

Image size check

3.2.6 QA Test Results

#	Test Title	Test Description	Test Input	Expected Correct Output	Result
1	Safari, correct size	Band creation attempt with correct Image size	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS
2	Safari, too large, correct size	Band creation attempt with incorrect Image size	Browser directed to URL, sign in details provided, Band creation with image attempted	Band not created successfully	FAIL
3	Chrome, correct size	Band creation attempt with correct Image size	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS

4	Chrome, too large, correct size	Band creation attempt with incorrect Image size	Browser directed to URL, sign in details provided, Band creation with image attempted	Band not created successfully	FAIL
5	Firefox, correct size	Band creation attempt with correct Image size	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS
6	Firefox, too large, correct size	Band creation attempt with incorrect Image size	Browser directed to URL, sign in details provided, Band creation with image attempted	Band not created successfully	FAIL

After investigation, the image size checking function was discovered to not be on the version which is on the deployment branch. Further testing after merging branch with origin and redeploying planned to verify correct outcome.

3.3 QA Test (3)

3.3.1 Category and Non-functional Requirement Being Tested

Storage and Data Integrity

Only images with .jpg, jpeg, or .png extensions are accepted as uploads.

3.3.2 Test Objectives

Images in the correct format are uploaded successfully, incorrectly formatted images are not.

With more time, this would be automated via a Selenium IDE or Selenium WebDriver scripted test suite. Due to the exigencies of time and lack of team experience, manual testing was used.

3.3.3 Hardware Setup

Consumer level computer setup (no loss of generality).

3.3.4 Software Setup

Safari 14.0.1 (16610.2.11.51.8), Chrome v87.0.4280.88 (Official Build) (64-bit), Firefox v83.0 (64-bit)

URL: `https://code-404.xyz/profile`

Once there and signed in, plus sign is clicked in order to create a band.

3.3.5 Feature to be Tested

Only correctly formatted images are uploaded

3.3.6 QA Test Results

#	Test Title	Test Description	Test Input	Expected Correct Output	Result
1	Safari, jpg format	Band creation attempt with correct Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS
2	Safari, jpeg format	Band creation attempt with correct Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS
3	Safari, png format	Band creation attempt with correct Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS
4	Safari, tiff format	Band creation attempt with incorrect Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band not created successfully	FAIL

5	Safari, ico format	Band creation attempt with incorrect Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band not created successfully	FAIL
6	Chrome, jpg format	Band creation attempt with correct Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS
7	Chrome, jpeg format	Band creation attempt with correct Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS
8	Chrome, png format	Band creation attempt with correct Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS

9	Chrome, tiff format	Band creation attempt with incorrect Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band not created successfully	FAIL
10	Chrome, ico format	Band creation attempt with incorrect Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band not created successfully	FAIL
11	Firefox, jpg format	Band creation attempt with correct Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS
12	Firefox, jpeg format	Band creation attempt with correct Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS

13	Firefox, png format	Band creation attempt with correct Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band created successfully	PASS
14	Firefox, tiff format	Band creation attempt with incorrect Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band not created successfully	FAIL
15	Firefox, ico format	Band creation attempt with incorrect Image format	Browser directed to URL, sign in details provided, Band creation with image attempted	Band not created successfully	FAIL

After investigation, the image format checking function was discovered to not be on the version which is on the deployment branch. Further testing after merging branch with origin and redeploying planned to verify correct outcome.

3.4 QA Test (4)

3.4.1 Category and Non-functional Requirement Being Tested

Environment

Node versions 12-13 and NPM versions 6.14 should be used to develop the application.

3.4.2 Test Objectives

Verify that proper development environment is present and utilized.

3.4.3 Hardware Setup

Consumer level computer setup (no loss of generality).

3.4.4 Software Setup

GNU bash, version 3.2.57(1)-release (x86_64-apple-darwin20) (Copyright (C) 2007 Free Software Foundation, Inc.)

URL: csc-648-848-04-jose-fall-2020-05/application/app/tests/nodeVersion.sh

Shell scripts stored in the tests folder, run with "npm run nodeTest".

3.4.5 Feature to be Tested

Node and NPM versions.

3.4.6 QA Test Results

#	Test Title	Test Description	Test Input	Expected Correct Output	Result
1	nodeTest	execute nodeVersion shell script	"npm run nodeTest" from the command line	"All Node and NPM versions current for project standards"	PASS

3.5 QA Test (5)

3.5.1 Category and Non-functional Requirement Being Tested

Coding Standards

VS Code version 1.29.0+ should be used as the code editor for the project.

