SW Engineering CSC 648/848 - Section 04 - FALL 2020

Final Project

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

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

Milestone/Version	Date
Milestone 5 v1.0	Dec 17, 2020
Milestone 4 v2.0	Dec 14, 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 17, 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 Milestone Documents

2.1 Milestone 1

SW Engineering CSC 648/848 - FALL 2020 - Team 5 - Milestone 1

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 1 v1.1	10/01/2020
Milestone 1 v2.1	10/14/2020

December 8, 2020

Contents

1 Executive Summary

String is a one stop platform for local bands, individual musicians, and listeners. Bands which are ready to perform and have upcoming performances can list upcoming events on their profile page and be found by a bigger audience. String is also aiming to change the landscape for listeners who like live music and how they spend their leisure time. Our platform being specific to local bands lets tourists and locals discover music and genres directly from the platform. They won't have to go to the bar and hear a random musician or a band anymore. With String they can pick and choose who and when they want to listen to in person and discover new bars, restaurants or cuisines while listening to the music they enjoy.

Our biggest competitors are reverbnation and Bandlab. These platforms offer lots of features but are generalised, and built for remote collaboration and exposure. We are also able to offer more specialized services to musicians aiming for in-person rehearsals, collaborations, and performances, aiming to give them the ability to do everything at a singular place. Local artists gain huge value from String.

According to IFPI(https://www.ifpi.org/our-industry/industry-data/) the total value of the recording industry in 2019 was US$20.2bn and reported an 8.2% global growth (that's +18.9% for Latin america alone) - As we explore and experiment with monetization we look at paid discovery options for bands, and brands as possible consumption based options that bands, music equipment brands and even restaurant joints could access. We are also exploring subscription based options where musicians and bands would pay a recurrent amount each month to keep track of their audience, this initially can be simple metrics on the people looking at their profiles but also later can be grown to integrate to the rest of their audiences through streaming platforms hosting their music.

There is no beating the benefit of in person connection, nor the value that a specfic location can bring. String capitalizes on that, and offers a unique new value as a result.

2 Main Use Cases

Title	A musician looking for a band to join
Actor	John Lennon
Action	John is a guitarist and singer from San Francisco just arrived in LA and is looking for a band who is open to playing the same type of music with him. Due to corona there are no live social gatherings to meet other musicians in the area. John signs up on String, he is then able to upload some of his content and post a request to join a band. Within minutes he has bands messaging him to send him samples and test out team dynamics. Within a week he already made new friends and signed into a band. Bands add him to their band page and they start playing music together.

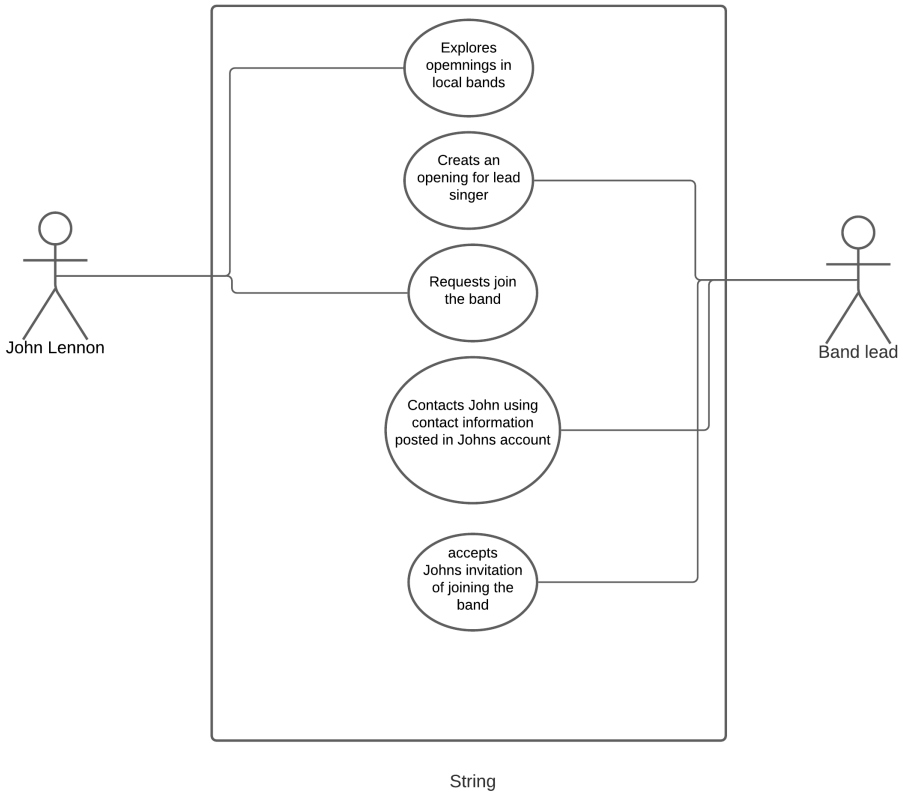


Figure 1: A musician looking for a band to join

Title	A band looking for musicians
Actor	Riot (a local band)
Action	Riot is one of the local bands in the city. Recently this band has been getting a lot of suggestions about adding an acoustic guitarist to their group. They are confused on how they should post this opening for free. The band lead discovers String. He makes an account and adds the band in his account and asks his other band members to join using their String accounts. Then they create an availability which is visible to other members on the platform and can request to join. The band reviews the applicants and contacts them with the contact information provided and if they see the applicant fit, then they add them to their band. They also see the feature of adding videos to their profile and so they add their previous recordings to the platform.

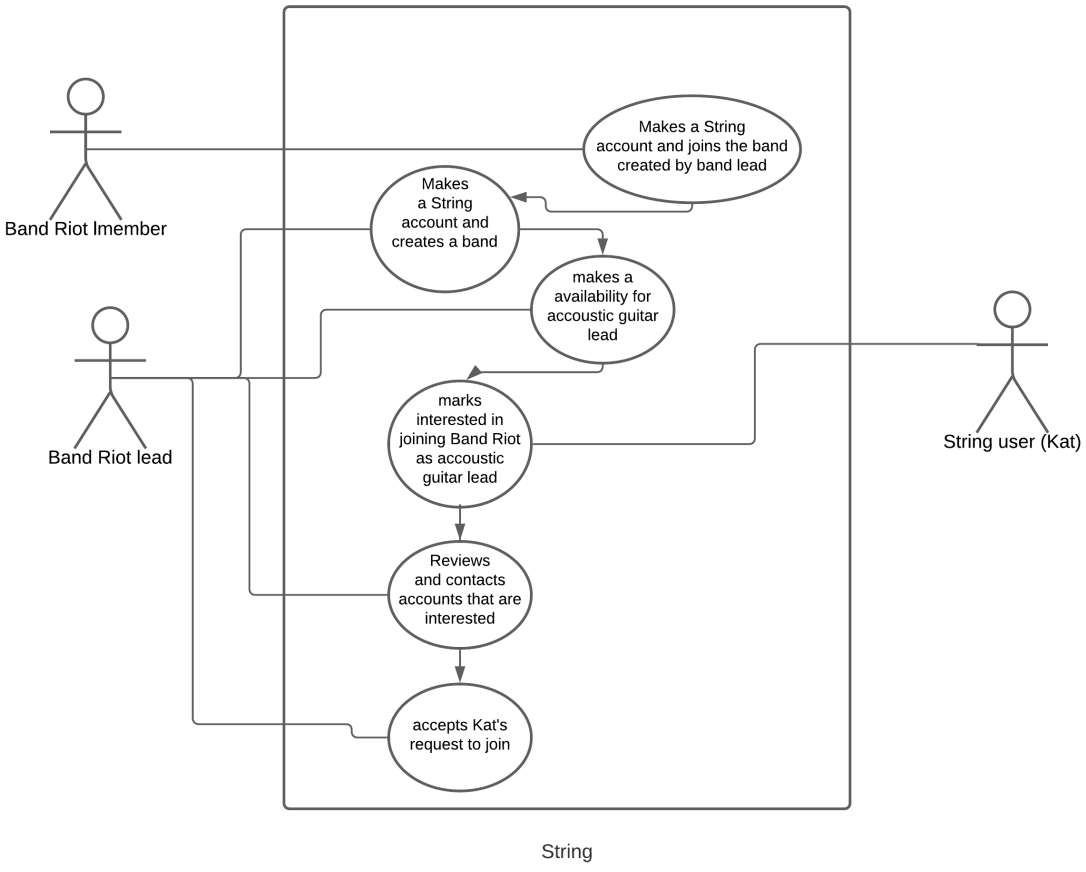


Figure 2: A band looking for musicians

Title	A band manager looking to have one place to manage/display all bands he manages
Actor	Aron
Action	Aron is a band manager who manages two local bands. He is doing this as a passion and not as his occupation. He wants a place where he can easily manage these bands and upload their materials for publicity. Aron comes to String and registers as a user. On the website he creates two band pages and asks all the players to register as users on String and join his request. He then uploads all the previous band performance videos on the platform and uses the band profile link to share via social media.

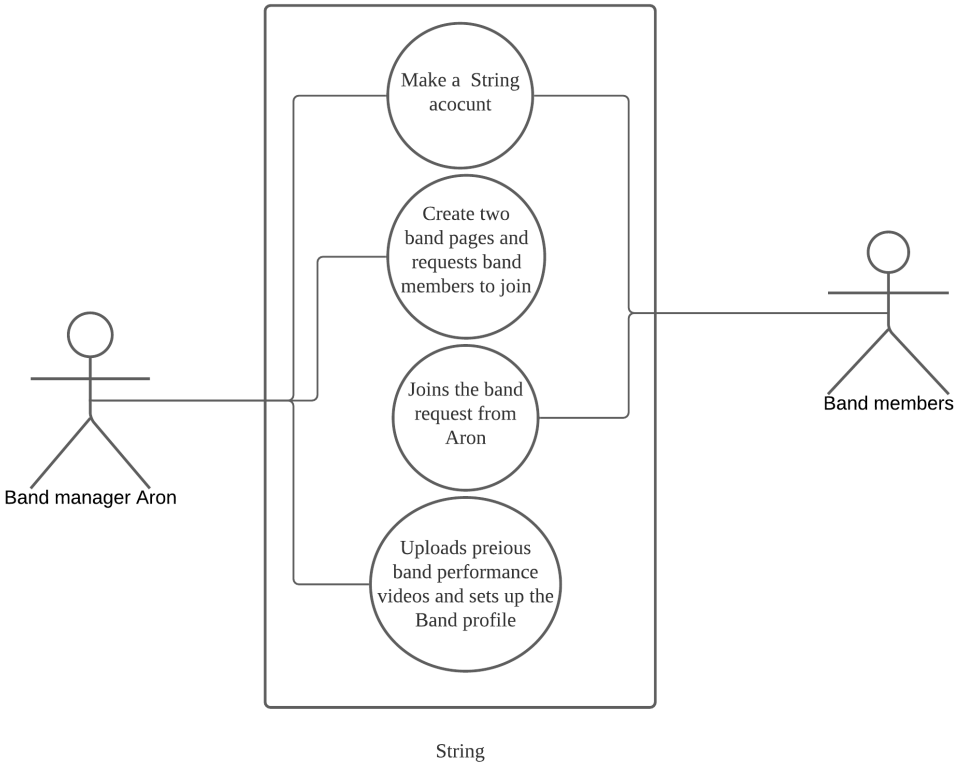


Figure 3: One place to manage and display

Title	Looking to discover more music in their area (as a listener)
Actor	Alan
Action	Alan is a local to the area, and enjoys discovering and listening to live music. However, he does not always enjoy staying out late at bars where local music performances usually happen. He searches online for local bands and visits String platform. He is able to see a list of local bands and their previous performances. He enjoys looking for local bands that have open rehearsals during the day (as indicated on their band profile description), showing up, meeting local musicians, listening to new songs, and helping to give the band feedback on new material.

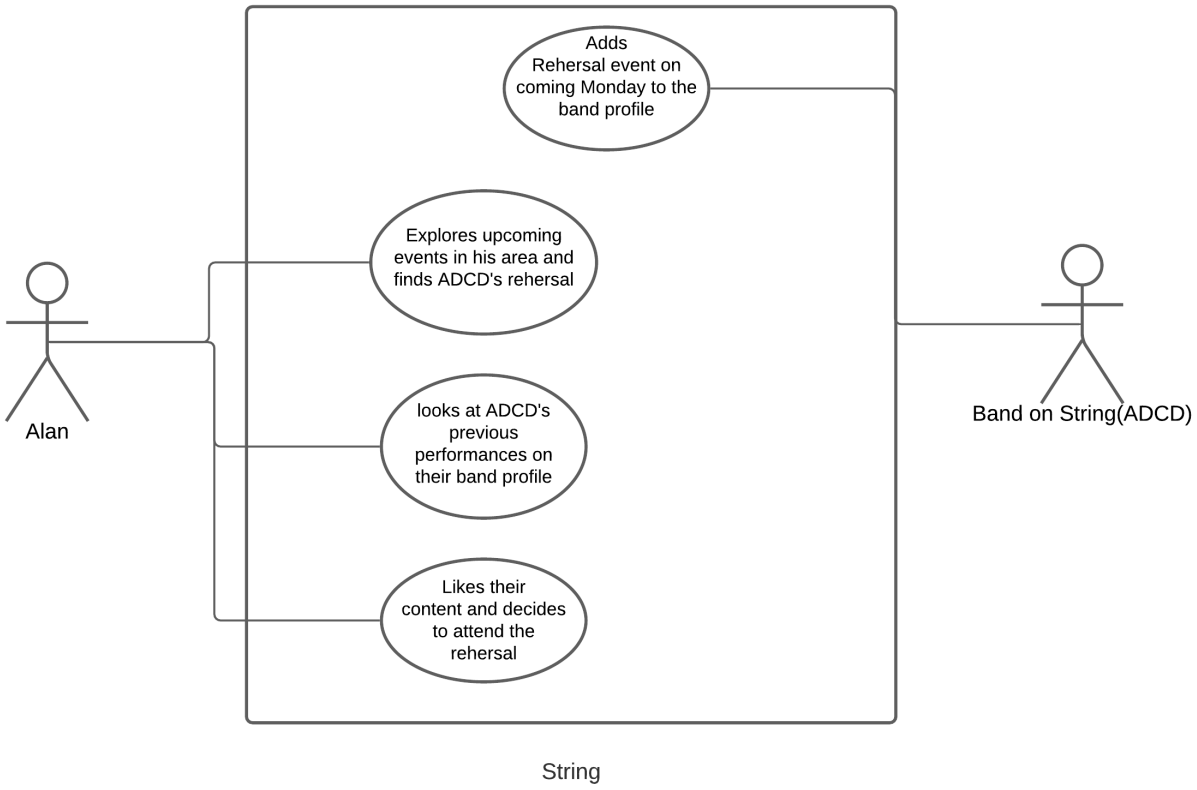


Figure 4: Looking to discover more music in their area

Title	Looking to sell products to local bands (looking for contact info)
Actor	RockyRoad
Action	A small store named RockyRoad is getting shipments of musical instruments and they are struggling with selling because of COVID-19, so they decide to expand into personal outreaching. They want to contact bands in their local area that would be interested in buying instruments from them. The manager looks at the bands in the local area via String and collects their contact information from the band profile page and contacts them about the store's offers. He is able to find some new customers looking to upgrade their instruments and he is able to sell his products to them.

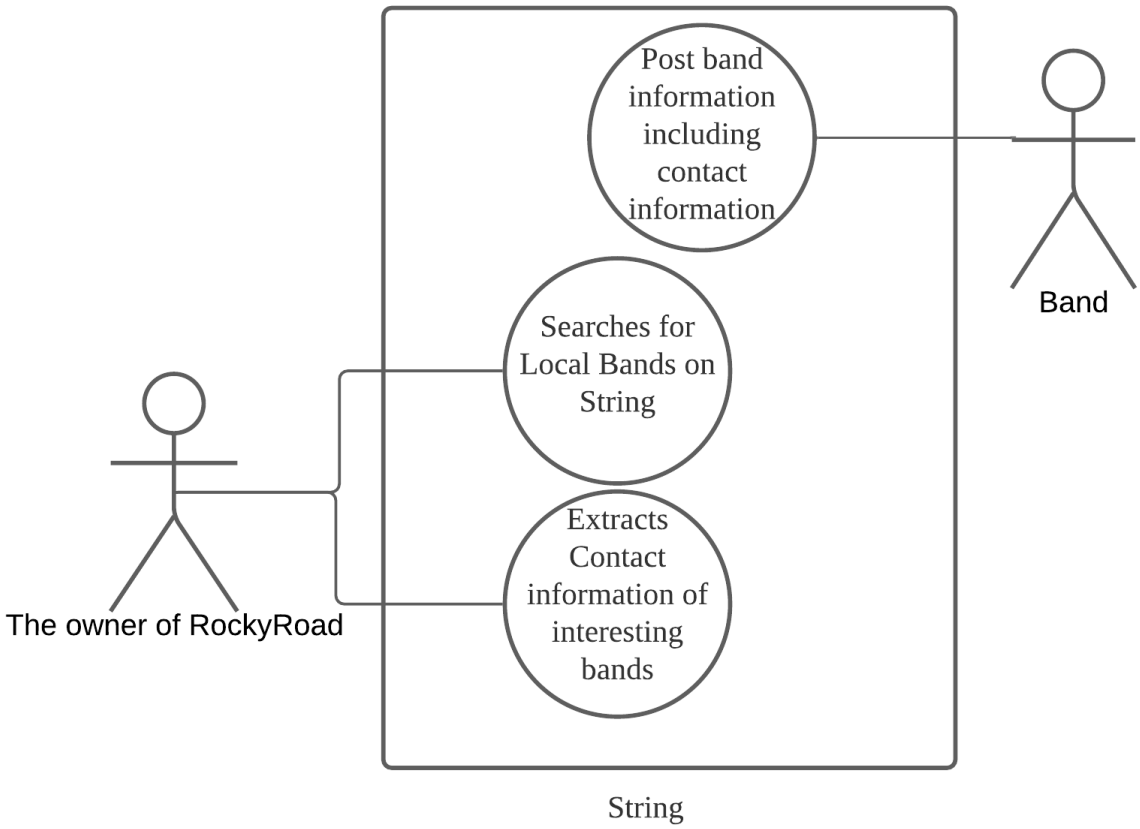


Figure 5: Looking to sell products to local bands

Title	Looking for bands to play
Actor	Angello
Action	Angello has a commercial property and he wants to rent it out . He also has a little dinner restaurant "At Angello's" that he wants to have live Jazz music at. He looks through local groups in the area. After careful consideration he gets in contact with 10 bands through contact information that he found on String. Lucky for him, local band Riot was looking for a place where they could store their equipment and practice so they rented Angello's commercial property. Later that day 3 Jazz bands got back to Angello and agreed to play in his restaurant. Angello schedules each band for Friday, Saturday and Sunday - all different bands and musicians. He is very happy with the result.

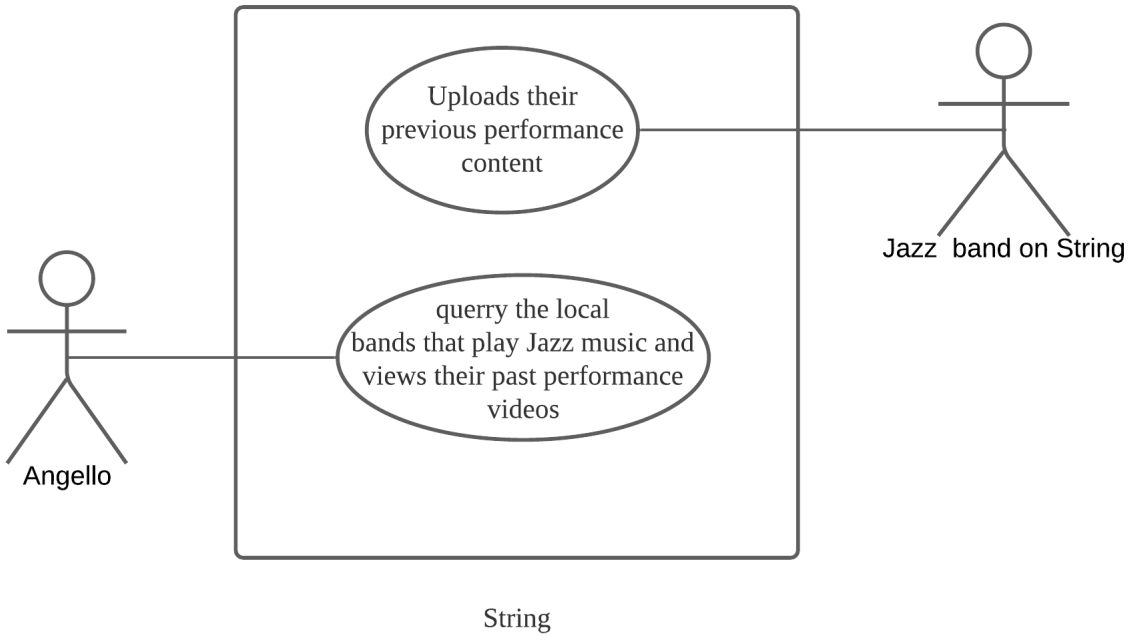


Figure 6: Looking for bands to play

3 List of main data items and entities

1. String Account: Account of a Registered user on the platform.
2. Administrator String Account: Account of an Administrator on the platform.
3. Registered user: A person who has a String account and is logged into that account.
4. Guest: A person who does not have a String account but is just viewing pages on our platform.
5. Administrator: A person who has an Administrator String account and is logged into that account. They have access over all information, items and entities on the platform. They are the ones who will manage inappropriate content.
6. Band: A band can be created by a registered user. It is an entity that other registered users can join if Band Admin permits it.
7. Band Admin: The person who created the band using their registered account, or anyone promoted to that role by an existing Band Admin
8. Band Member: A registered user who joins a Band, with a Band Admin's permission.
9. Band post: Image or video uploaded by bands
10. Invitation: Sent by a registered user to join a band in response to availability post
11. Event Entry: A band can create an entity known as an event in order to provide information on an upcoming rehearsal, or performance - with sections for information and Set List.
12. Event List: A registered user who is a part of many bands can create an event list, which will list in one place all upcoming events for all the bands that they belong to.
13. Set Entry: A song which the band is planning to play for a specific event/performance. Includes details such as run time, tempo, and order in performance.
14. Set List: A list that a band can create, tied to an event, detailing the song list (and possibly order) that the group will be performing for said event.
15. Repertoire Entry: A song which the band can play, including details such as run time, genre, links to past video, etc.
16. Repertoire List: A list that a band can create which holds the names, notes on, etc. of the current songs the band is rehearsing and can play.