3.5.2 Test Objectives

VS Code version verified above the "floor" designated for project.

3.5.3 Hardware Setup

Consumer level computer setup (no loss of generality).

3.5.4 Software Setup

GNU bash, version 3.2.57(1)-release (x86_64-apple-darwin20) (Copyright (C) 2007 Free Software Foundation, Inc.)

URL: csc-648-848-04-jose-fall-2020-05/application/app/tests/nodeVersion.sh

Shell scripts stored in the tests folder, run with "npm run vscodeVersion".

3.5.5 Feature to be Tested

VS Code (Project IDE) version verification.

3.5.6 QA Test Results

#	Test Title	Test Description	Test Input	Expected Correct Output	Result
1	vscodeVersion	execute vscodeVersion shell script	"npm run vscodeVersion" from the command line in appropriate directory	"VS Code version current for project standards"	PASS

4 Code Review

4.1 Introduction

1. We are using Controller Pattern for our backend and our frontend is a react app with client side routing. For coding we used the ES6 syntax.

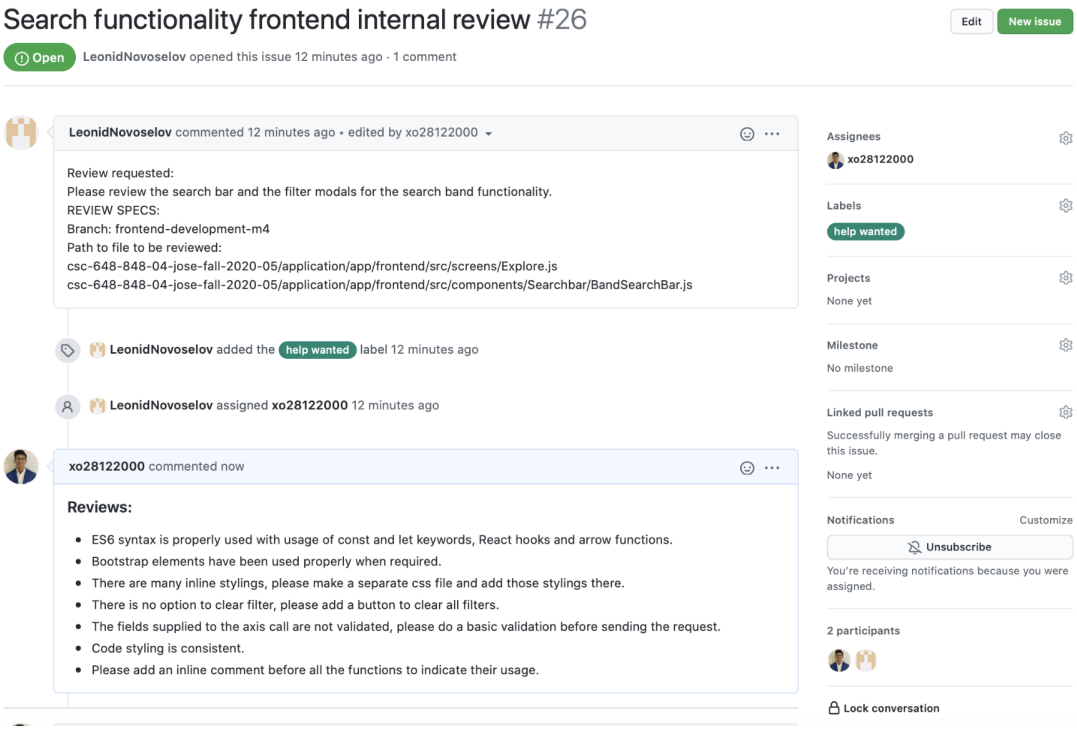
2. We will have our Search functionality reviewed.

   (a) For an internal review, Frontend Developer Leonid sent code to Team Lead Jainam for comment and review.

   (b) In another internal review, Backend Developer Ritesh sent code to Backend Lead Warren for comment and review.

   (c) For the external review, the team lead will send the backend handling code for search functionality to Nimiksha (Team 2)

4.2 Internal Reviews



Backend Review #27

Closed **riteshcode9** opened this issue 1 hour ago · 1 comment

riteshcode9 commented 1 hour ago

Can you please review some of the routes that I created inside the backend branch?
Path to the file:
csc-648-848-04-jose-fall-2020-05/application/app/backend/routes/band/index.js
csc-648-848-04-jose-fall-202005/application/app/backend/controllers/bandControllers/bandController.js

riteshcode9 added the bug label 1 hour ago

riteshcode9 added this to the **backend-development-m4** milestone 1 hour ago

riteshcode9 assigned **wsmarshall** 1 hour ago

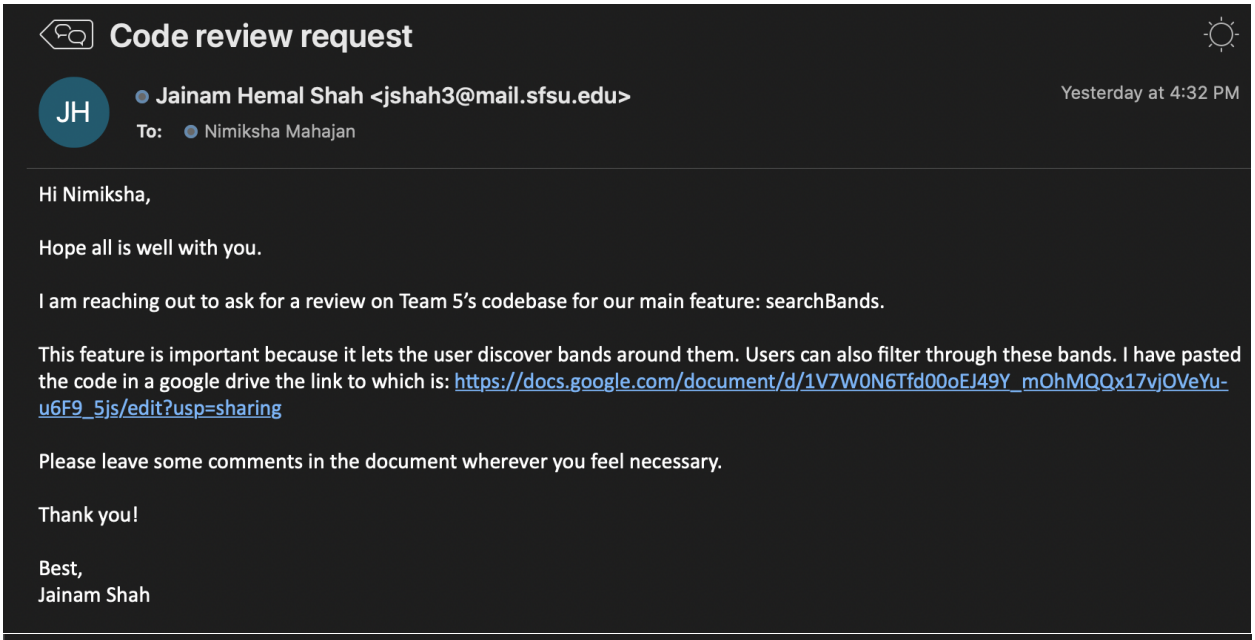
wsmarshall commented now

Looks good so far. Couple of notes:

- Route testing should be done in Postman, with all routes saved for ease of re-use
- Functions should have brief explanatory comments for increased readability
- Formatting looks good and consistent

wsmarshall closed this now

4.3 External Review



Code review request

JH

Jainam Hemal Shah <jshah3@mail.sfsu.edu>

To: Nimiksha Mahajan

Yesterday at 4:32 PM

Hi Nimiksha,

Hope all is well with you.