4 Functional Requirements

4.1 Guest

1. Guest shall be able to view all Bands.
2. Guest shall be able to view all Band Members.
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 view all Bands' Set List.
6. Guest shall be able to view all Bands' Repertoire List.
7. Guest shall be able to access contact information of all Bands.
8. Guest shall be able to search any Band.
9. Guest shall be able to filter all Bands based on their geographic location.
10. Guest shall be able to filter all Bands based on the genre of music the Bands have selected.
11. Guest shall be able to register an account with a valid email address.
12. Guest shall be able to contact an Administrator to get help at any time.

4.2 Registered User

1. Registered User should be a Guest.
2. Registered User should have a String Account.
3. Registered User shall be able to view their String Account.
4. Registered User shall be able to log in using their email and password.
5. Registered User shall be able to change their email in their String Account.
6. Registered User shall be able to change their password in their String Account.
7. Registered User shall be able to delete 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 if they are a musician in their String Account.
10. Registered User shall be able to edit the field to specify which instrument(s) they play in their String Account.
11. Registered User shall be able to edit their current geographical location in their String Account.
12. Registered User shall be able to edit their contact information in their String Account.
13. Registered User be able to send an invitation to join a Band.
14. Registered User be able to create Band(s).
15. Registered User be able to view all the Bands they are a Band Member of.

4.3 Administrator

1. Administrator shall be able to log in using their email and password.
2. Administrator shall be able to change their email in their Administrator String Account.
3. Administrator shall be able to change their password in their Administrator String Account.
4. Administrator shall be able to create a Band.
5. Administrator shall be able to delete any Band.
6. Administrator shall be able to delete any Registered User.
7. Administrator shall be able to add registered users to any Band.
8. Administrator shall be able to delete any content posted by any Band.
9. Administrator shall be able to delete any Event Entry in any Band.
10. Administrator shall be able to delete any Set List in any Event Entry in any Band.
11. Administrator shall be able to delete any Repertoire Entry in any Band.
12. Administrator shall be able to remove any Band Member in any Band.
13. Administrator shall be able to change Band Admin in any Band.

4.4 Band

1. Band can only be created by a Registered User.
2. Band shall have only one Band Admin.
3. Band shall have at least one Band Member.
4. The Registered User who creates the Band is set as the Band Admin.
5. Band shall have a field indicating if they are accepting invitations from Registered Users.
6. Band shall have a minimum of one Band Member.
7. Band which have no Band Members shall be deleted.
8. Band shall have a geographic location.
9. Band shall have Event List indicating upcoming events.
10. Band shall have Band Post(s).
11. Band shall have contact information.
12. Band shall have a Repertoire List.

4.5 Band Member

1. Band Member should be registered users.
2. Band Member shall be able to add Band posts to their Band.
3. Band Member shall be able to delete Band posts to their Band.
4. Band Member shall be able to leave their Band at any time.
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 edit an Event Entry in the Event List of their Band.
8. Band Member shall be able to add a Set List to an Event Entry of their Band.
9. Band Member shall be able to delete a Set List to an Event Entry of their Band.
10. Band Member shall be able to edit a Set List to an Event Entry of their Band.

11. Band Member shall be able to add a Repertoire Entry in the Repertoire List of their Band.
12. Band Member shall be able to delete a Repertoire Entry in the Repertoire List of their Band.
13. Band Member shall be able to edit a Repertoire Entry in the Repertoire List of their Band.

4.6 Band Admin

1. Band Admin should be a Band member.
2. Band Admin shall remove a Band member of their Band.
3. Band Admin shall be able to assign other Band Member as Band Admin.
4. Band Admin shall be able to view all Invitations in their Band.
5. Band Admin shall be able to accept an Invitation in their Band.
6. Band Admin shall be able to delete an Invitation in their Band.
7. Band Admin shall be able to delete their Band.
8. Band Admin shall be able to change the name of their Band.

4.7 Band Post

1. Band Post can contain an image.
2. Band Post can contain a video.
3. Band Post must contain a description.
4. Band Post must contain a title.

4.8 Invitation

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

4.9 Event Entry

1. Event Entry must have a title field.
2. Event Entry must have a location field indicating where the event will take place.
3. Event Entry must have a date field indicating when the event will take place.
4. Event Entry must have a field indicating which Band created it.
5. Event Entry shall have a description field.
6. Event Entry shall have a Set List.

4.10 Repertoire Entry

1. Repertoire Entry must have a field indicating song's name.
2. Repertoire Entry shall have a field indicating song's tempo.
3. Repertoire Entry shall have a field indicating song's mood.
4. Repertoire Entry shall have a field indicating song's run time.
5. Repertoire Entry shall have a list of links.
6. Repertoire Entry shall have a field indicating ift the song is in the rehearsal stages.

4.11 Set Entry

1. Set Entry shall have a field indicating song's name.
2. Set Entry shall have a field indicating song's mood.
3. Set Entry shall have a field indicating song's tempo.
4. Set Entry shall have a field indicating song's approximate run time.

4.12 Event List

1. Event List shall contain a list of Event Entries.

4.13 Repertoire List

1. Repertoire List shall contain a list of Repertoire Entries.

4.14 Set List

1. Set List shall contain a list of Set Entries.

2. Set List must contain at least one Set Entry

3. Set List shall have Set Entries in projected performance order.

5 Non-functional Requirements

- 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.
    3. Passwords set by users should be at most 21 characters long.
    4. Passwords should contain at least one uppercase letter.
    5. Passwords should contain at least one number.
    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.
    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 platform.
    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.
    2. All api shall be well documented in detail with implementation details.
- 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.
- 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.
6. Images with .jpg, .jpeg or .png extensions only are accepted

6 Competitive Analysis

Company Feature	AMY	Bandcamp	BandLab	Drooble	Reverbnation	String
Multiple group/event management	-	-	-	-	-	+
Repertoire list	-	-	-	-	-	+
Event public/private details	+	-	-	-	-	++

Side-by-side Comparison

Company	Strengths	Weaknesses
Amy	• Can find or list related services • Multiple profile types available • Can organize jam sessions	• Tiny social media reach • Complicated interface • Does not come up on search engines consistently for related searches • Not prominent (no discussion on internet) • Limits videos to 20 seconds(!)
Bandcamp	• Easy to contact and get together w/ other bands • Bands actually get paid • Vinyl Purchases and direct digital downloads	• Cannot download songs directly to mobile app • Most users are artists, not fans
BandLab	• Great music creation tools • Strong community of DIY members	• Prominent latency issues
Drooble	• Incentivizes participation by offering money for reviews of music, enabling feedback for other users • Rewards social currency gathering by letting musicians purchase promotion tool access • Provides a goal for participants by running a platform specific popular music chart	• Payment rises with services used (no clear cost) • No networking except for online only • Users mostly in Eastern Europe
Reverbnation	• The most features, including music hubs, vlogs, collabs, and promotions • Large fan reach and sales potential • Easy distribution of music to major digital platforms	• Contests are just a time sink for artists

Table 1: Competitor Strengths and Weaknesses

Company	Pricing	Social Media	Onboarding Experience
Amy	• Free	• Facebook, Twitter, Instagram, Youtube	Facebook/Google Integration
Bandcamp	• Free (revenue share on sales, 10-15%), optional paid tier (10 - 50 USD/mo.)	• Facebook, Twitter, Instagram	• Allows for own domain
BandLab	• Free	• Facebook, Twitter, Instagram	• A bit involved (user categories for role, activity, and music genres)
Drooble	• Variable according to services purchased	• Facebook, Instagram	• Email or Google sign in
Reverbnation	• Free, or 13 USD/mo., or 20 USD/mo.	• Facebook, Instagram, Twitter, Youtube	• Intro tour, Upload prev. work, complete user profile

Table 2: Competitor Pricing, Social Media, and Onboarding Experience Comparisons

None of the other competitors that we examined had any substantial focus on group membership management or in person musical collaboration; therefore, our feature list is rich in management tools toward that end, in order to fully take advantage of that underserved population. The Repertoire and Set Lists allow for easy management tools for musical groups to prepare for events, and Event Lists allow for easy performance, rehearsal, and casual 'jam session' management. Users can be a part of many Bands at once on our platform, which means that String can be a one-stop hub for musicians to find, join, view, manage, and ultimately make music with differing groups of people in their area. All these features make String the best platform for musicians in a specific local geography who want to (or already do) play in groups together, either for fun, or for paid gigs.

7 High-level System Architecture and Technologies Used

- Server Host: AWS EC2 t2.micro 1 vCPU 1GB RAM
- Operating System: Ubuntu 16.04 Server
- Database: MySQL
- Web Server: Caddy v1
- Server-Side Language: JS (Node.js)
- Additional Technologies: Web Framework: Express.js, React.js, Redux
- IDE: VS Code
- Web Analytics: Google Analytics
- SSL Cert: Lets Encrypt

8 Team

- Team Lead and Github Master: Jainam Shah
  - Worked on setting up team meetings and making the agenda for the meeting.
  - Maintained communication with team members to execute appointed tasks.
  - Worked to create the content of the M1 document
    - Formed the non functional requirements.
    - Helped create use case diagrams.
    - Helped introduce data items and entities
    - Helped developing the use cases
- Frontend Lead: Alfredo Diaz
  - Attended all team meetings on time and actively responded on slack.
  - Worked on setting up Trello dashboard and getting team members added to the board.
  - Worked to create the content of the M1 document
    - Researched the competitors and took responsibility for the competitor analysis section.
    - Helped create use case diagrams.
    - Worked on the executive summary to make it more compelling
    - Helped developing the use cases
- Frontend Developer: Leonid Novoselov
  - Attended all team meetings on time and actively responded on slack.
  - Worked to create the content of the M1 document
    - Helped create the functional requirements.
    - Worked on the executive summary to make it more compelling
    - Helped developing the use cases
- Frontend Developer: Eric Chen
  - Attended all team meetings and actively responded on slack.
  - Worked to create the content of the M1 document.
    - Helped create use case diagrams.

        * Helped developing the use cases

- Backend Lead and Documentation Master: Warren Singh
    - Attended all team meetings on time and actively responded on slack.
    - Worked on compiling and updating the M1 document using LaTeX.
    - Worked to create the content of the M1 document
        * Helped create the functional requirements.
        * Helped with the competitive analysis section
        * Helped introduce new data items and entities
        * Helped developing the use cases
- Backend Developer and Database Architect: Ritesh Panta
    - Attended all team meetings on time and actively responded on slack.
    - Worked to create the content of the M1 document
        * Helped create the functional requirements.
        * Helped introduce new data items and entities
        * Helped developing the use cases

9 Checklist

- Team found a time slot to meet outside of the class: **DONE**
- Github master chosen: **DONE**
- Team decided and agreed together on using the listed SW tools and deployment server: **DONE**
- Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing: **DONE**
- Team lead ensured that all team members read the final M1 and agree/ understand it before submission: **DONE**
- Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.): **DONE**

2.2 Milestone 2

SW Engineering CSC 648/848 - FALL 2020

Milestone 2

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 2 v2.0	November 17, 2020
Milestone 2 v1.0	October 29, 2020
Milestone 1 v2.0	October 14, 2020
Milestone 1 v1.0	October 1, 2020

December 4, 2020

Contents

1 Data Definitions V2

1. String Account: Account of a Registered user on the platform.
   (a) Email - an email address of the Registered User
   (b) Password - used to log into the String Account.
   (c) Name - the name of the Registered User.
   (d) Phone number - (optional) phone number of the Registered User.
   (e) Location - (optional) the address of the Registered User.
   (f) Location Coordinates - (optional) the address coordinates of the Registered User.
   (g) Role - (optional) list of roles the Registered User can perform.
   (h) Genre(s) - (optional) list of primary musical interests (i.e. classical, jazz, rock, electronic)
   (i) Profile picture - (optional) Registered User's picture.
2. Administrator String Account: Account of an Administrator on the platform (i.e. the team building and managing the String Platform).
   (a) Email: an email address of the Administrator.
   (b) Password: used to log into the Administrator String Account.
   (c) Name: the name of the Administrator.
   (d) Role: role of the Administrator in the team (i.e frontend lead, team lead, content manager, etc.).
3. Registered User: A person who has a String account and is logged into that account
4. Guest: A person who does not have a String account and is just viewing pages on our platform
5. Administrator: A person who has an Administrator String account and is logged into that account. They have access over all information, items and entities on the platform. They are the ones who will manage inappropriate content.
6. Band Member: A registered user who joins a Band, with a Band Admin's permission.
   (a) Band Admin: if the Band member has admin access in the band. (a boolean value)
   (b) Role: Role that the Band Member has in the Band.
   (c) Date joined: When the Band Member joined the Band.

7. Band: A band can be created by a registered user. It is an entity that other registered users can join if Band Admin permits.
    (a) Band Name: (required) Name of the band. (unique)
    (b) Band logo image: Logo image of the Band.
    (c) Band Description: A short description of what the band is about.
    (d) Band Links: list of links.
        i. Link: the link pointing to another webpage.
        ii. Short description: a small description of the link.
    (e) Band Post: Image or video uploaded by bands.
        i. Media: Link pointing to an image or a video (could be on youtube or Vimeo or Google Drive).
            A. Type of media: either image or video.
            B. Link: link to that image or video.
        ii. Title (required): title of what the Band Post is about.
        iii. Description (optional): description of the Band Post
    (f) Location (required): Address that the band belongs to and where they are performing.
    (g) Location Coordinates (required): latitude and longitude of the address of the Band
    (h) Event Entry: A band can create an entity known as an event in order to provide information on an upcoming rehearsal, or performance - with sections for information and Set List.
        i. Event title: (required) The title of the event.
        ii. Date of event: (required) month day and year the event is taking place.
        iii. Start Time of event: (required) time at which the event will begin.
        iv. End time of event: (required) time at which event will end.
        v. Location of event (required): Address and description of where the event will be held.
        vi. Coordinates: (required) latitude and longitude of the address of the Event
        vii. Event description: description of the event.
        viii. Set Entry: A song which the band is planning to play for a specific event/performance. Includes details such as run time, tempo, and order in performance.
            A. Song: name of the song.
            B. Run time: length of the song.

(i) Repertoire Entry: A song which the band can play, including details such as:

i. Song - name of the song.

ii. Run time - length of the song.

iii. Genre - genre of the song.

iv. Link - Link to a site with more description/media about the performance of this song.

8. Invitation: Sent by a registered user to join a band in response to availability post

(a) Message: A short paragraph of why the user would want to join the Band

(b) Date: Date on which the Invitation was sent.

(c) Sender account: the String Account who sent the invitation

(d) Receiver band: the Band that the invitation was sent to.

2 Functional Requirements V2

2.1 Priority 1

2.1.1 Guest

1. Guest shall be able to view all Bands.
2. Guest shall be able to view all Band Members.
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 access contact information of all Bands.
6. Guest shall be able to search for any Band by name.
7. Guest shall be able to filter any Bands by genre.
8. Guest shall be able to filter Bands by location.
9. Guest shall be able to register an account with a valid email address.
10. Guest shall be able to view all Events' Set List.
11. Guest shall be able to view all Bands' Repertoire List.

2.1.2 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 view all the Bands information they are a Band Member of.
8. Registered User shall be able to change their password in their String Account.

9. Registered User shall be able to edit the field indicating their name in their String Account.
10. Registered User shall be able to edit the field indicating their role.
11. Registered User shall be able to edit their current geographical location in their String Account.
12. Registered User shall be able to edit their contact information in their String Account.

2.1.3 Band

1. Band can only 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 contact information.
8. Band shall have an Event List indicating upcoming events.
9. Band shall have a Repertoire List.

2.1.4 Band Member

1. Band Member should 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 Set Entry to an Event Entry of their Band.
8. Band Member shall be able to delete a Set Entry to an Event Entry of their Band.
9. Band Member shall be able to add a Repertoire Entry to their Band.
10. Band Member shall be able to delete a Repertoire Entry to their Band.

2.1.5 Band Admin

1. Band Admin should be a Band member.
2. Band Admin shall remove a Band member of their Band.
3. Band Admin shall be able to view all Invitations in their Band.
4. Band Admin shall be able to accept an Invitation in their Band.
5. Band Admin shall be able to delete an Invitation in their Band.

2.1.6 Band Post

1. Band Post shall contain an image.
2. Band Post shall contain a video.
3. Band Post shall contain a description.
4. Band Post shall contain a title

2.1.7 Invitation

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 was it sent to.
5. Accepting an Invitation shall result in adding the registered User who sent it to the Band as a Band member.

2.1.8 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 when the event will take place.
4. Event Entry shall have a time field indicating when the event will take place.
5. Event Entry shall have a field indicating which Band created it.

6. Event Entry shall have a description field.

7. Event Entry shall have a list of Set Entries.

2.1.9 Repertoire Entry

1. Repertoire Entry shall have a field indicating song's name.

2. Repertoire Entry shall have a field indicating song's genre.

3. Repertoire Entry shall have a field indicating song's run time.

4. Repertoire Entry shall have a link.

2.1.10 Set Entry

1. Set Entry shall have a field indicating song's name.

2. Set Entry shall have a field indicating song's run time.

3. Set Entry shall have a field indicating play order.

2.2 Priority 2

2.2.1 Guest

1. Guest shall be able to add review to a Band.

2. Guest shall be able to play songs attached with the Repertoire Entry.

3. Guest shall be able to search Band Post. location.

2.2.2 Registered User

1. Registered Users shall be able to change their email in their String Account.

2. Registered Users shall be able to like a band post.

3. Registered Users shall be able to add a comment on a band post.

4. Registered Users shall be able to RSVP to an event.

2.2.3 Administrator

1. Administrator shall be able to change their email in their Administrator String Account.
2. Administrator shall be able to change their password in their Administrator String Account.
3. Administrator shall be able to delete any Set List in any Event Entry in any Band.
4. Administrator shall be able to delete any Repertoire Entry in any Band.
5. Administrator shall be able to log in using their email and password.
6. Administrator shall be able to delete any Band.
7. Administrator shall be able to delete any content posted by any Band.
8. Administrator shall be able to delete any Event Entry in any Band

2.2.4 Band

1. Band shall have a Repertoire List

2.2.5 Band Member

1. Band Member shall be able to edit an Event Entry in the Event List of their Band.
2. Band Member shall be able to edit a Set List to an Event Entry of their Band.
3. Band Member shall be able to edit a Repertoire Entry in the Repertoire List of their Band.

2.2.6 Band Admin

1. Band Admin shall be able to assign other Band Member as Band Admin.
2. Band Admin shall be able to change the name of their Band.

2.2.7 Event Entry

1. Event Entry shall have an image
2. Event Enrty shall have an RSVP list

2.2.8 Repertoire Entry

1. Repertoire Entry shall have a playable song link.

2.2.9 Set Entry

1. Set List shall contain a list of Set Entries.
2. Set List must contain at least one Set Entry
3. Set List shall have Set Entries in projected performance order.

2.2.10 Repertoire List

1. Repertoire List shall contain a list of Repertoire Entries

2.2.11 Band Post

1. Band Post shall have a count of likes.
2. Band Posts shall have a comment section for participation from Registered Users.

2.3 Priority 3

2.3.1 Guest

1. Guest shall be able to contact an Administrator to get help at any time.

2.3.2 Registered User

1. Registered User shall be able to delete their String Account.
2. Registered User shall be able to add comments to an Event.

2.3.3 Administrator

1. Administrator shall be able to create a Band.
2. Administrator shall be able to delete any Registered User.
3. Administrator shall be able to add registered users.
4. Administrator shall be able to add registered users to any Band.
5. Administrator shall be able to remove any Band Member in any Band.
6. Administrator shall be able to change Band Admin in any Band.

2.3.4 Band

1. Band shall have at least one Band Member.
2. Band which have no Band Members shall be deleted.
3. Band shall have one or more Band Admin(s).

2.3.5 Band Member

1. Band Members shall have access to a text chat viewable only by Band Members.
2. Band Members can invite Registered Users to an Event.

2.3.6 Band Admin

1. Band Admin shall be able to delete their Band.
2. Band Admin shall be able to make other Band Members to be an additional Band Admin.

2.3.7 Repertoire Entry

1. Repertoire Entry shall have a list of links.
2. Repertoire Entry shall have a field indicating if the song is in the rehearsal stages.
3. Repertoire Entry shall have notes.

2.3.8 Event Entry

1. Event Entry shall have a list of registered users invited to the event.
2. Event Entry shall have a section for Registered Users to comment and discuss.