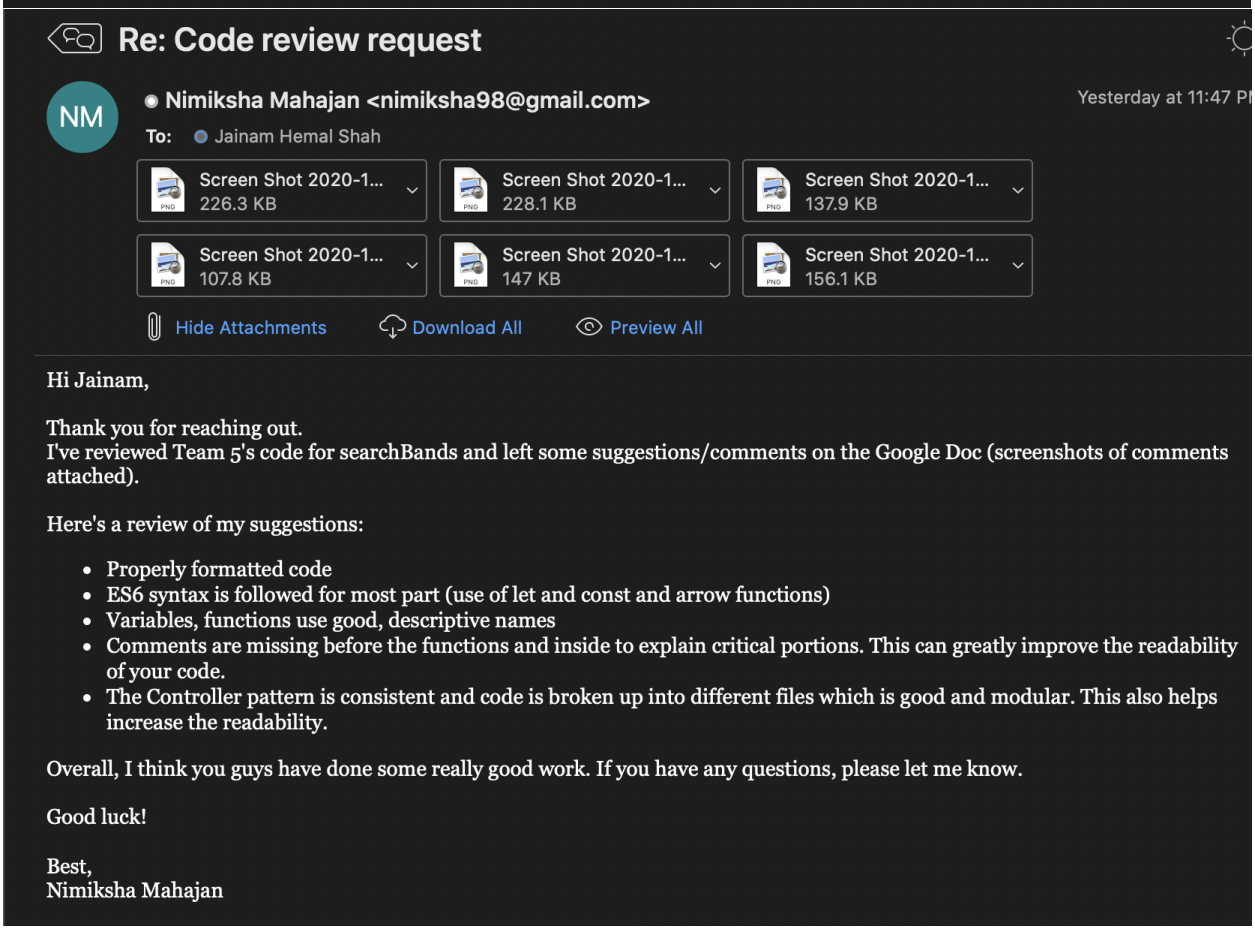
I am reaching out to ask for a review on Team 5's codebase for our main feature: searchBands.

This feature is important because it lets the user discover bands around them. Users can also filter through these bands. I have pasted the code in a google drive the link to which is: https://docs.google.com/document/d/1V7W0N6Tfd00oEJ49Y_mOhMQQx17vjOVeYu-u6F9_5js/edit?usp=sharing

Please leave some comments in the document wherever you feel necessary.

Thank you!

Best,
Jainam Shah



Re: Code review request

NM

Nimiksha Mahajan <nimiksha98@gmail.com>

To: Jainam Hemal Shah

Yesterday at 11:47 PM

Screen Shot 2020-1... 226.3 KB

Screen Shot 2020-1... 228.1 KB

Screen Shot 2020-1... 137.9 KB

Screen Shot 2020-1... 107.8 KB

Screen Shot 2020-1... 147 KB

Screen Shot 2020-1... 156.1 KB

Hide Attachments Download All Preview All

Hi Jainam,

Thank you for reaching out.
I've reviewed Team 5's code for searchBands and left some suggestions/comments on the Google Doc (screenshots of comments attached).

Here's a review of my suggestions:

- Properly formatted code
- ES6 syntax is followed for most part (use of let and const and arrow functions)
- Variables, functions use good, descriptive names
- Comments are missing before the functions and inside to explain critical portions. This can greatly improve the readability of your code.
- The Controller pattern is consistent and code is broken up into different files which is good and modular. This also helps increase the readability.

Overall, I think you guys have done some really good work. If you have any questions, please let me know.

Good luck!

Best,
Nimiksha Mahajan

Route being reviewed: `searchBands`

app/backend/routes/band/index.js

```
const express = require("express");

const nodeGeocoder = require("node-geocoder");
const geoCoder = nodeGeocoder({
 provider: "openstreetmap",
});

const bandController = require("../../controllers/bandController/bandController.js");
const isUser = require("../../helpers/middlewares/isUser");
const awsS3 = require("../../lib/aws/s3");

const multer = require("multer");
const geocode = require("../../helpers/middlewares/geocode.js");
const storage = multer.diskStorage({
 destination: "backend/uploads",
 filename: function (req, file, cb) {
   cb(null, file.originalname);
 },
});
const upload = multer({ storage: storage });

let bandRouter = express.Router();

// some lines here are neglected which do not pertain to searchBands route
bandRouter.post("/searchBands", geocode, bandController.searchBands);

module.exports = bandRouter;
```

app/backend/helper/middlewares/geocode.js

```
// should run the location library and then
// change the req.body.lat, req.body.long to what the function returns
```

```
};
*/

module.exports = geocode;
```

app/backend/controllers/bandController/bandController.js

```
const bandQueries = require("../../db/queries/band.js");
const awsS3 = require("../../lib/aws/s3");
const isUser = require("../../helpers/middlewares/isUser.js");

const searchBands = (req, res) => {
 var search = {
   name: req.body.name ? req.body.name + "%" : "%",
   genre: req.body.genre ? req.body.genre : "%",
   locationLat: req.body.locationLat ? req.body.locationLat : null,
   locationLong: req.body.locationLong ? req.body.locationLong : null,
   isLookingForMember: req.body.isLookingForMember
     ? req.body.isLookingForMember
     : 0,
 };

 bandQueries
   .searchBands(
     search.name,
     search.genre,
     search.locationLat,
     search.locationLong,
     search.isLookingForMember
   )
   .then((retObj) => {
     return res.send({ success: true, result: retObj });
   })
```

Nimiksha Mahajan
9:33 PM Today

middlewares are not usually used in controllers, they are generally used while defining routes.

Nimiksha Mahajan
9:33 PM Today

goodwork with naming everything. it's descriptive

Nimiksha Mahajan
9:33 PM Today

it would be helpful if you can add comments describing this variable. Also because you are using ES6 syntax, please change this variable to let.

```
const pool = require("../index");
let bandQueries = {};

//TODO figure out difference between filename/imgUrl, if any (and proper use)

bandQueries.searchBands = (
 name,
 genre,
 locationLat,
 locationLong,
 isLookingForMember
) => {
 if (!locationLat || !locationLong) {
   //no location provided
   return new Promise((resolve, reject) => {
     //console.log("reached no location search query");
     pool.query(
       `SELECT * from BAND WHERE (name LIKE '${name}' AND genre LIKE '${genre}'AND
isLookingForMember >= ${isLookingForMember})`,
       (err, results) => {
         if (err) {
           return reject(err);
         } else {
           //console.log("no errors");
           //console.log(results);
           return resolve(results);
         }
       }
```

Nimiksha Mahajan
9:34 PM Today

good explanatory comment.

```
/** //this is the old version
const geocode = async (req, res, next) => {
 if (req.body.location) {
   try {
     const retObj = await geoCoder.geocode({
       address: req.body.location.street,
       city: req.body.location.city,
       state: req.body.location.state,
       zipcode: req.body.location.zip,
       country: "United States",
     });
     req.body.latitude = retObj[0].latitude;
     req.body.longitude = retObj[0].longitude;
     next();
```

Nimiksha Mahajan
9:33 PM Today

I think it is a is good idea to keep this. Old code might be needed in the future for reference. It helps that the comments label this as old code, so it's not confusing

```
const geoCoder = nodeGeocoder({
 provider: "openstreetmap",
});

const geocode = async (req, res, next) => {
 if (req.body.location) {
   try {
     const retObj = await geoCoder.geocode({
       street: req.body.location.street,
       city: req.body.location.city,
       state: req.body.location.state,
       postalcode: req.body.location.zip,
       country: "United States",
     });
     if (retObj.length > 0) {
       req.body.locationLat = retObj[0].latitude;
       req.body.locationLong = retObj[0].longitude;
     }
```

Nimiksha Mahajan
9:32 PM Today

the openstreetmap provider seems to be only a testing library and is not recommended for production. I'd recommend changing it.