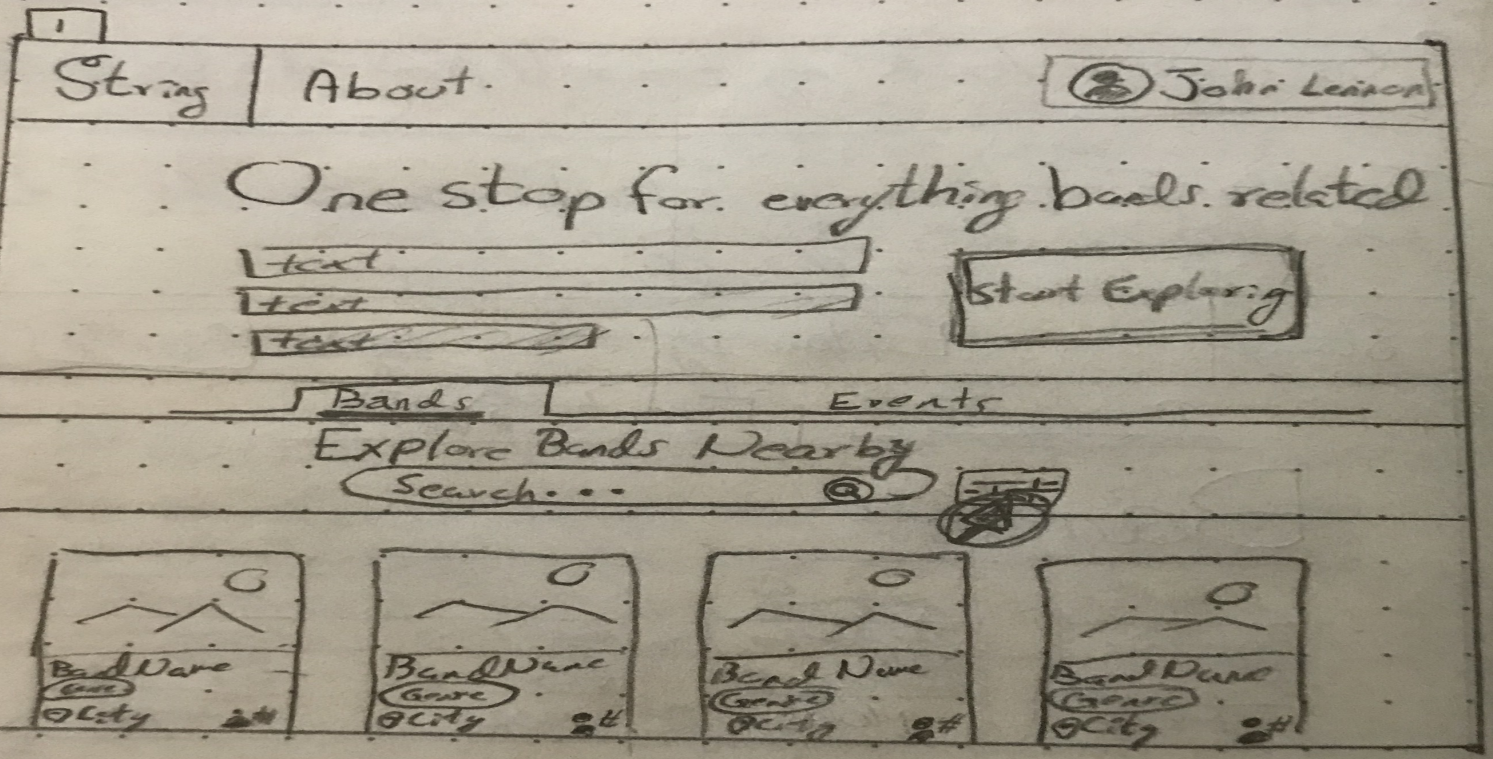
3 UI Mockups and Storyboards

3.1 Use Case 1: A solo artist looking for a band to join

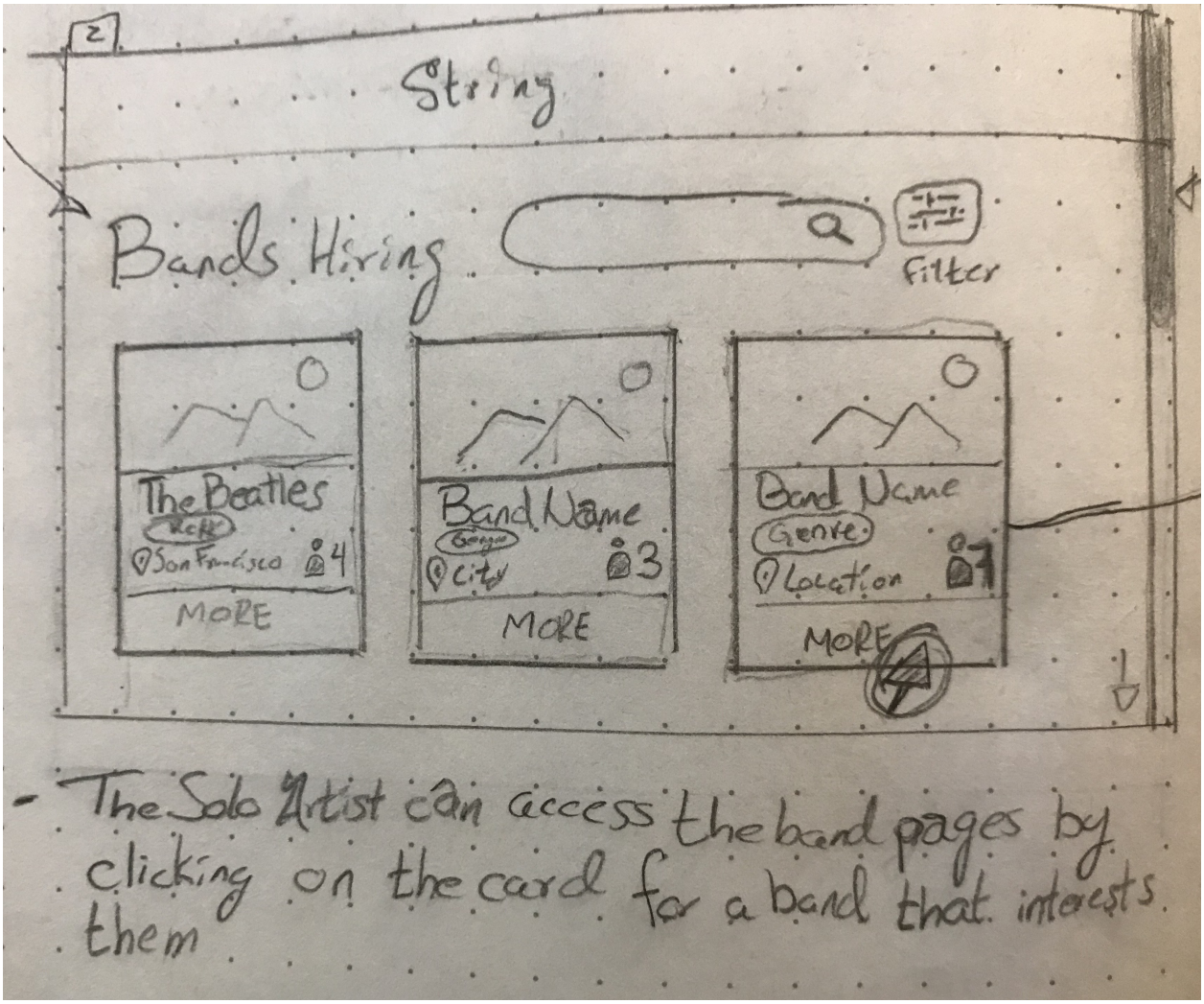
1. A registered user as solo artist goes to the explore page and taps the filter button
2. In the filter modal user selects to browse Bands by “Bands that are hiring”.
3. Scroll down to find more
4. The user gets a list of all the bands in their city that are looking for new members
5. Solo artists can access the band pages
6. Inside the band pages registered users can click the “apply” apply button to send a notice of interest on the role to that band
7. Alternatively, they can reach out directly to the bands that have their contact information set as public

Use case (1)

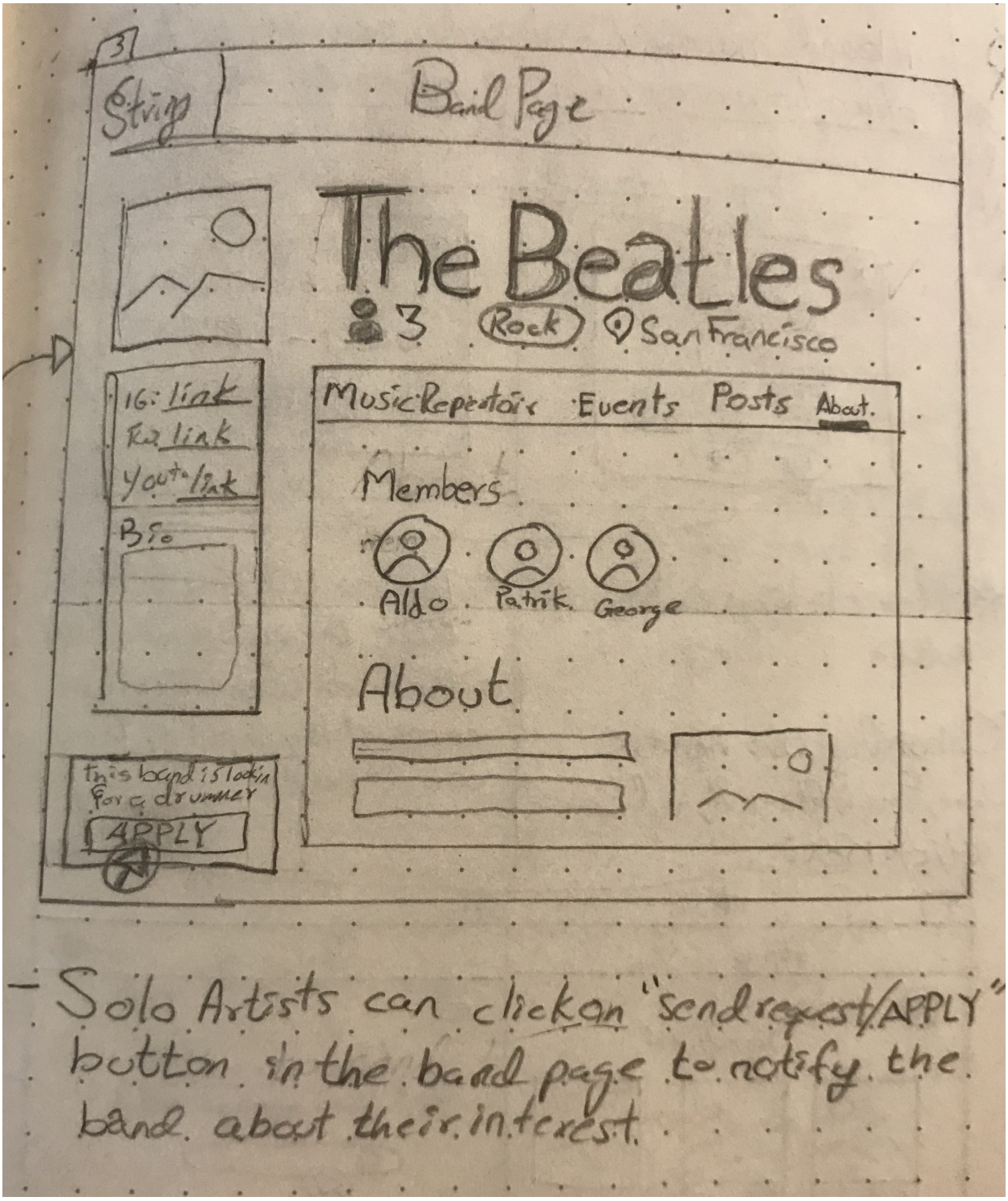
A solo artist looking for a band to join



- A registered user as a solo artist goes to the explore page and taps the filter button
- In the filter modal user selects to browse bands by "Bands that are hiring"



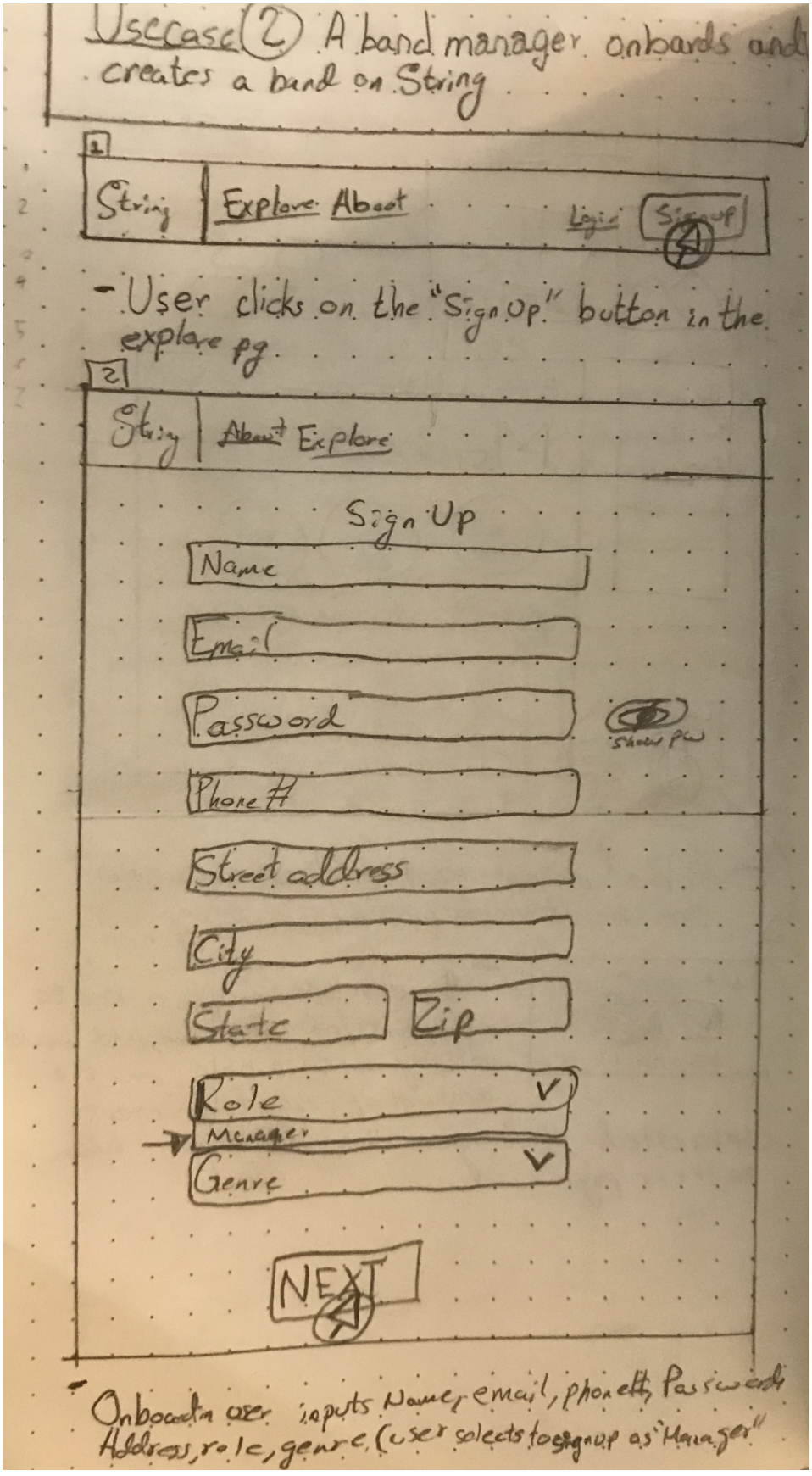
- The Solo Artist can access the band pages by clicking on the card for a band that interests them

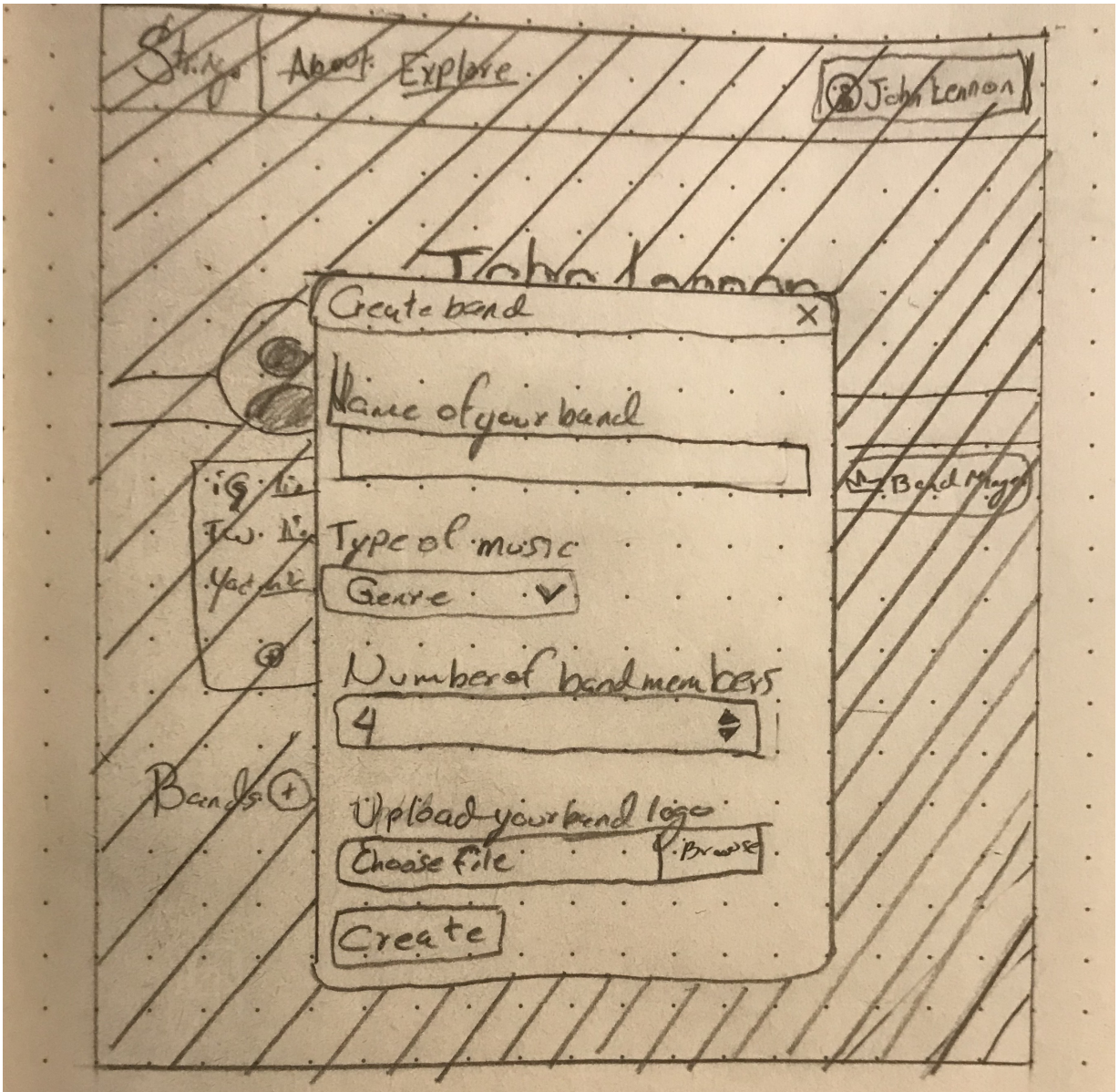


- Solo Artists can click on "send request/APPLY" button in the band page to notify the band about their interest

3.2 Use Case 2: A band manager onboards, searching to have one place manage/display all bands he manages

1. User clicks on the sign up button in the explore page or any other public page.
2. Onboarding user inputs: name, last name, email, phone number, and password, address, role, instrument and genre
3. User selects to sign ups as “Band Manager” and is prompted to create a band after signup (can also just click the “+” button under the bands section)
4. The user may complete the details for the band page or finish later (Band name,logo/image, street address, genre, description, links, events, posts, music repertoire)


Usecase (2) A band manager onboards and creates a band on String
1
String
Explore
About
Login
Sign up
- User clicks on the "Sign Up" button in the explore pg.
2
String
About
Explore
Sign Up
Name
Email
Password
Show PW
Phone #
Street address
City
State
Zip
Role
Manager
Genre
NEXT
- Onboarding user inputs Name, email, phone#, Password, Address, role, genre. (user selects to sign up as "Manager"



- User is prompted to create a band after sign up (can also click the "+" button under the Bands section)

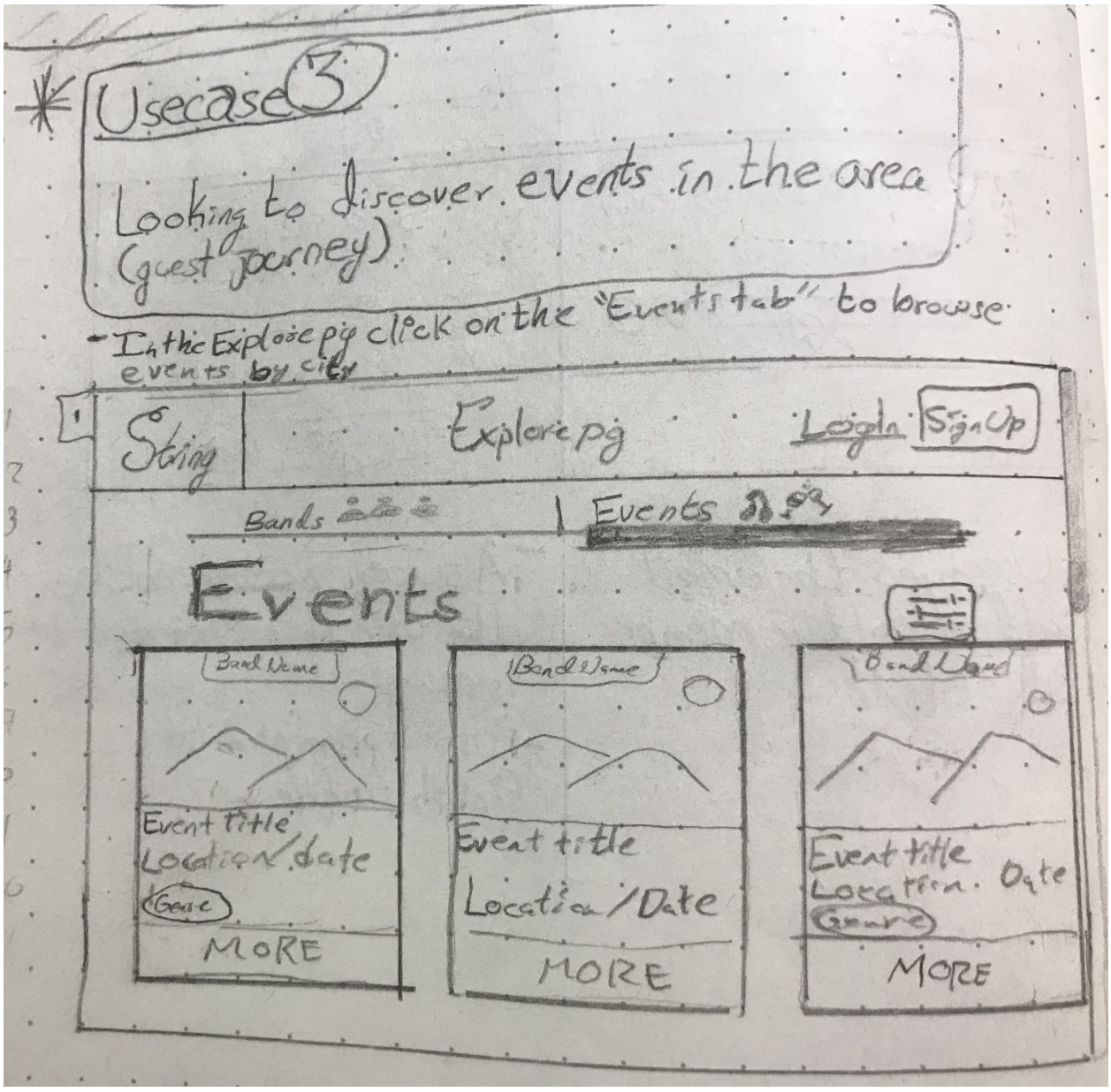
- The user may complete the details for the band page or finish later (Band name, logo/image, street address, genre, description, links, events, posts, music repertoire).