Nimiksha Mahajan
9:32 PM Today

it would be helpful to use comments before the function to describe what it does

```
     );
   });
 } else {
   //if they provide a location (locationLat, locationLong found in band controller calling function)
   return new Promise((resolve, reject) => {
    pool.query(`select *, POWER( SIN( ((locationLat-${locationLat})*0.01745329252)/2 ), 2) + COS( ${locationLat} * 0.01745329252 ) * COS( locationLat * 0.01745329252 ) * POWER( SIN( ((locationLong-${locationLong})*0.01745329252)/2 ), 2) AS temp from BAND WHERE (name LIKE '${name}' AND genre LIKE '${genre}'AND isLookingForMember >= ${isLookingForMember}) order by (6371 * 2 * ATAN2( SQRT(temp), SQRT(1-temp) ));`,
       (err, results) => {
         if (err) {
           return reject(err);
         } else {
           return resolve(results);
         }
       }
     );
   });
 }
};

module.exports = bandQueries;
```

Nimiksha Mahajan
9:34 PM Today

A comment describing this will be helpful. I think this is part is calculating and sorting by distance between two given coordinates.
It is very technical and so a comment should be present to explain it.

5 Self-check on Best Practices for Security

5.1 Major Assets under protection

1. Passwords - by storing hashed passwords
2. Image assets - by setting s3 bucket to public read only
3. User data - by not logging any data on the client or server side and by using passport encrypted sessions

5.2 Encrypted Passwords in the DB

We encrypt password using bcryptjs `https://www.npmjs.com/package/bcryptjs` functions hashSync to create a hashed password from the raw text password when registering and compare the raw text password using compareSync when logging in.

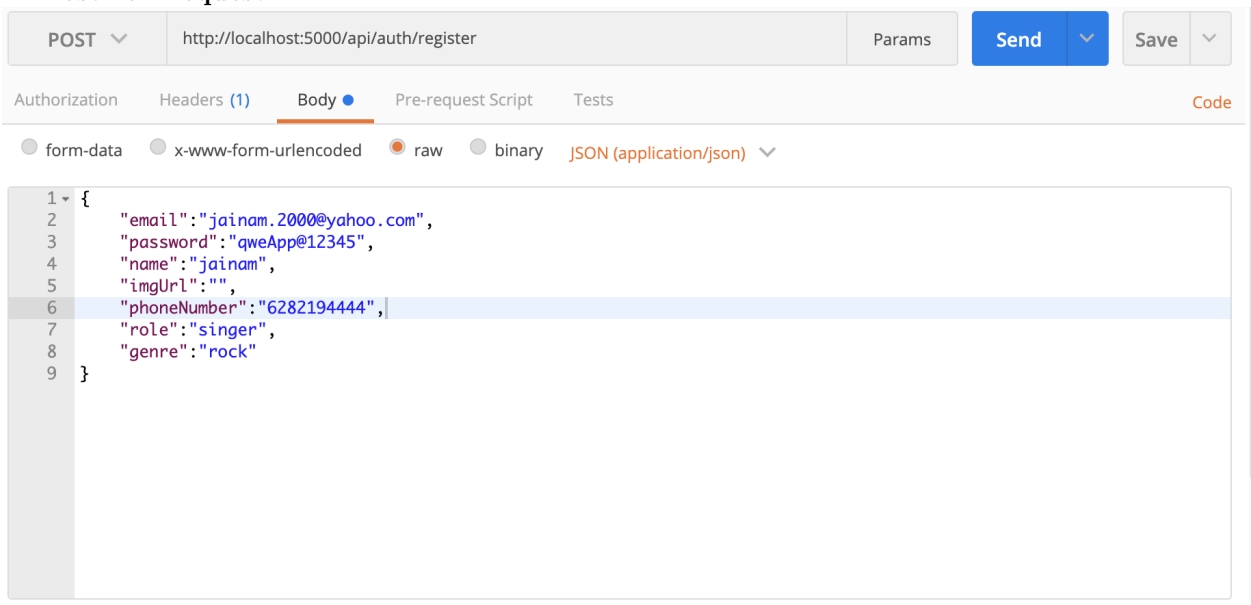
Register function:

```
function(req, email, password, done) {
  const hashedPassword = bcrypt.hashSync(password, 12);
  stringAccountQueries
    .register(
      email,
      hashedPassword,
      req.body.name,
      req.body.imgUrl,
      req.body.phoneNumber,
      JSON.stringify(req.body.location),
      req.body.locationLat,
      req.body.locationLong,
      req.body.role,
      req.body.genre
    )
    .then(data => {
      if (data.affectedRows !== 1) {
        return done(null, false, {
          message: "unable to create user"
        });
      }
      delete req.body.password;
      return done(null, req.body);
    })
    .catch(err => {
      return done(err.sqlMessage);
    });
```