3.3 Use Case 3: Looking to discover music events in their area (guest journey)

1. In the explore page tap the Events button to browse events by city.

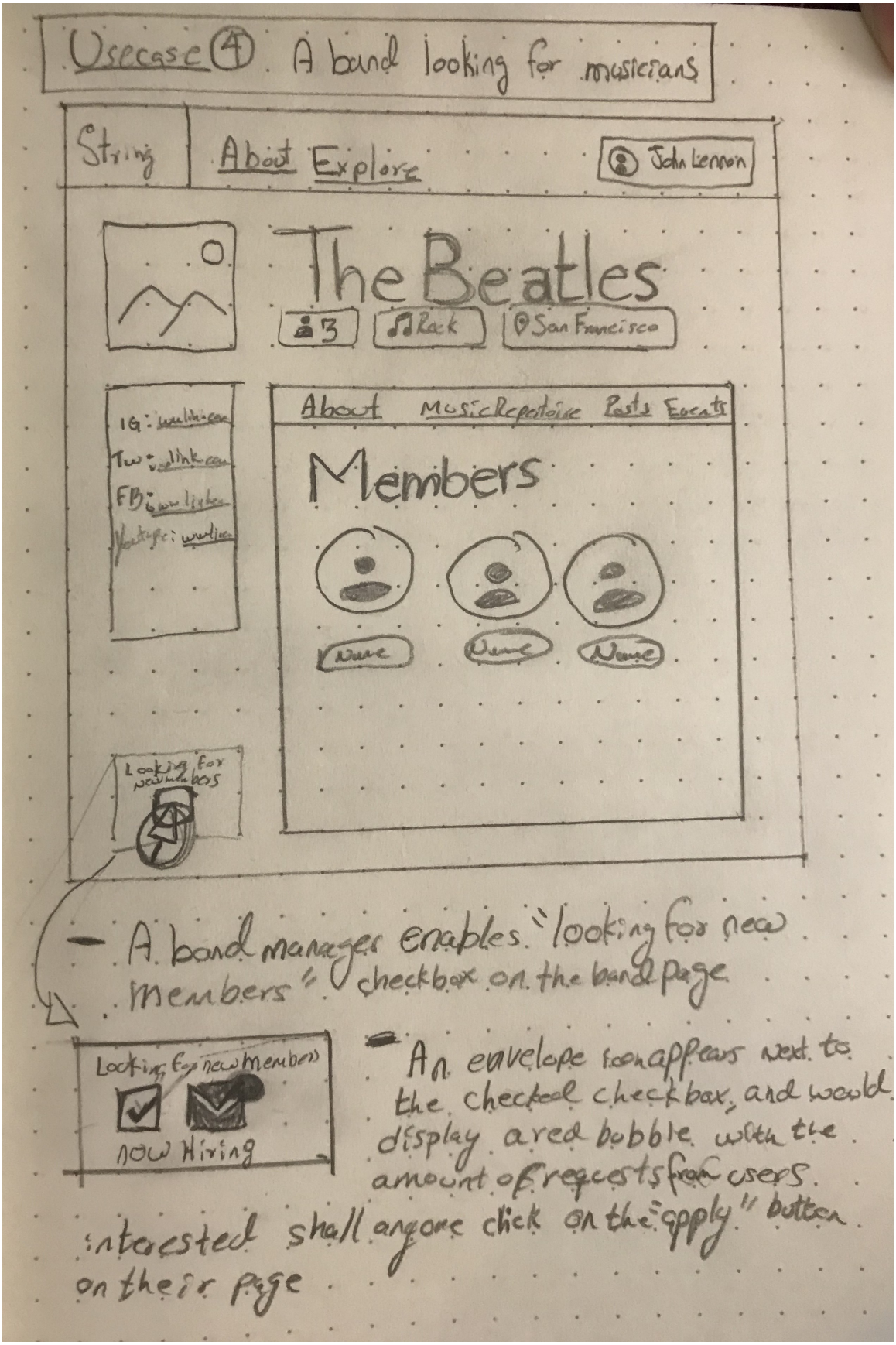
2. Scroll down to find more.

3. For filtering results of any kind the filter button located on the top right of the screen opens an overlay with filter preferences (i.e. zip, genre, city, size).



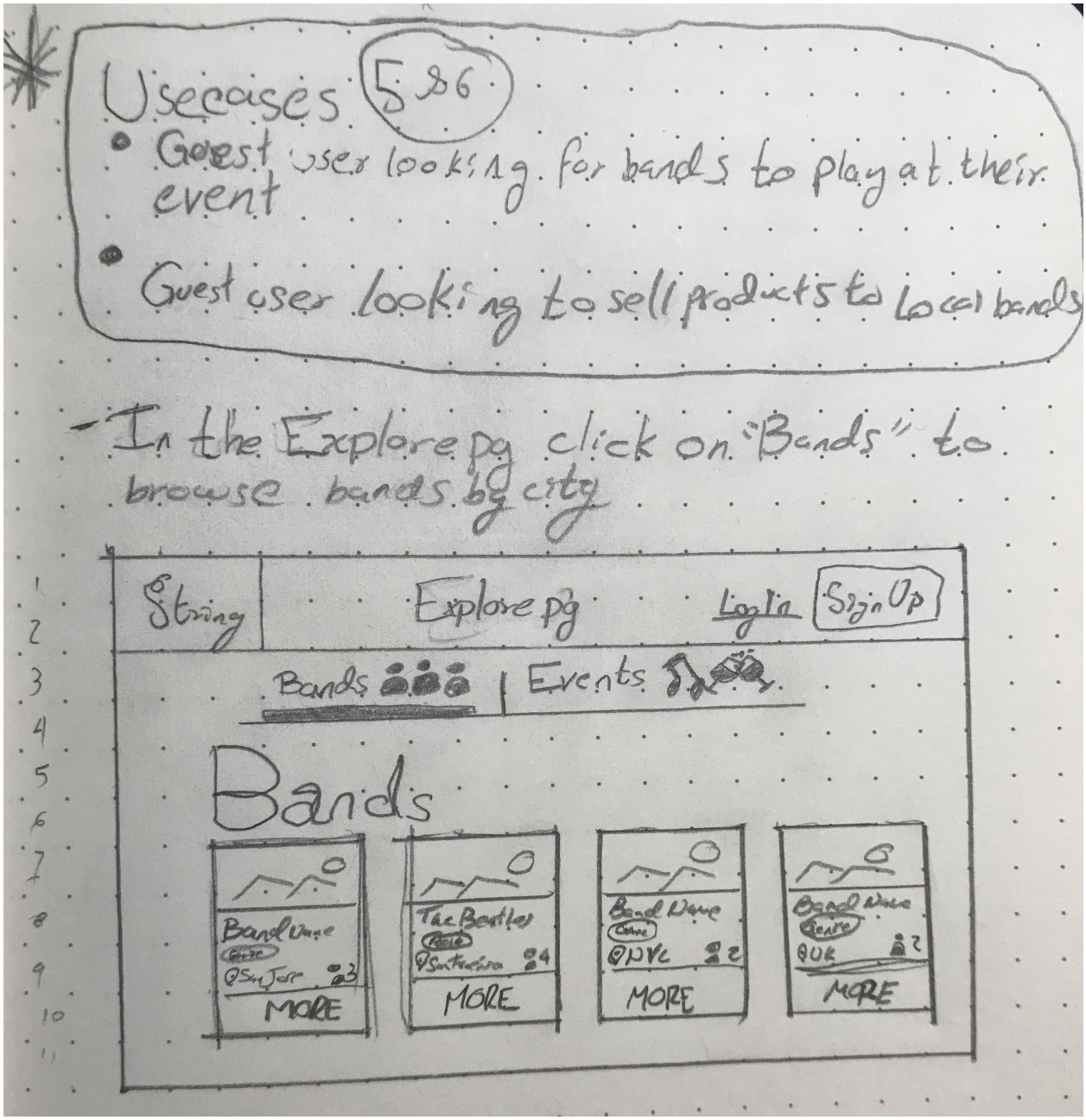
3.4 Use Case 4: A band looking for musicians

1. A band manager enables "looking for new members" checkbox on the band page
2. An envelope icon appears next to the checked checkbox and would display a red bubble with the amount of requests from users interested shall anyone click on the "apply" button on their page


Usecase 4. A band looking for musicians
String
About Explore
John Lennon
The Beatles
3
Rock
San Francisco
About Music Repertoire Posts Events
Members
Looking for new members
- A band manager enables "looking for new members" checkbox on the band page
Looking for new members
now Hiring
- An envelope icon appears next to the checked checkbox, and would display a red bubble with the amount of requests from users interested shall anyone click on the "apply" button on their page

3.5 Use Cases 5 & 6: Guest user looking to sell products to local bands (looking for contact info), Guest user looking for bands to play at their event

1. In the explore page tap the Bands button to browse Bands by city.
2. Scroll down to find more.
3. For filtering results of any kind the filter button located on the top right of the screen opens an overlay with filter preferences (i.e. zip, genre, city, size).
4. By tapping on either one result the guest user is funneled to the respective bands page.
5. Guests can reach out directly to the bands that have their contact information set as public.


Usecases
5 & 6
• Guest user looking for bands to play at their event
• Guest user looking to sell products to local bands
- In the Explore pg click on "Bands" to browse bands by city
String
Explore pg
Log In
Sign Up
Bands
Events
Bands
MORE
MORE
MORE
MORE

4 High level database architecture and organization

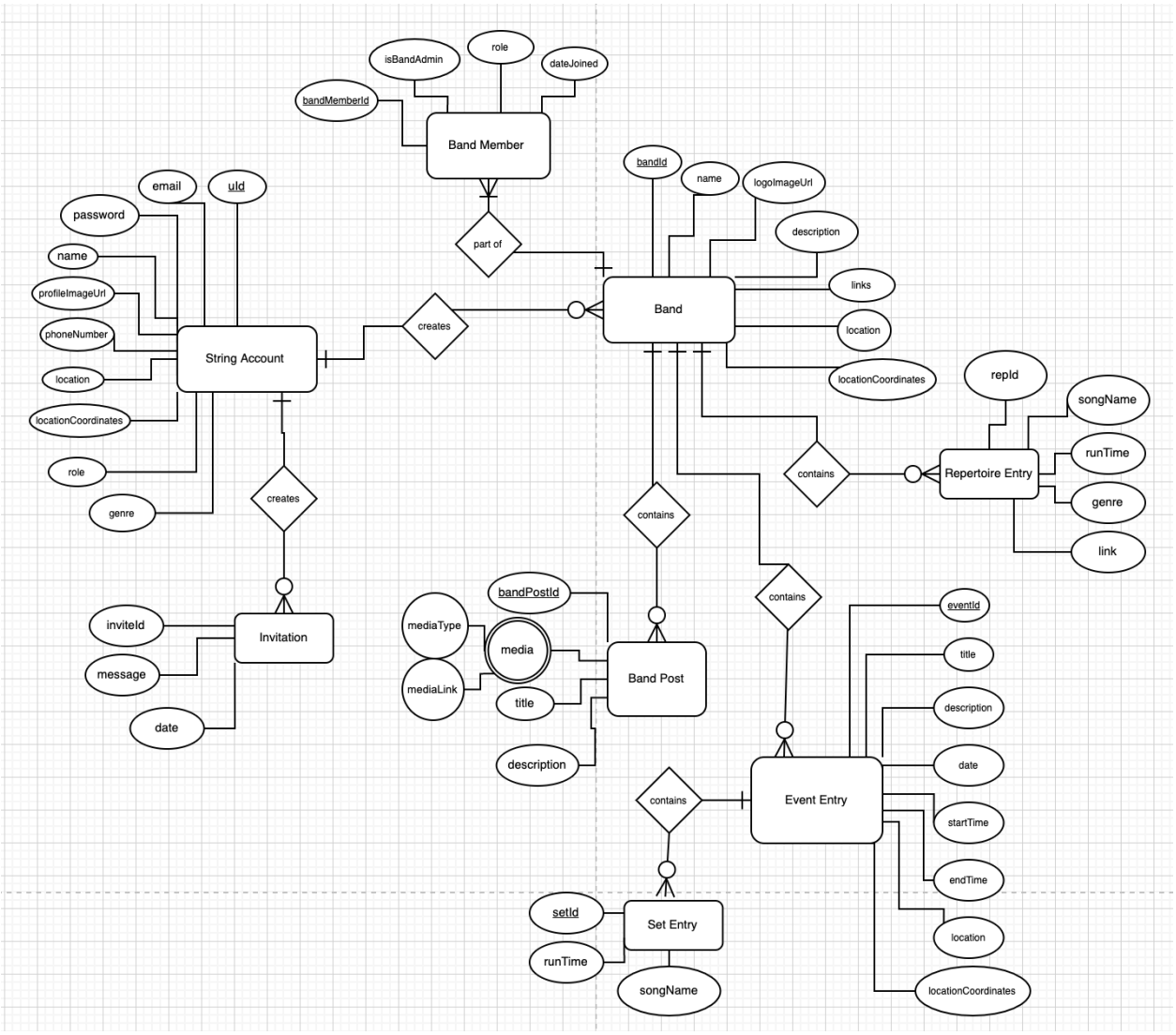
4.1 Business Rules

1. One Guest(with a email) can have one String Account.
2. Each String Account can be a part of zero or more Band(s).
3. Each Band can have one or more Band member(s)
4. Each Band can have zero or more Event Entry.
5. Each Event can have zero or more Event Entry.
6. Each Band can have zero or more Repertoire Entry.
7. Each Band can have zero or more Band Post.
8. Each String Account can send zero or more invitation(s)

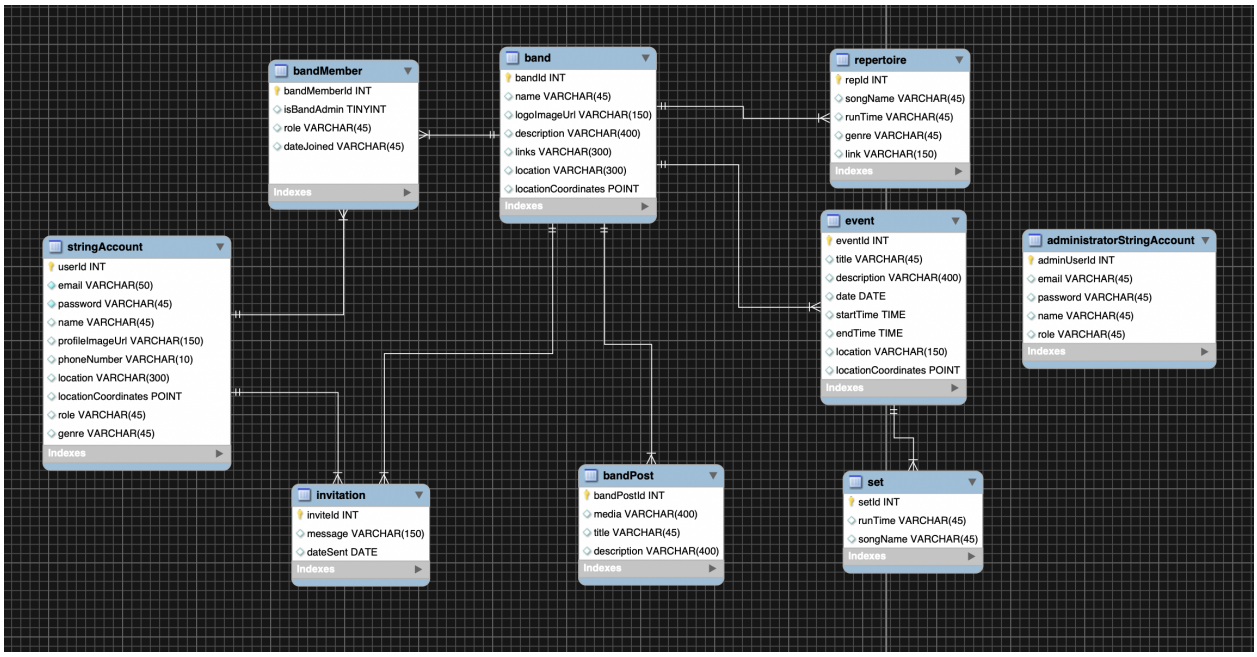
4.2 Entities Description

Same as described in Data Definitions V2 (section 1 of this document)

4.3 ERD


isBandAdmin
role
dateJoined
bandMemberId
Band Member
part of
email
uId
password
name
profileImageUrl
phoneNumber
location
locationCoordinates
role
genre
String Account
creates
creates
inviteId
message
date
Invitation
bandId
name
logoImageUrl
description
links
location
locationCoordinates
Band
contains
contains
repId
songName
runTime
genre
link
Repertoire Entry
contains
bandPostId
mediaType
media
mediaLink
title
description
Band Post
contains
eventId
title
description
date
startTime
endTime
location
locationCoordinates
Event Entry
contains
setId
runTime
songName
Set Entry

4.4 Database model



We chose to use MySQL database because most of our team members were familiar with it and AWS RDS offered MySQL server.

4.5 Media Storage

We will be using S3 buckets to store profile pictures of Individuals and Bands.

For image files we will accept files with extension jpg, jpeg, png and gif.

The size of the file should be less than 10mb.

We will not be storing videos but if Bands want to upload video on their Band Post they will upload a link to the video uploaded on other websites like youtube.

4.6 Search/filter architecture and implementation

We will have a search feature for Bands and for events. The searching will be through band name and event title. The search function will use SQL like and % functions to get results matching search query.

We will have some filters out of which location filter will be the most important. We take the location of all Bands and of registered users which will be converted to coordinates using Here map's api which will be stored in the database as a POINT element. We would then use MySQL geospacial functions to sort using this point and the user's location coordinates.