Login function:

```
function(req, email, password, done) {
  stringAccountQueries
    .login(email)
    .then(data => {
      if (
        data &&
        data.length === 1 &&
        !bcrypt.compareSync(password, data.password)
      ) {
        // check password
        delete data.password;
        return done(null, {
          ...data[0],
          location: JSON.parse(data[0].location)
        });
      } else {
        return done(null, false, { message: "Incorrect field" });
      }
    })
    .catch(err => {
      return done(err.sqlMessage);
    });
  }
),
```

Postman request:



Corresponding database entry in mySQL:

```
mysql> select * from STRINGACCOUNT;
+--------+---------------------+--------------------------------------------------------------+--------+-----------------+-------------+----------+-------------+--------------+--------+-------+
| userId | email               | password                                                     | name   | profileImageUrl | phoneNumber | location | locationLat | locationLong | role   | genre |
+--------+---------------------+--------------------------------------------------------------+--------+-----------------+-------------+----------+-------------+--------------+--------+-------+
|      1 | jainam.2000@yahoo.com | $2a$12$p1VAgZ.7xitljGnIx.MYaOic7p.cB8HoPxnLmrJhTUTsdT.3hSQVG | jainam |                 | 6282194444  | NULL     |        NULL |         NULL | singer | rock  |
+--------+---------------------+--------------------------------------------------------------+--------+-----------------+-------------+----------+-------------+--------------+--------+-------+
```

5.3 Data Validation during Registration

1. Name - contain 1 - 30 characters
2. Email - should be valid form
3. Password - should be 8-21 characters long, with at least 1 number 1 uppercase letter and 1 lowercase letter
4. Location - if supplied, it should be a valid location in United States

Name, email, and password validation:

```
if (!registerName || !registerEmail || !registerPassword) {
  alert("Name, Email and Password are required fields");
} else if (
  registerName.length > 30 ||
  registerName.replace(/[^a-zA-Z ]/g, "").length < 1 ||
  registerName.replace(/[^a-zA-Z ]/g, "").length > 30
) {
  alert("Name should contain atleast one and atmost 30 characters");
} else if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(registerEmail)) {
  alert("Please enter a valid email");
} else if (
  !/(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,21}/.test(registerPassword)
) {
  alert(
    `Please enter a valid password containing 8 to 21 characters
    containing atleast 1 number, 1 uppercase and 1 lowercase letter.`
  );
} else {
```

Location validation using geoCoder's geocode function:

```
async (req, res, next) => {
  if (req.body.location) {
    try {
      const retObj = await geoCoder.geocode({
        street: req.body.location.street,
        city: req.body.location.city,
        state: req.body.location.state,
        postalcode: req.body.location.zip,
        country: "United States"
      });
      if (retObj.length > 0) {
        req.body.locationLat = retObj[0].latitude;
        req.body.locationLong = retObj[0].longitude;
      }

      next();
    } catch (err) {
      return res.send({ success: false, error: "geolocation error" });
    }
  } else {
    next();
  }
```

6 Self-check: Adherence to Original Non-functional Specifications

Key: All unmarked items are "DONE".