5 High Level APIs and Main Algorithms

- /api/register
  - POST request
  - Will be used to create a new String account
- /api/login
  - POST request
  - Will be used to log into an existing String account
- /api/getBands
  - POST request
  - Will be used to get Bands according to search value and filters applied
- /api/getEvents
  - POST request
  - Will be used to get Events according to search value and filters applied
- /api/getBand?bandId
  - GET request
  - Will be used to get Band corresponding to bandId
- /api/getEvent?eventId
  - GET request
  - Will be used to get Event corresponding to eventId

The following api are only accessible to a user logged into their string account:

- /api/user/changeLocation
  - POST request
  - Will be used to change the location of the registered user
- /api/band/create

  - POST request
  - Will be used to create a band

- /api/band/invite/send
  - POST request
  - Will be used to create an invitation to join a band

The following api are only accessible to a band member of a band:

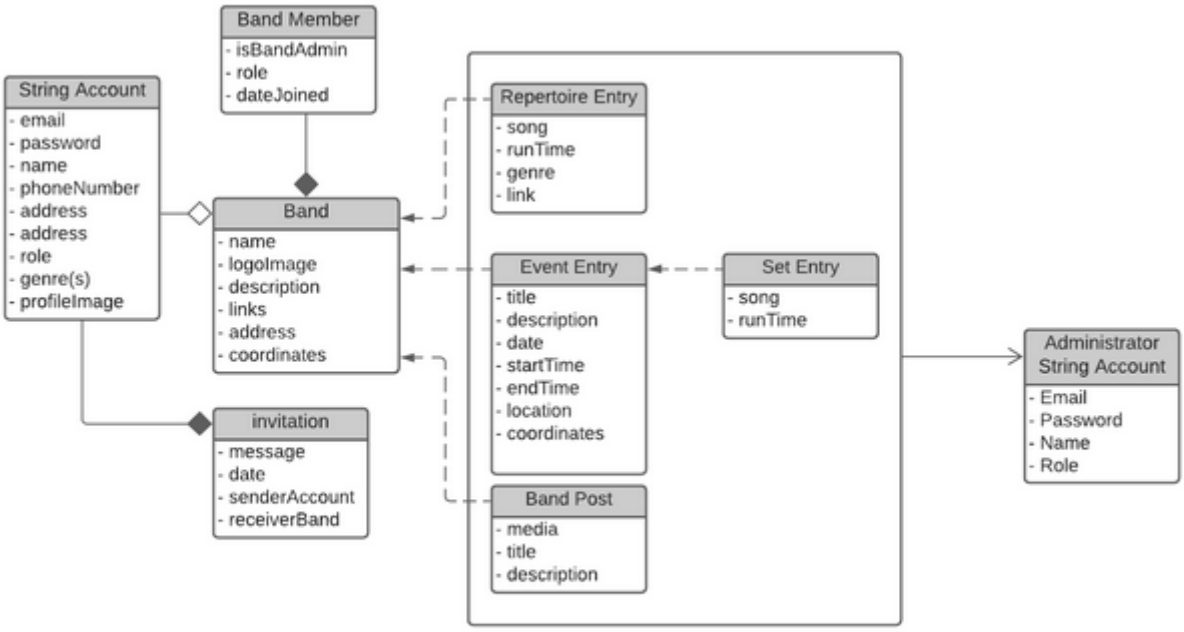
- /api/band/rep/create
  - POST request
  - Will be used to create a repertoire entry for a band
- /api/band/event/create
  - POST request
  - Will be used to create an event entry for a band
- /api/band/post/create
  - POST request
  - Will be used to create a band post for a band
- /api/band/rep/delete
  - POST request
  - Will be used to delete a repertoire entry for a band
- /api/band/event/delete
  - POST request
  - Will be used to delete an event entry for a band
- /api/band/post/delete
  - POST request
  - Will be used to delete a band post for a band

The following api are only accessible to a Band Admin of a band:

- /api/band/delete
  - POST request
  - Will be used to delete a band
- /api/band/invite/accept
  - POST request
  - Will be used to accept an invite to join a band from a registered user

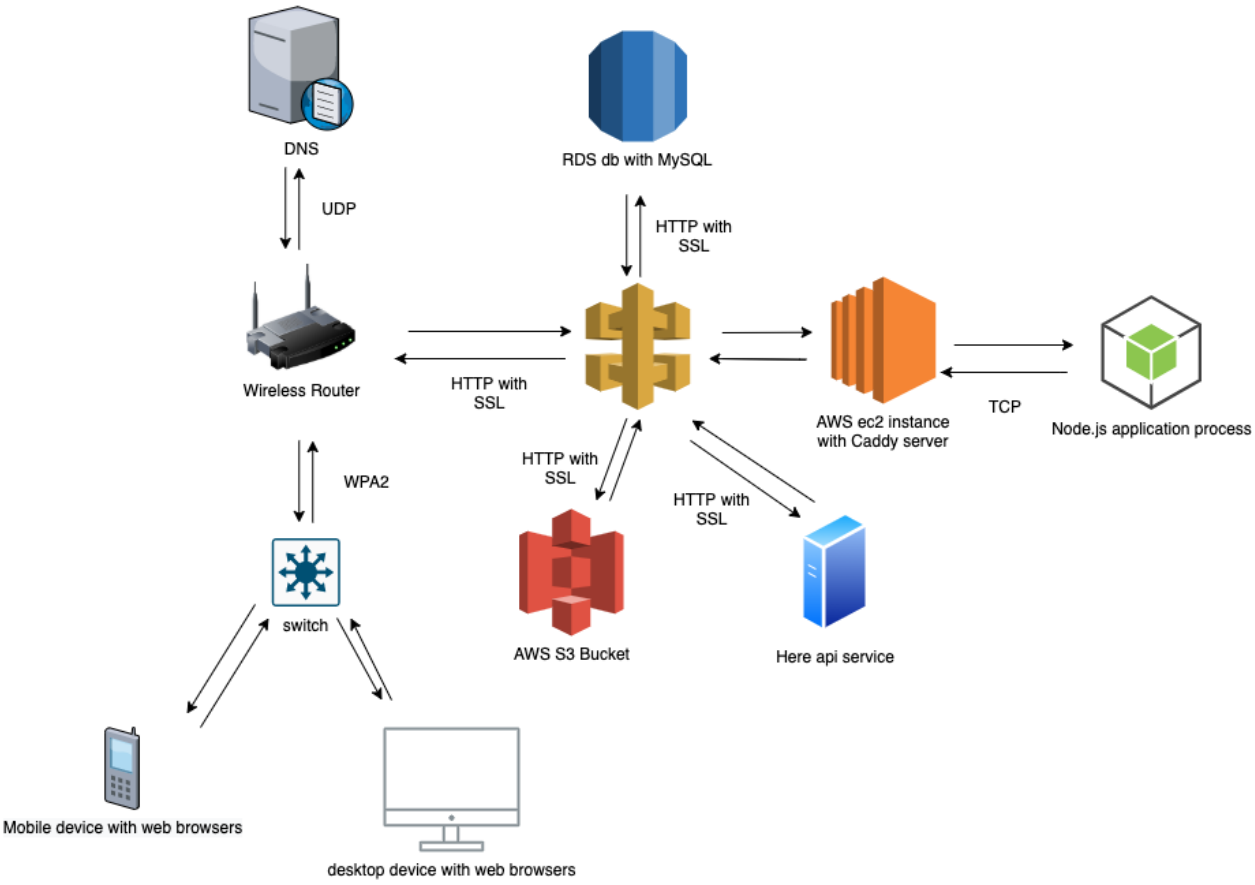
We have not changed any of our existing frameworks but have added AWS S3 to store image assets and will be using Here api to convert a location to its coordinates.

6 High Level UML Diagrams

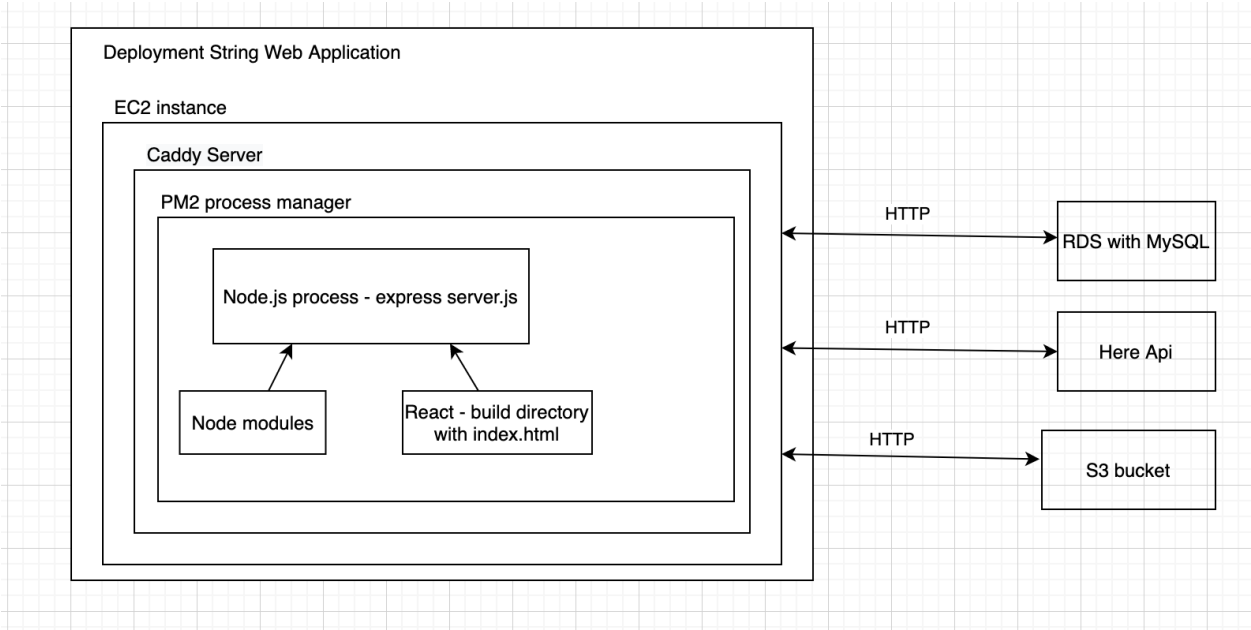


7 High Level Application Network and Deployment Diagrams

7.1 Application Networks Diagram



7.2 Deployment Diagram



8 Key Risks at time of writing

8.1 Skills risks

- No risks as currently assessed

8.2 Schedule risks

- Time shortage: Our project has a complex UI with different user views and can take longer than a month to complete
  - Complete priority 1 and basic functioning of the app before moving to priority 2 or optimising the app UI.

8.3 Technical risks

- Working with multer and s3 buckets could be challenging to manage images and assets.
  - Use youtube tutorials to follow along how to upload images using aws sdk and multer, such as this.
- Using redux to store the state persistently between screens has a steep learning curve.
  - Use boilerplate redux and change the functionality and methods to work with our requirements.

8.4 Teamwork risks

- No risks as currently assessed.

8.5 Legal/content risks

- Band uploading inappropriate content on the String platform.
  - Legal document with behavioral guidelines that only content that is legally permissible and age appropriate should be uploaded or used on the platform.
  - Use Administrator String account to monitor the postings and remove them if they are not in accordance with the terms & conditions of the platform.
- Band copying other band's work(images/videos) or Fraudulent bands

  - Our solution to this would be a set of legally binding terms users agree to when they register stating all work uploaded is their own and have rights to, and String is not responsible for any content that is not such.

- Safety of guests in the case of a fraudulent event (or other issues with the information posted on the platform)
  - Legal document that String is not responsible for user safety, that the user acknowledges that they are ultimately responsible for themselves, and that String's responsibility ends with hosting the information as received from the originator(s).

- Setting up the Terms and Condition documents.
  - Use boilerplate Terms and Conditions document and change it according to our needs and specifications.
  - In the event of further application development, legal consulting is a possibility.

9 Project Management

- We are using Trello as our project management tool:
  - We create cards that multiple team members are assigned to and notified of as they change from "current week - TO DO" to "IN PROGRESS" to "Ready for review - MERGE" (on applicable tasks) and finally to "SHIPPED - DONE."
  - On the board we also have a section for "References & Resources" where we keep things like the links to the shared google drive, the zoom room, and the milestone documents.
  - We also create further modifications to cards as applicable, for instance to do lists within cards this makes sure everyone is aware of what other members are doing at any given point.
- We distributed the parts of the milestone amongst the frontend and backend teams:
  - The Frontend team worked on part 3 and frontend of the vertical prototype.
  - The Backend team worked on part 4,5,6,7 and API development of the vertical prototype.
- We did updates before every meeting in order to come to an understanding of what both teams had worked on and were planning on doing until the next meeting
- We completed some parts: 1,2,8,9 together during the meeting time because they concern all team members.
- From next milestone on:
  - We would follow a similar management pattern.
  - Tasks that concern the team will be done during team meetings.
  - Tasks that concern a specific sub team will be done on their time and a report will be presented in the beginning of the meeting.
  - All the tasks will be uploaded on Trello which will be a central place to know who is working on which part of the application and its progress.

10 Detailed list of contributions

- Jainam Shah: (Team Lead)

    1. Organised two weekly meetings and built agenda and plans to complete the milestone in time.
    2. Managed and distributed tasks amongst team members.
    3. Worked on the defining data entities, prioritising functional requirements and indentifying key risks.
    4. Verified and completed the ERD and UML diagrams and database structure.
    5. Created the S3 and file management part of the backend and built a home screen as per design for the vertical prototype.
    6. Deployed the vertical prototype on the EC2 instance.

- Alfredo Diaz: (Frontend lead)

    1. Timely attendance and participation in discussions during meetings.
    2. Worked on the defining data entities, prioritising functional requirements and indentifying key risks.
    3. Created user journeys and clarified functionality of components on main screens and created the paper UI Mockups.
    4. Coordinated with the fronted team what components we could use from existing libraries & assigned tasks to build basic components for vertical prototype.
    5. Shared card component vision by handing off high fidelity wireframes for vertical prototype.

- Leonid Novoselov: (Frontend developer)

    1. Participation in discussions during meetings.
    2. Worked on the defining data entities, prioritising functional requirements and indentifying key risks.
    3. Worked on creating paper UI mockups and storyboards.
    4. Worked on the display card component for the vertical prototype.

- Eric Chen: (Frontend developer)

    1. Attended weekly meetings and participated in discussions.

2. Worked on the defining data entities, prioritising functional requirements and indentifying key risks.

- Warren Singh: (Backend lead)

1. Timely attendance and participation in discussions during meetings.
2. Worked on the defining data entities, prioritising functional requirements and indentifying key risks.
3. Helped create Business Rules and database design.
4. Edited and Compiled M2 document
5. Coordinated with back end team on workflow for vertical prototype.
6. Contributed in defining routes for the vertical prototype.

- Ritesh Panta: (Backend developer)

1. Timely attendance and participation in discussions during meetings.
2. Worked on the defining data entities, prioritising functional requirements and indentifying key risks.
3. Contributed in defining business rules and creating network and deployment diagrams.
4. Contributed in building the MYSQL queries for the vertical prototype.

2.3 Milestone 3

SW Engineering CSC 648/848 - FALL 2020

Milestone 3

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 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 1, 2020

Contents

1 Data Definitions V2

1. String Account: Account of a Registered user on the platform.
   (a) Email - an email address of the Registered User
   (b) Password - used to log into the String Account.
   (c) Name - the name of the Registered User.
   (d) Phone number - (optional) phone number of the Registered User.
   (e) Location - (optional) the address of the Registered User.
   (f) Location Coordinates - (optional) the address coordinates of the Registered User.
   (g) Role - (optional) list of roles the Registered User can perform.
   (h) Genre(s) - (optional) list of primary musical interests (i.e. classical, jazz, rock, electronic)
   (i) Profile picture - (optional) Registered User's picture.
2. Administrator String Account: Account of an Administrator on the platform (i.e. the team building and managing the String Platform).
   (a) Email: an email address of the Administrator.
   (b) Password: used to log into the Administrator String Account.
   (c) Name: the name of the Administrator.
   (d) Role: role of the Administrator in the team (i.e frontend lead, team lead, content manager, etc.).
3. Registered User: A person who has a String account and is logged into that account
4. Guest: A person who does not have a String account and is just viewing pages on our platform
5. Administrator: A person who has an Administrator String account and is logged into that account. They have access over all information, items and entities on the platform. They are the ones who will manage inappropriate content.
6. Band Member: A registered user who joins a Band, with a Band Admin's permission.
   (a) Band Admin: if the Band member has admin access in the band. (a boolean value)
   (b) Role: Role that the Band Member has in the Band.
   (c) Date joined: When the Band Member joined the Band.

7. Band: A band can be created by a registered user. It is an entity that other registered users can join if Band Admin permits.
    (a) Band Name: (required) Name of the band. (unique)
    (b) Band logo image: Logo image of the Band.
    (c) Band Description: A short description of what the band is about.
    (d) Band Links: list of links.
        i. Link: the link pointing to another webpage.
        ii. Short description: a small description of the link.
    (e) Band Post: Image or video uploaded by bands.
        i. Media: Link pointing to an image or a video (could be on youtube or Vimeo or Google Drive).
            A. Type of media: either image or video.
            B. Link: link to that image or video.
        ii. Title (required): title of what the Band Post is about.
        iii. Description (optional): description of the Band Post
    (f) Location (required): Address that the band belongs to and where they are performing.
    (g) Location Coordinates (required): latitude and longitude of the address of the Band
    (h) Event Entry: A band can create an entity known as an event in order to provide information on an upcoming rehearsal, or performance - with sections for information and Set List.
        i. Event title: (required) The title of the event.
        ii. Date of event: (required) month day and year the event is taking place.
        iii. Start Time of event: (required) time at which the event will begin.
        iv. End time of event: (required) time at which event will end.
        v. Location of event (required): Address and description of where the event will be held.
        vi. Coordinates: (required) latitude and longitude of the address of the Event
        vii. Event description: description of the event.
        viii. Set Entry: A song which the band is planning to play for a specific event/performance. Includes details such as run time, tempo, and order in performance.
            A. Song: name of the song.
            B. Run time: length of the song.

(i) Repertoire Entry: A song which the band can play, including details such as:

i. Song - name of the song.

ii. Run time - length of the song.

iii. Genre - genre of the song.

iv. Link - Link to a site with more description/media about the performance of this song.

8. Invitation: Sent by a registered user to join a band in response to availability post

(a) Message: A short paragraph of why the user would want to join the Band

(b) Date: Date on which the Invitation was sent.

(c) Sender account: the String Account who sent the invitation

(d) Receiver band: the Band that the invitation was sent to.

2 Functional Requirements V2

2.1 Priority 1

2.1.1 Guest

1. Guest shall be able to view all Bands.
2. Guest shall be able to view all Band Members.
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 access contact information of all Bands.
6. Guest shall be able to search for any Band by name.
7. Guest shall be able to filter any Bands by genre.
8. Guest shall be able to filter Bands by location.
9. Guest shall be able to register an account with a valid email address.
10. Guest shall be able to view all Events' Set List.
11. Guest shall be able to view all Bands' Repertoire List.

2.1.2 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 view all the Bands information they are a Band Member of.
8. Registered User shall be able to change their password in their String Account.

9. Registered User shall be able to edit the field indicating their name in their String Account.
10. Registered User shall be able to edit the field indicating their role.
11. Registered User shall be able to edit their current geographical location in their String Account.
12. Registered User shall be able to edit their contact information in their String Account.

2.1.3 Band

1. Band can only 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 contact information.
8. Band shall have an Event List indicating upcoming events.
9. Band shall have a Repertoire List.

2.1.4 Band Member

1. Band Member should 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 Set Entry to an Event Entry of their Band.
8. Band Member shall be able to delete a Set Entry to an Event Entry of their Band.
9. Band Member shall be able to add a Repertoire Entry to their Band.
10. Band Member shall be able to delete a Repertoire Entry to their Band.

2.1.5 Band Admin

1. Band Admin should be a Band member.
2. Band Admin shall be able to remove a Band member of their Band.
3. Band Admin shall be able to view all outgoing and incoming Invitations in their Band.
4. Band Admin shall be able to issue or accept an outgoing or incoming Invitation in their Band.
5. Band Admin shall be able to delete any outgoing or incoming Invitation in their Band.

2.1.6 Band Post

1. Band Post shall contain a field for an image.
2. Band Post shall contain a field for a link to a video.
3. Band Post shall contain a field for a description.
4. Band Post shall contain a title.

2.1.7 Invitation

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 incoming Invitation shall result in adding the Registered User to the Band as a Band member.

2.1.8 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 what time the event will take place.
5. Event Entry shall have a field indicating which Band created it.

6. Event Entry shall have a description field.

7. Event Entry shall have a list of Set Entries.

2.1.9 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.

2.1.10 Set Entry

1. Set Entry shall have a field indicating song's name.

2. Set Entry shall have a field indicating song's run time.

3. Set Entry shall have a field indicating order in the playlist.

2.2 Priority 2

2.2.1 Guest

1. Guests shall be able to add reviews to a Band.

2. Guests shall be able to play songs linked in Repertoire Entries.

2.2.2 Registered User

1. Registered Users shall be able to change their email details in their String Account.

2. Registered Users shall be able to like a band post.

3. Registered Users shall be able to add a comment on a band post.

4. Registered Users shall be able to RSVP to an event.

2.2.3 Administrator

1. Administrator shall be able to change their email in their Administrator String Account.

2. Administrator shall be able to change their password in their Administrator String Account.

3. Administrator shall be able to delete any Set List in any Event Entry in any Band.

4. Administrator shall be able to delete any Repertoire Entry in any Band.
5. Administrator shall be able to log in using their email and password.
6. Administrator shall be able to delete any Band.
7. Administrator shall be able to delete any content posted by any Band.
8. Administrator shall be able to delete any Event Entry in any Band

2.2.4 Band Member

1. Band Members shall be able to edit an Event Entry in the Event List of their Band.
2. Band Members shall be able to edit a Set List to an Event Entry of their Band.
3. Band Members shall be able to edit a Repertoire Entry in the Repertoire List of their Band.

2.2.5 Band Admin

1. Band Admins shall be able to assign a different Band Member as Band Admin.
2. Band Admins shall be able to change the name of their Band.

2.2.6 Event Entry

1. Event Entry shall have an image
2. Event Entry shall have an RSVP list

2.2.7 Repertoire Entry

1. Repertoire Entry shall have a playable song link.

2.2.8 Set Entry

1. Set List shall contain a list of Set Entries.
2. Set List must contain at least one Set Entry
3. Set List shall have Set Entries in projected performance order.

2.2.9 Repertoire List

1. Repertoire List shall contain an (unordered) list of Repertoire Entries

2.2.10 Band Post

1. Band Post shall have a displayed count of likes.

2. Band Posts shall have a comment section for participation from Registered Users.

2.3 Priority 3

2.3.1 Guest

1. Guest shall be able to contact an Administrator to get help at any time.

2.3.2 Registered User

1. Registered User shall be able to delete their String Account.

2. Registered User shall be able to add comments to an Event.

2.3.3 Administrator

1. Administrator shall be able to create a Band.

2. Administrator shall be able to delete any Registered User.

3. Administrator shall be able to add registered users.

4. Administrator shall be able to add registered users to any Band.

5. Administrator shall be able to remove any Band Member in any Band.

6. Administrator shall be able to change Band Admin in any Band.

2.3.4 Band

1. Band which have less than one Band Member shall be deleted.

2. Band shall have one or more Band Admin(s).

2.3.5 Band Member

1. Band Members shall have access to a text chat viewable only by Band Members.

2. Band Members shall be able to invite Registered Users to an Event.

2.3.6 Band Admin

1. Band Admins shall be able to delete their Band.

2. Band Admins shall be able to make other Band Members to be an additional Band Admin.

2.3.7 Repertoire Entry

1. Repertoire Entry shall have a list of links.

2. Repertoire Entry shall have a field indicating if the song is in the rehearsal stages.

3. Repertoire Entry shall have a field for a short description of or notes on the song.

2.3.8 Event Entry

1. Event Entry shall have a list of registered users invited to the event.

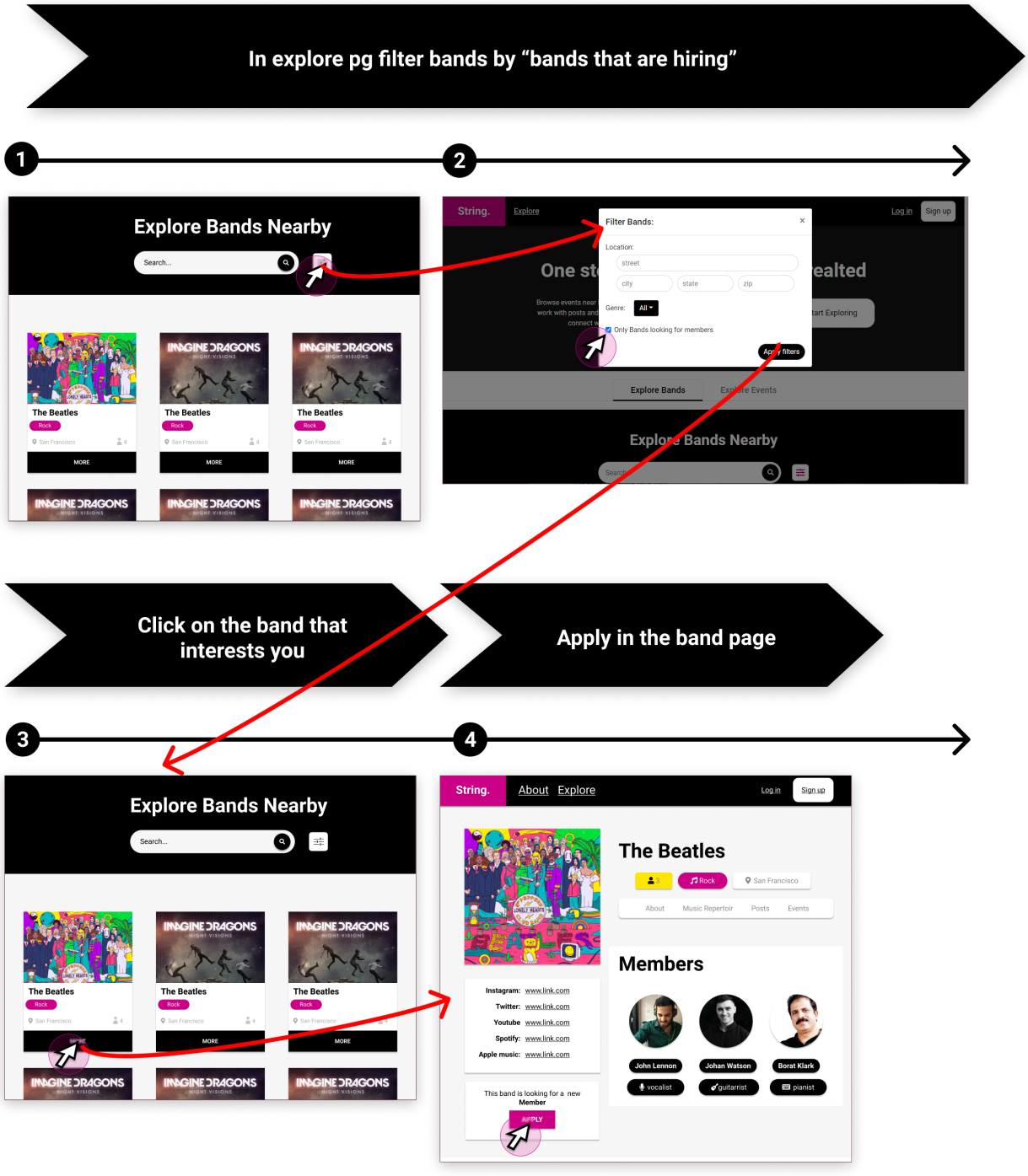
2. Event Entry shall have a section for Registered Users to comment and discuss.

3 Wireframes

3.1 Use Case 1: A solo artist looking for a band to join

1. A registered user as solo artist goes to the explore page and taps the filter button
2. In the filter modal user selects to browse Bands by “Bands that are hiring”.
3. Scroll down to find more
4. The user gets a list of all the bands in their city that are looking for new members
5. Solo artists can access the band pages
6. Inside the band pages registered users can click the “apply” apply button to send a notice of interest on the role to that band
7. Alternatively, they can reach out directly to the bands that have their contact information set as public

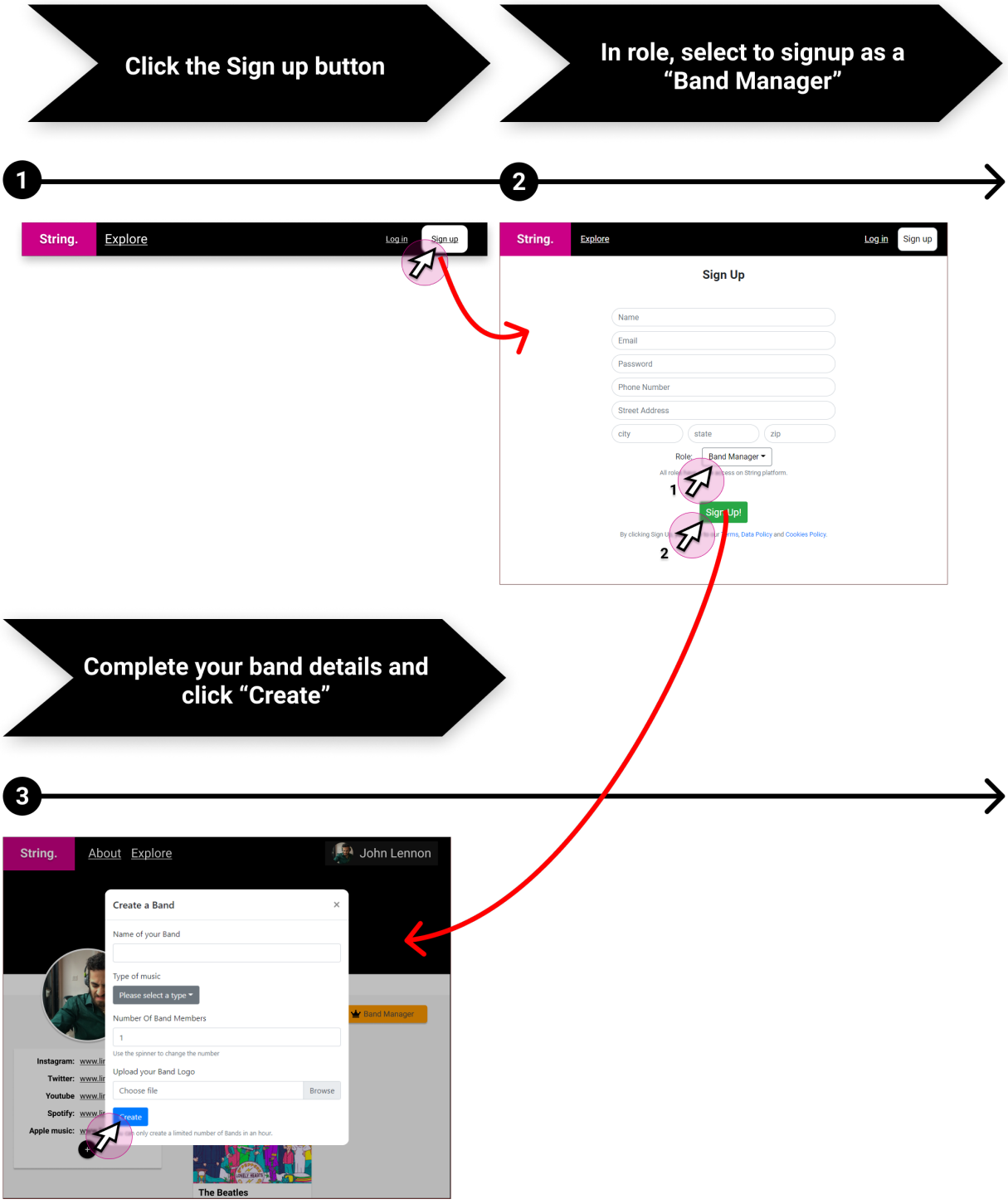
Usecase 1:
A solo artist looking to join a band



3.2 Use Case 2: A band manager onboards, searching to have one place manage/display all bands he manages

1. User clicks on the sign up button in the explore page or any other public page.
2. Onboarding user inputs: name, last name, email, phone number, and password, address, role, instrument and genre
3. Users select onboard as either: a solo artist, a manager, or a band.
4. Users that select to sign ups as “Band Manager” are prompted to create a band after signup
5. The user may complete the details for the band page or finish later (Band name,logo/image, street address, genre, description, links, events, posts, music repertoire)

Usecase 2:
A band manager onboards and create a bandon String

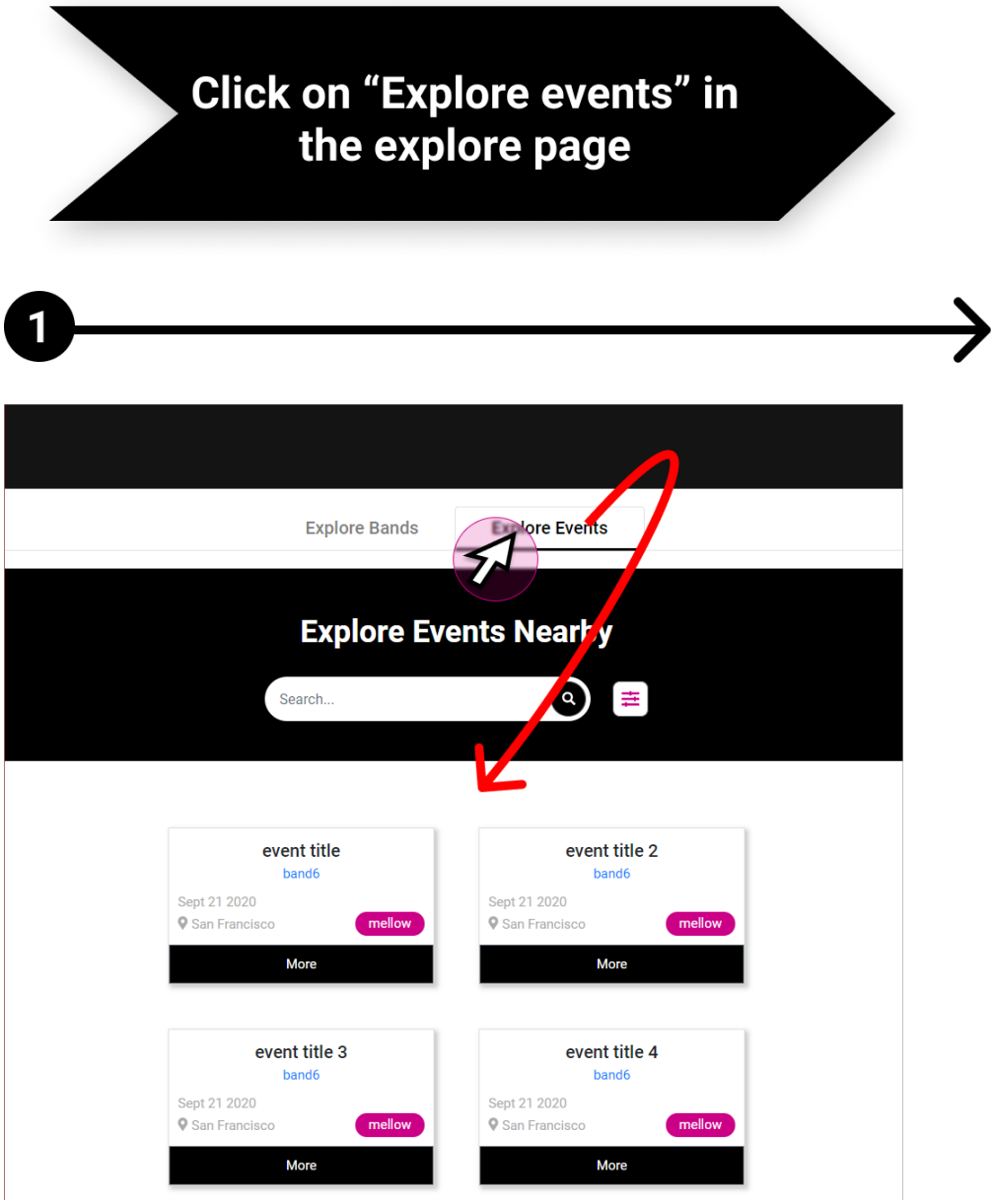


3.3 Use Case 3: Looking to discover music events in their area (guest journey)

1. In the explore page tap the Events button to browse events by city.
2. Scroll down to find more.
3. For filtering results of any kind the filter button located on the top right of the screen opens an overlay with filter preferences (i.e. zip, genre, city, size).

Usecase 3:

Looking to discover events in the area (guest journey)

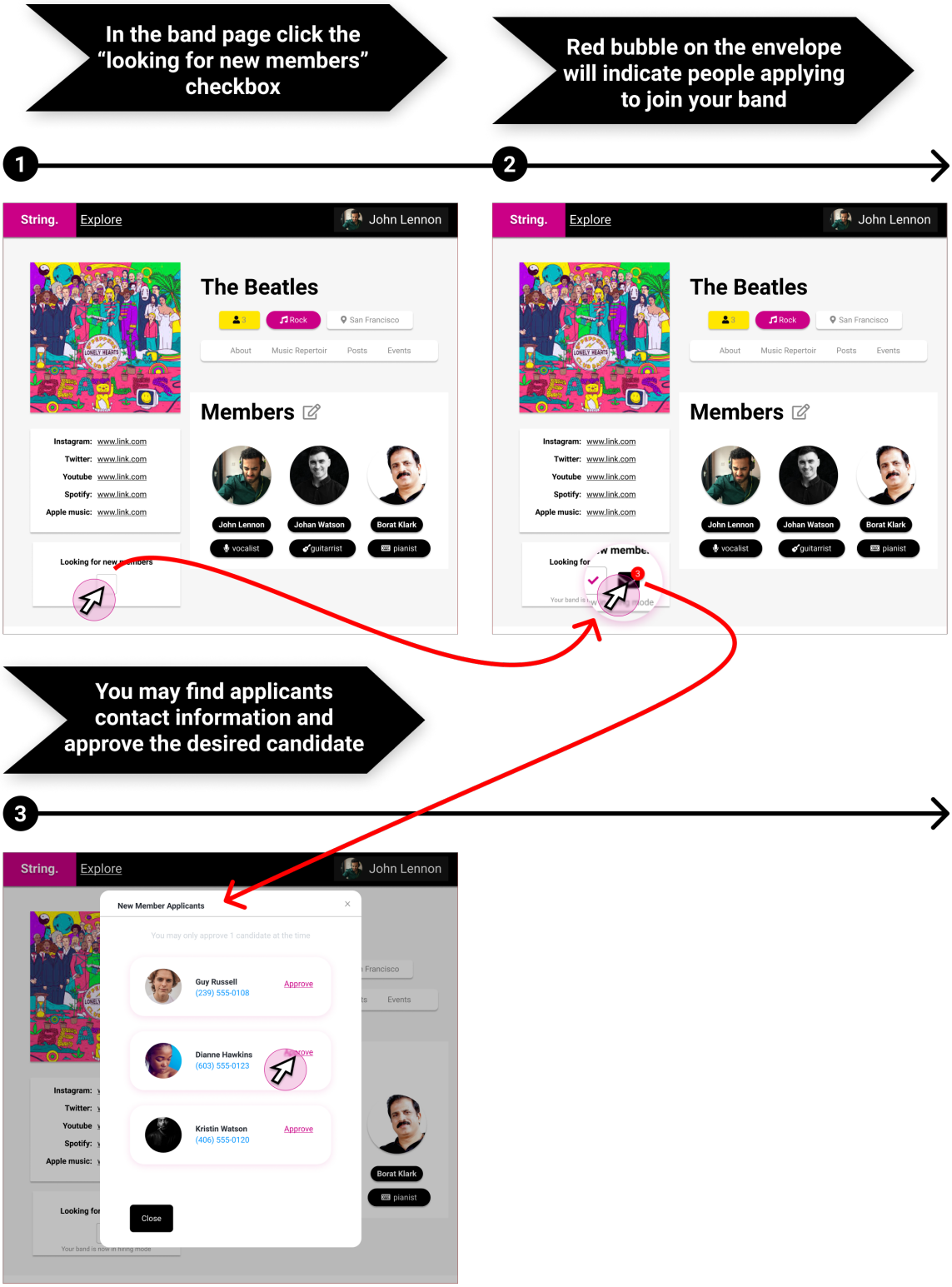


3.4 Use Case 4: A band looking for musicians

1. A band manager enables "looking for new members" checkbox on the band page
2. An envelope icon appears next to the checked checkbox and would display a red bubble with the amount of requests from users interested shall anyone click on the "apply" button on their page

Usecase 4:

A band looking for musicians



3.5 Use Cases 5 & 6: Guest user looking to sell products to local bands (looking for contact info), Guest user looking for bands to play at their event

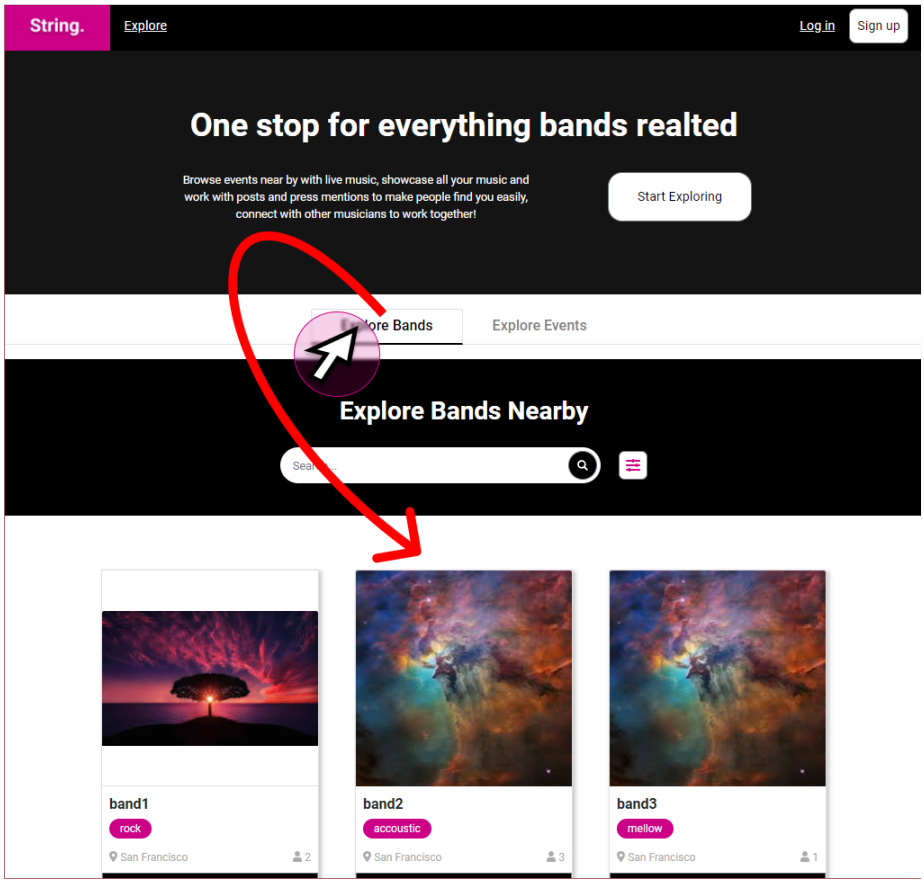
1. In the explore page tap the Bands button to browse Bands by city.
2. Scroll down to find more.
3. For filtering results of any kind the filter button located on the top right of the screen opens an overlay with filter preferences (i.e. zip, genre, city, size).
4. By tapping on either one result the guest user is funneled to the respective bands page.
5. Guests can reach out directly to the bands that have their contact information set as public.