- Scalability
  1. The website should be able to host at least 300 users in the same hosted region as the EC2 instance.
  2. A load balancer shall not be used while deployment.
- Compatibility
  1. Website must be compatible with Safari 12.0+, Chrome 81+, Firefox 70+
  2. Website should be compatible on mobile with a minimum width of 300px and desktop with any dimensions.
- Security
  1. All passwords should be hashed using bcrypt algorithm before storing it in the database.
  2. Passwords set by users should be at least 8 characters long. (Backend, in progress)
  3. Passwords set by users should be at most 21 characters long. (Backend, in progress)
  4. Passwords should contain at least one uppercase letter. (Backend, in progress)
  5. Passwords should contain at least one number. (Backend, in progress)
  6. Passwords should not be logged in the production build.
  7. Any private keys including but not limited to Mysql database password should be pushed to github.
  8. Website should always have SSL encryption enabled.
  9. Login api calls shall be limited to 20 calls per ip each hour.
  10. Sign up api calls shall be limited to 20 calls per ip each hour.
  11. Application shall be secured from sql injection attacks. (Backend, in progress)
  12. Application shall be secured against HTTP Parameter Pollution attacks.
- Network and Deployment capabilities
  1. Application should be deployed on AWS EC2 instance.
  2. Only the master branch of github should be deployed on the AWS server instance.

3. Any and all deployment should be handled only by the DevOps Lead.

- Legal

1. Privacy policy and terms and conditions of use should be accepted by the user before creating an account on the (Frontend, in progress)
2. The user generated data collected by the website shall not be shared with any third party without user consent.

- Documentation

1. All api shall be well documented on a high level usage view. (Backend, in progress)
2. All api shall be well documented in detail with implementation details. (Dropped due to time constraints)

- UI/UX

1. All pages on the website should have consistent design and color.
2. Color combination for the website should be calming and should not cause strain on eyes.
3. All text in the page should be readable with a minimum font size of 8px and should have contrasting font than the background color.
4. All fonts used shall be web safe fonts.
5. Users should be able to navigate between pages.

- Coding standards

1. VS code version 1.29.0+ should be used as the code editor for the project.
2. VS code extension Prettier version 2.2+ should be enabled ands used to format all documents before pushing any commits to github.
3. Code shall have proper indentation and spacing.
4. Code should have error handling to prevent failure.
5. Functions should have at least one comment explaining its functionality at the beginning of the function. (Backend, in progress)

- Environment

1. Node version 12-13 and npm version 6.14.8 should be used to develop the application.

- Internalization

    1. English should be the language used in the site.
    2. All locations on the website should be validated to be in the United States.
    3. Dollar should be the only currency accepted in all tractions on the website.
    4. Distance should be measured in Miles.
    5. Time should be measured only in HST, AKDT, PDT, MST, MDT, CDT and EDT time zones.

- Storage and data integtity

    1. Videos uploaded by users should be less than 20 frames per second.
    2. Videos uploaded by users should have length less than 120 seconds.
    3. Videos uploaded by users should have file size less than 200mb.
    4. Videos should have .mp4 or .mpv extension.
    5. Images uploaded by users should have file size less than 8mb. (Frontend and Backend, in progress)
    6. Images with .jpg, .jpeg or .png extensions only are accepted. (Frontend and Backend, in progress)

7 List of contributions to the three parts of this milestone

- Jainam Shah: (Team Lead)
    1. Worked on setting up bi-weekly checkins and distributed tasks and setup schedule to get the M4 document completed and reviewed.
    2. Worked with Leonid and Nimiksha to complete an internal and an external code review.
    3. Worked to find and add security best practices to our application and reported them in the document.
    4. Worked to check if the current project is adhering to original Non-functional requirements and reported them.
- Alfredo Diaz: (Frontend lead)
    1. Worked on polishing the product summary for M4 document.
    2. Worked on setting up and completing the Usability test plan.
    3. Worked to collect feedback from actual users for the Usability test plan.
    4. Attended meetings regularly and maintained proper communication through slack.
- Leonid Novoselov: (Frontend developer)
    1. Worked on getting frontend code reviewed by Jainam.
    2. Worked on setting up and completing the Usability test plan.
    3. Worked to collect feedback from actual users for the Usability test plan.
    4. Attended meetings regularly and maintained proper communication through slack.
- Eric Chen: (Frontend developer)
    1. No contributions.
- Warren Singh: (Backend lead)
    1. Worked on setting up and typesetting the Latex document for M4.
    2. Worked on setting up and completing the QA test plan.
    3. Worked on finding optimal ways of automation QA testing.
    4. Attended meetings regularly and maintained proper communication through slack.

- Ritesh Panta: (Backend developer)

    1. Worked on setting up and completing the QA test plan.
    2. Worked on finding optimal ways of automation QA testing.
    3. Attended meetings regularly and maintained proper communication through slack.