Usecase 5 & 6:

- Guests looking for bands to play at their event/venue
- Guests looking to sell products to local bands

Click on “Explore Bands” in the explore page

1



4 High level database architecture and organization

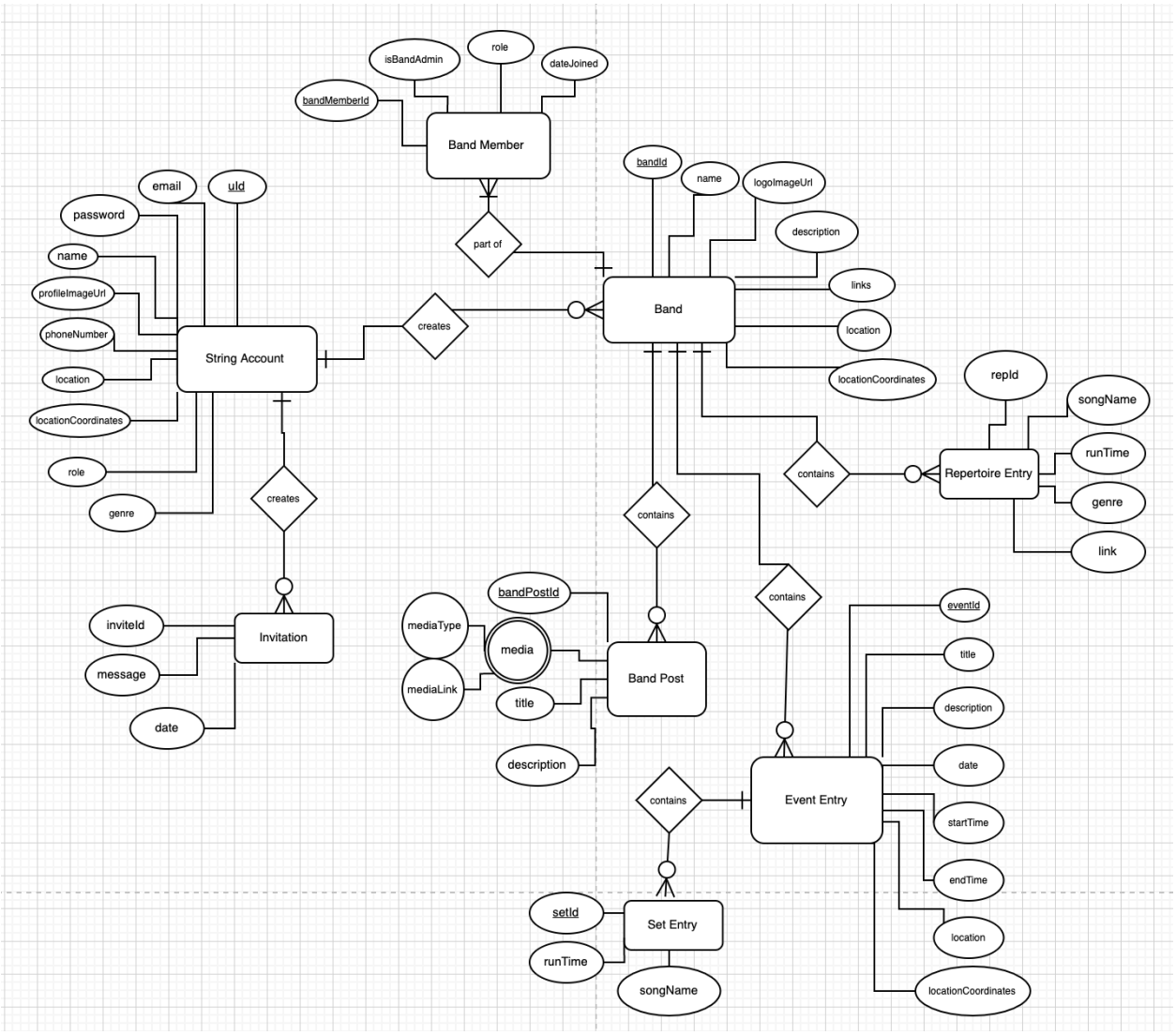
4.1 Business Rules

1. One Guest(with a email) can have one String Account.
2. Each String Account can be a part of zero or more Band(s).
3. Each Band can have one or more Band member(s)
4. Each Band can have zero or more Event Entry.
5. Each Event can have zero or more Event Entry.
6. Each Band can have zero or more Repertoire Entry.
7. Each Band can have zero or more Band Post.
8. Each String Account can send zero or more invitation(s)

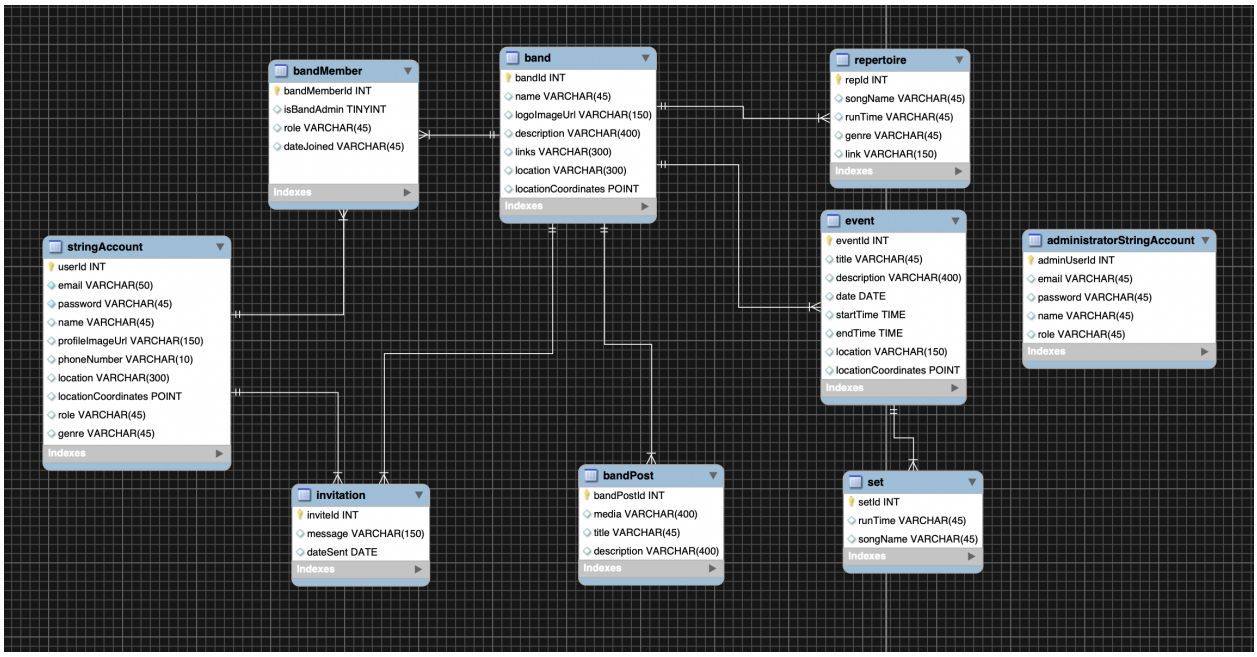
4.2 Entities Description

Same as described in Data Definitions V2 (section 1 of this document)

4.3 ERD


isBandAdmin
role
dateJoined
bandMemberId
Band Member
part of
email
uId
password
name
profileImageUrl
phoneNumber
location
locationCoordinates
role
genre
String Account
creates
creates
inviteId
message
date
Invitation
bandId
name
logoImageUrl
description
links
location
locationCoordinates
Band
contains
contains
contains
repId
songName
runTime
genre
link
Repertoire Entry
bandPostId
mediaType
media
mediaLink
title
description
Band Post
eventId
title
description
date
startTime
endTime
location
locationCoordinates
Event Entry
contains
setId
runTime
songName
Set Entry

4.4 Database model



We chose to use MySQL database because most of our team members were familiar with it and AWS RDS offered MySQL server.

4.5 Media Storage

We will be using S3 buckets to store profile pictures of Individuals and Bands.

For image files we will accept files with extension jpg, jpeg, png and gif.

The size of the file should be less than 10mb.

We will not be storing videos but if Bands want to upload video on their Band Post they will upload a link to the video uploaded on other websites like youtube.

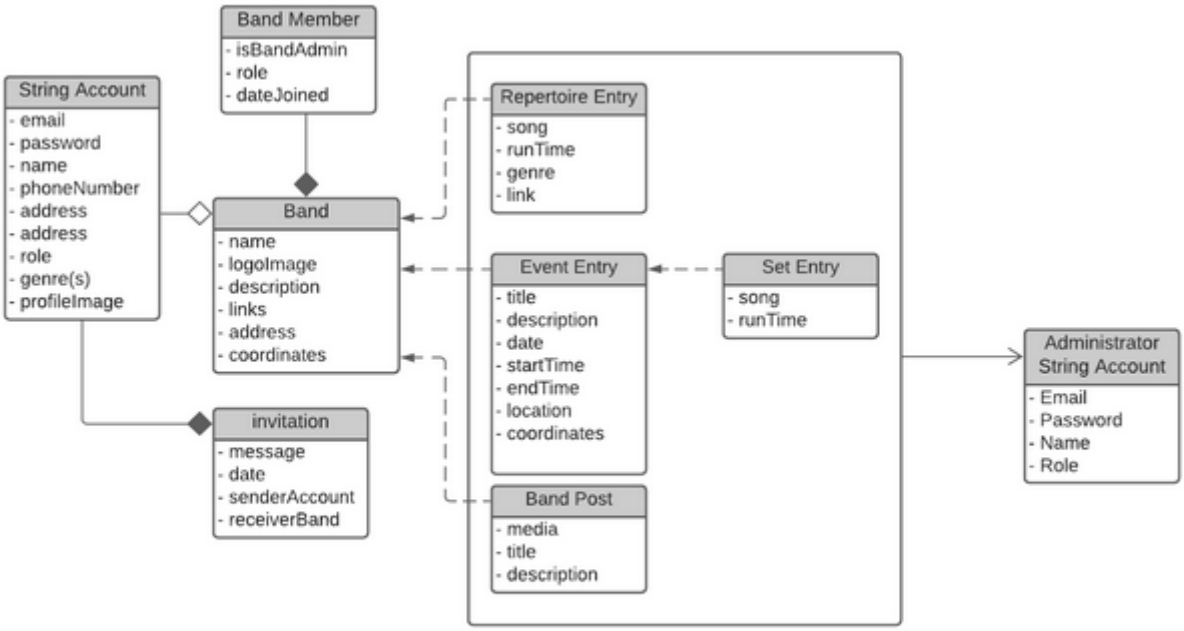
4.6 Search/filter architecture and implementation

We will have a search feature for Bands and for events. The searching will be through band name and event title. The search function will use SQL like and % functions to get results matching search query.

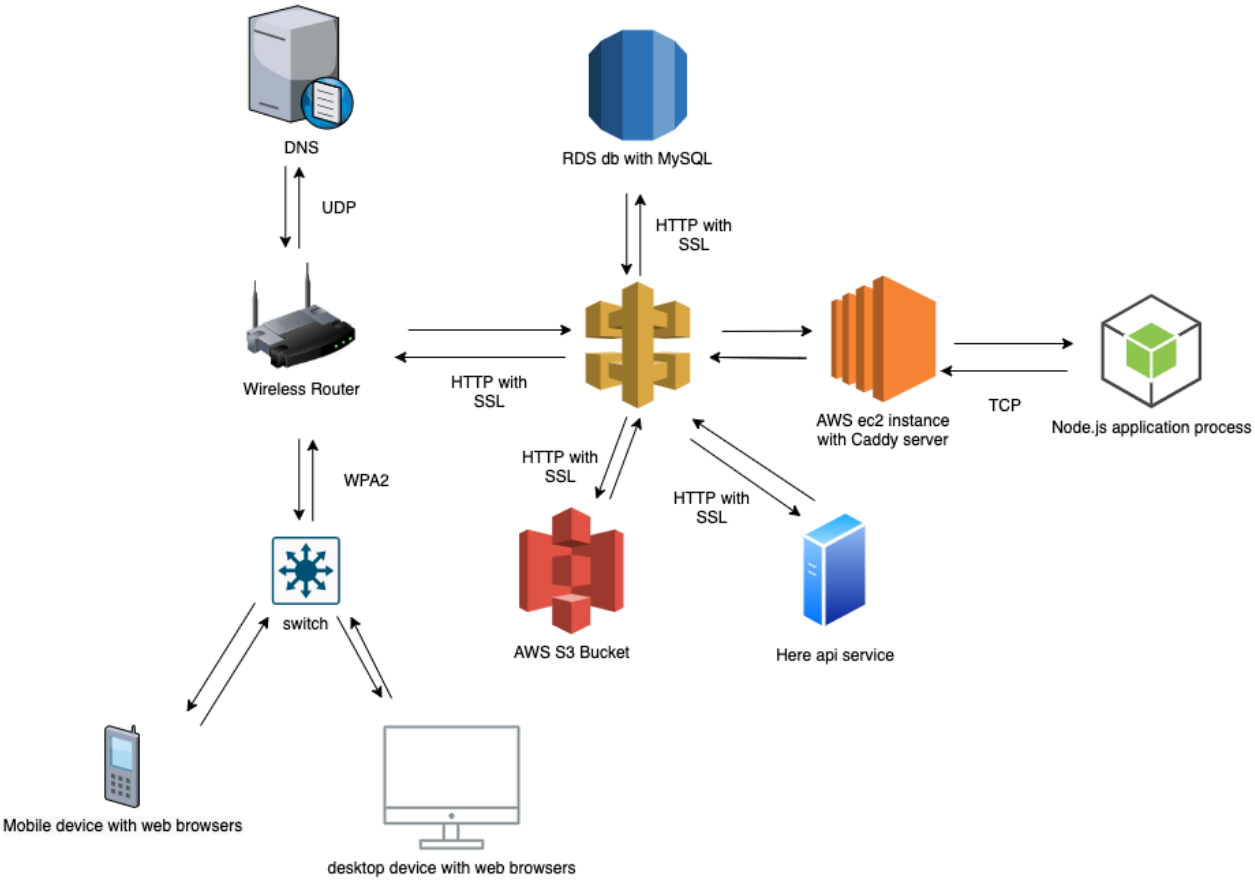
We will have some filters out of which location filter will be the most important. We take the location of all Bands and of registered users which will be converted to coordinates using Here map's api which will be stored in the database as a POINT element. We would then use MySQL geospacial functions to sort using this point and the user's location coordinates.

5 High Level UML, Application Network and Deployment Diagrams

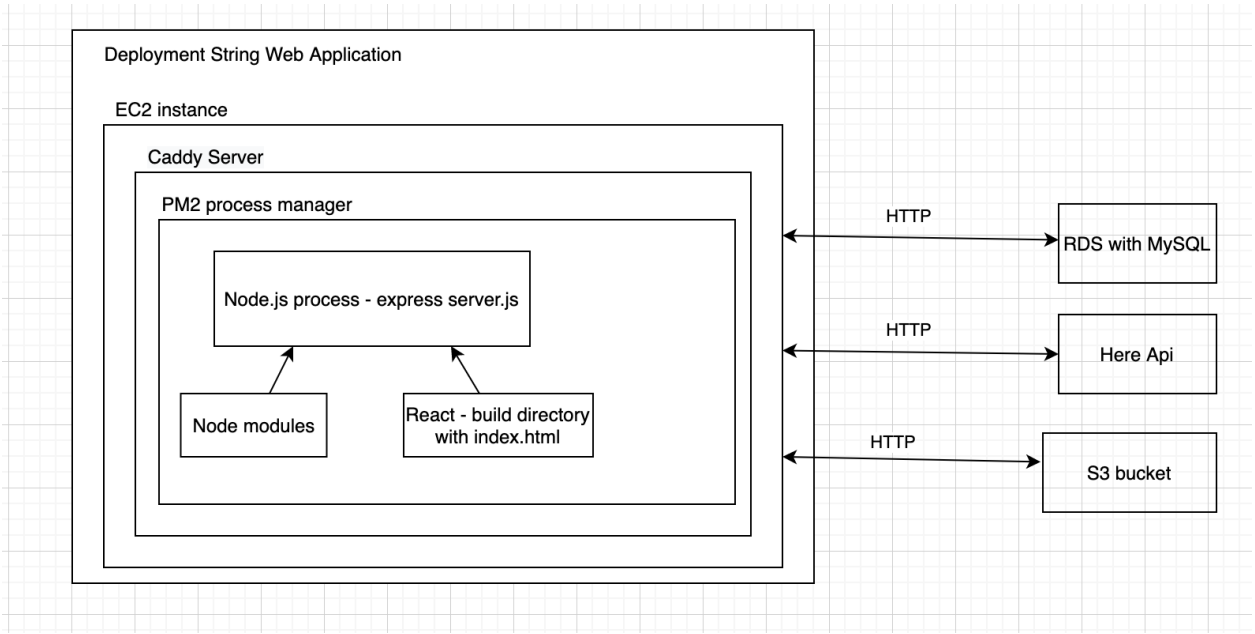
5.1 High Level UML Diagram



5.2 Application Networks Diagram



5.3 Deployment Diagram



6 List of contributions to the three parts of this milestone

- Jainam Shah: (Team Lead)
    1. Conducted meetings and paired programming with team showing and explaining how to make routes, controllers, middlewares and helpers.
    2. Helped provide timely feedback and help on UI design.
    3. Build the auth routes - login, register and logout and band routes - createBand and searchBand.
    4. managed multer, S3 for file handling and passport for auth.
    5. Build the frontend of the application, connected redux and integraded the routes from backend.
    6. Made querry for sorting bands nearest to a given location(based on coordinates) in MySQL.
- Alfredo Diaz: (Frontend lead)
    1. Took lead on designing the UI using Figma and completed all designs by themselves.
    2. Communicated with frontend team the requirements for horizontal prototype.
    3. Build story boards using figma to describpe user interactions in usecases.
    4. Helped building the milestone document, regularly attended the meeting and communicated ideas.
- Leonid Novoselov: (Frontend developer)
    1. Build the Navbar and user profile page frontend from scratch.
    2. Attended meetings regularly, shared their opinions when they could and helped with populating the prototype.
- Eric Chen: (Frontend developer)
    1. Was assigned to complete band profile page but got sick and wasn't able to communicate it.
- Warren Singh: (Backend lead)
    1. Led backend team meetings and did peer programming.
    2. Coordinated meetings with frontend lead to coordinate M2 revisions and M3 layout and changes
    3. Helped build createBand, searchBand route
    4. scafolded createEvent, searchEvents, getBands routes.
    5. Handled formatting and typesetting for M3 and M2 v2 documents, regularly attended the meeting and communicated ideas.

- Ritesh Panta: (Backend developer)

    1. Scafolded and worked on getBandInfo, getEvent, getBandEvent routes.
    2. Helped backend team with building querries in MySQL.
    3. Attended meetings regularly, shared their opinions when they could and helped with populating the prototype.

2.4 Milestone 4

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 14, 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 14, 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.

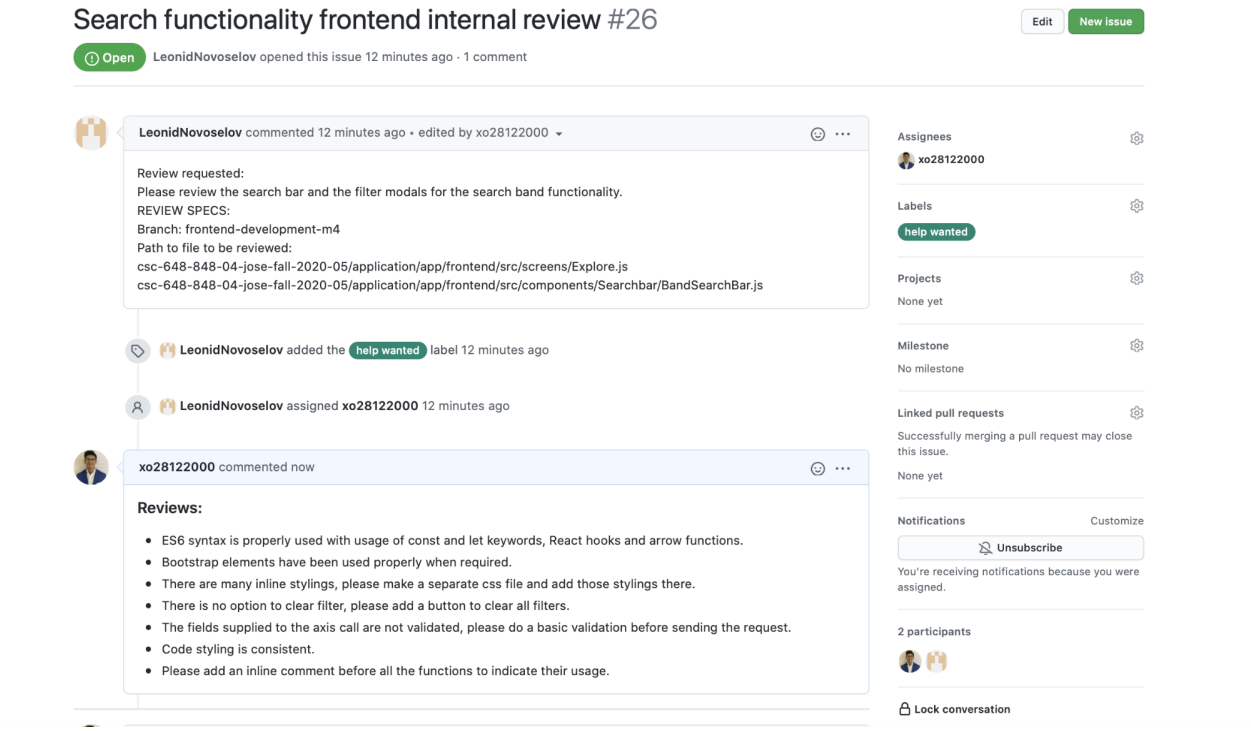
2. We will have our Search functionality reviewed.

   (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

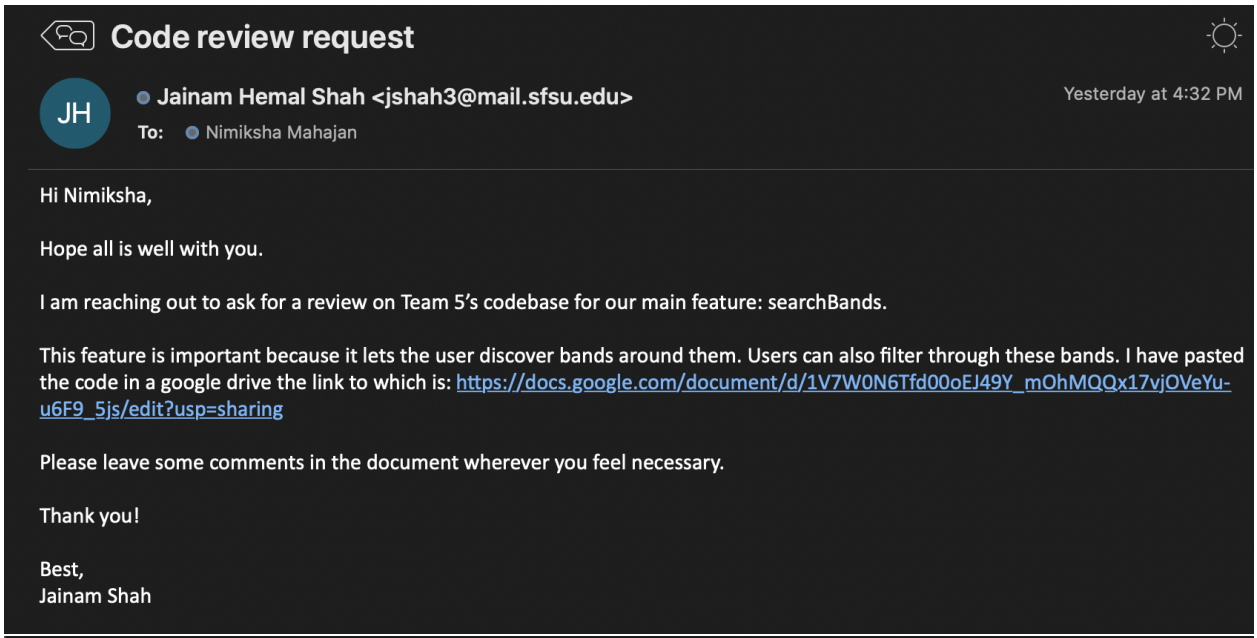
wsmarshall commented now

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.

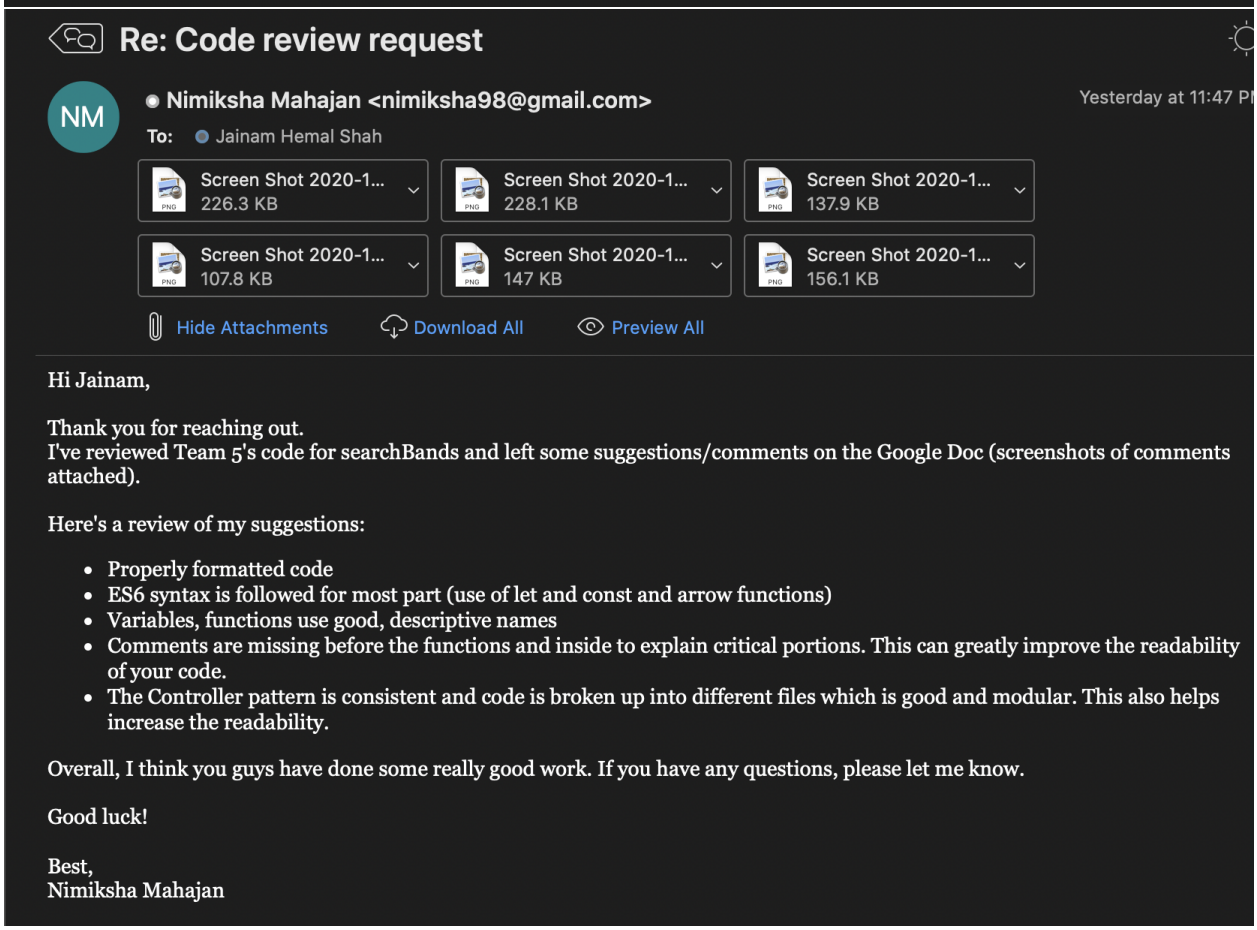
I am reaching out to ask for a review on Team 5's codebase for our main feature: searchBands.

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 });
   })
```

```
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
9:33 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.

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.

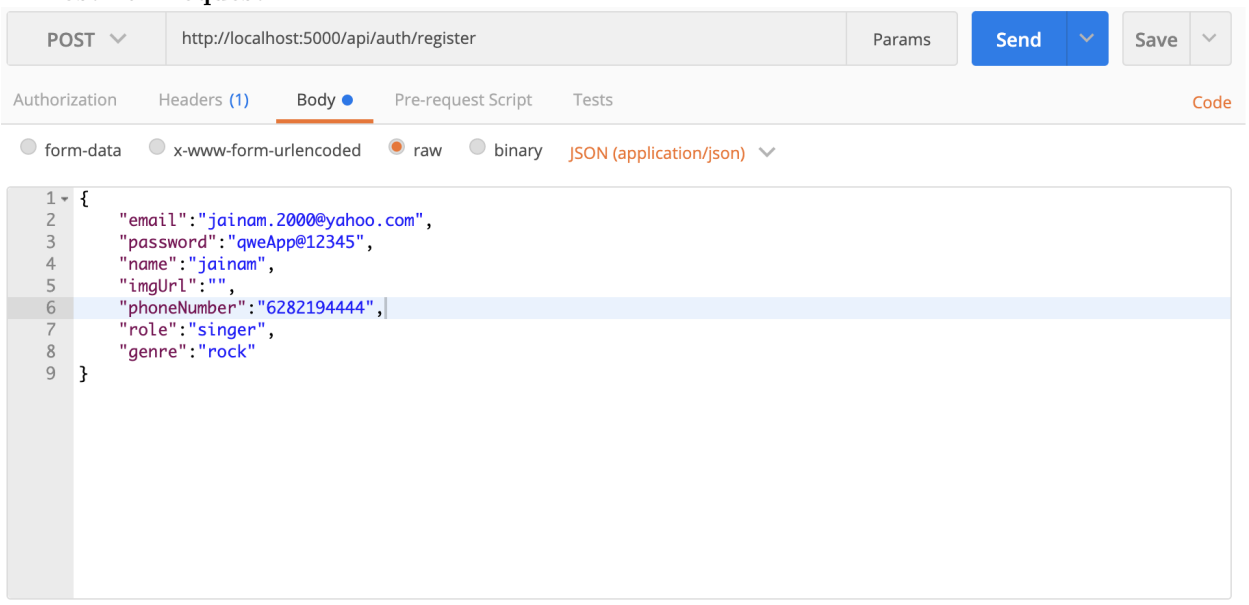
Register function:

```
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.

3 Product Screenshots

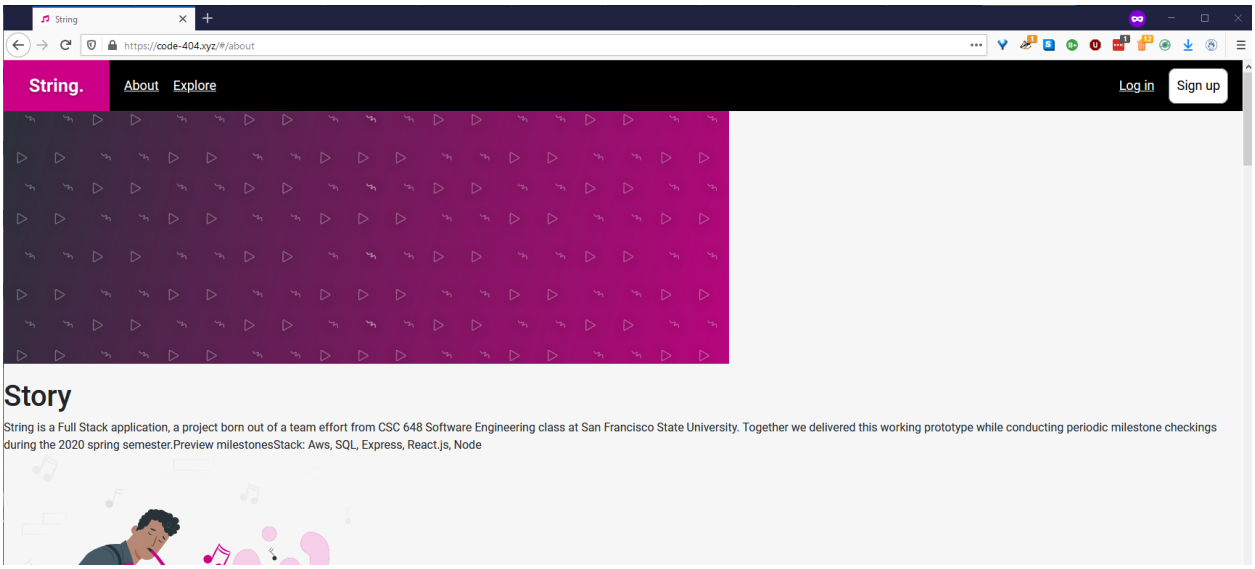


Figure 1: The About page

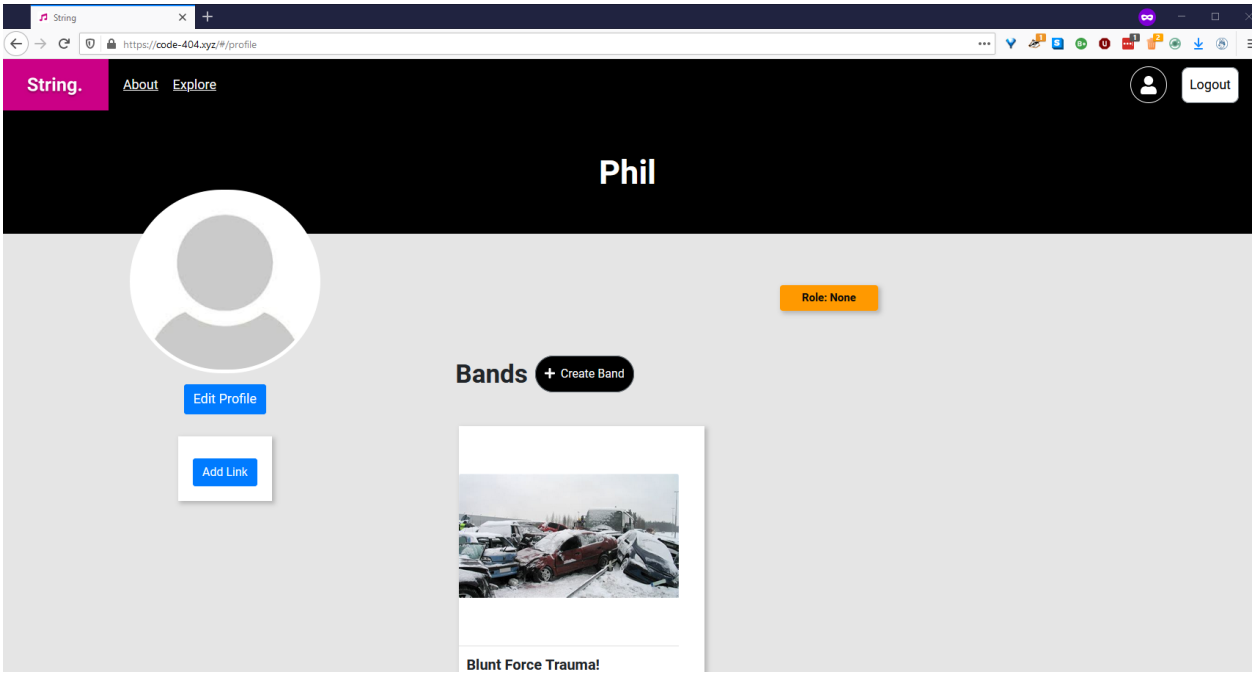


Figure 2: The User Profile page

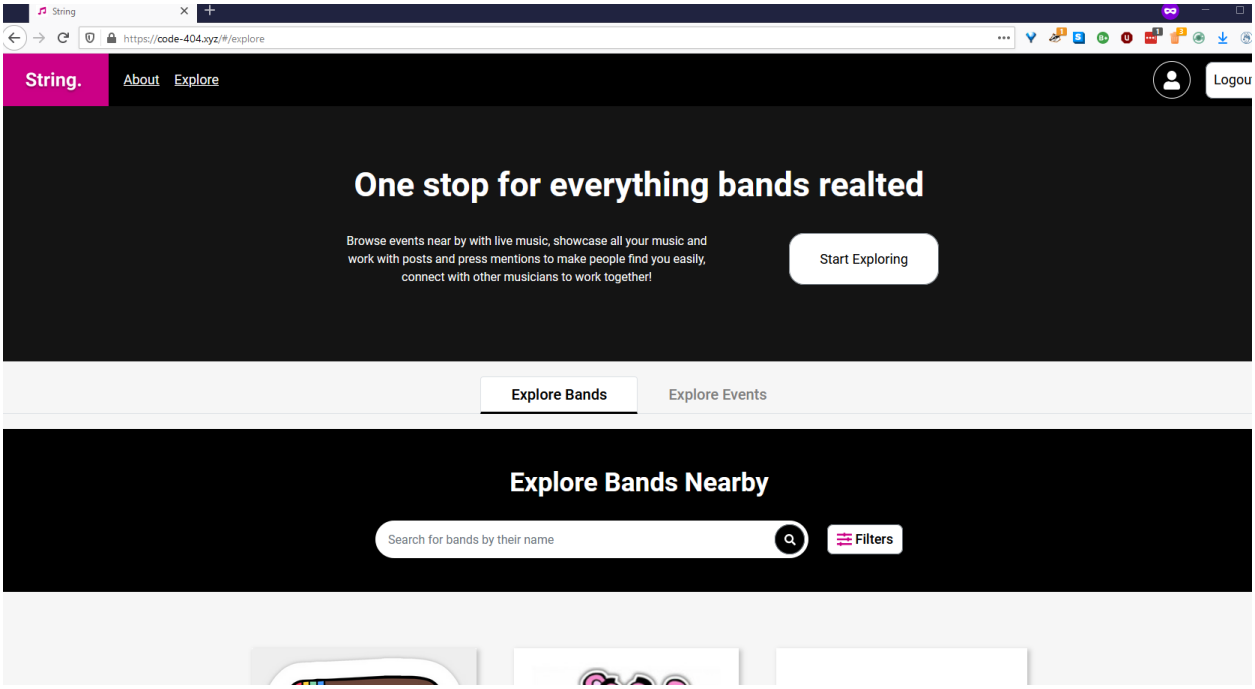


Figure 3: The Explore main section

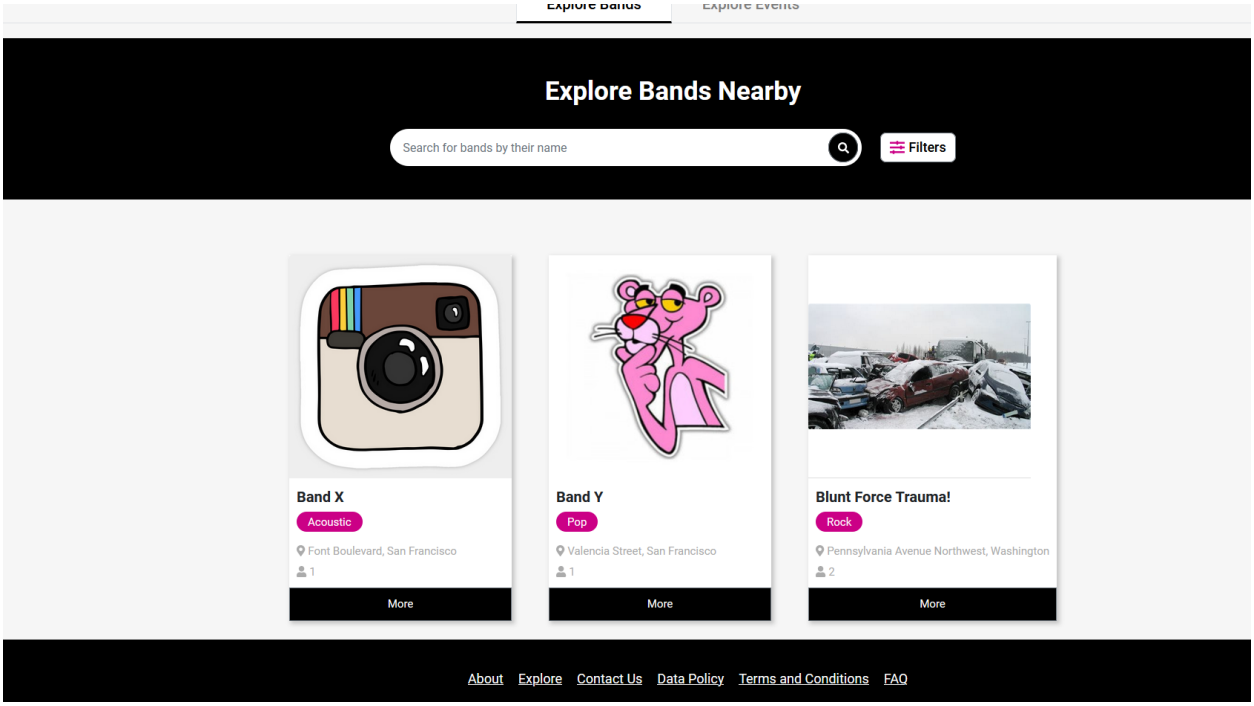


Figure 4: The Explore Bands section

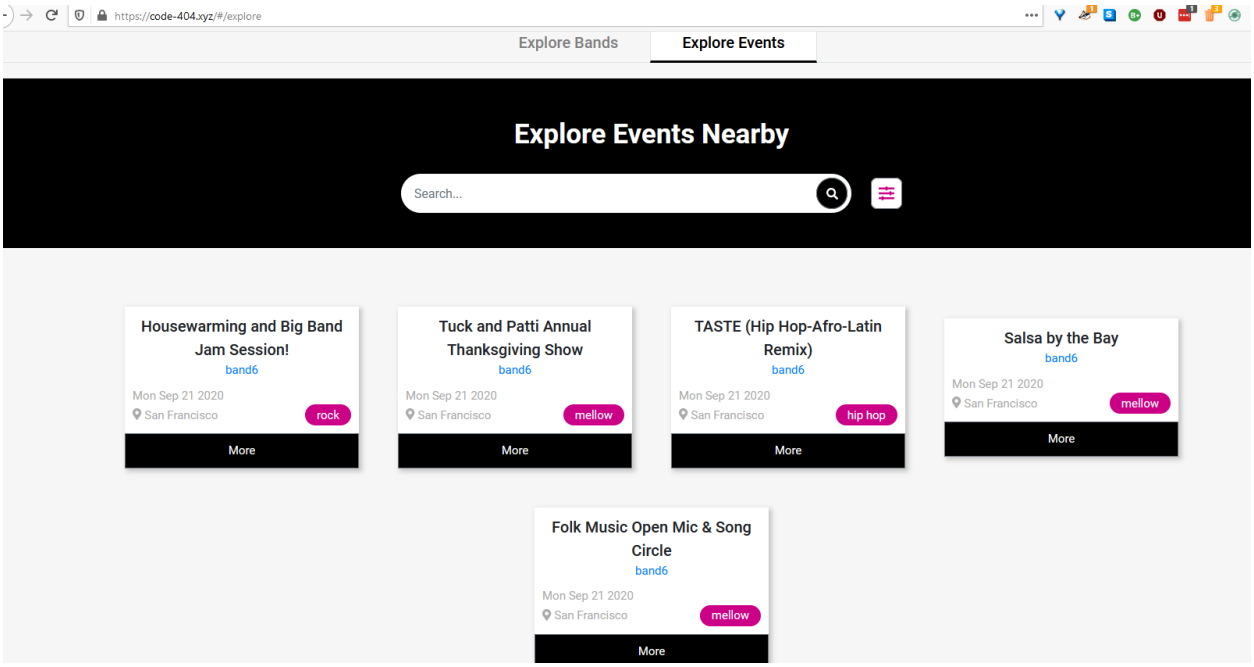


Figure 5: The Explore Events section

4 Key Database Table Screenshots

```
mysql> desc stringaccount
    -> ;
+----------------+---------------+------+-----+-----------------------------------------+----------------+
| Field          | Type          | Null | Key | Default                                 | Extra          |
+----------------+---------------+------+-----+-----------------------------------------+----------------+
| userId         | int           | NO   | PRI | NULL                                    | auto_increment |
| email          | varchar(350)  | NO   | UNI | NULL                                    |                |
| password       | varchar(200)  | NO   |     | NULL                                    |                |
| name           | varchar(100)  | NO   |     | NULL                                    |                |
| profileImageUrl | varchar(150) | YES  |     | notProvidedByUser                       |                |
| phoneNumber    | varchar(15)   | YES  |     | not specified                           |                |
| links          | varchar(500)  | YES  |     | NULL                                    |                |
| location       | varchar(500)  | YES  |     | 1600 Pennsylvania Ave NW, Washington DC 20500 |          |
| locationLat    | decimal(30,15) | YES |     | 38.897667500000000                      |                |
| locationLong   | decimal(30,15) | YES |     | -77.038767900000000                     |                |
| role           | varchar(50)   | YES  |     | not yet specified                       |                |
| genre          | varchar(50)   | YES  |     | not yet specified                       |                |
+----------------+---------------+------+-----+-----------------------------------------+----------------+
12 rows in set (0.01 sec)

mysql>
```

Figure 6: The Stringaccount table

```
mysql> desc band;
+--------------------+----------------+------+-----+---------+----------------+
| Field              | Type           | Null | Key | Default | Extra          |
+--------------------+----------------+------+-----+---------+----------------+
| bandId             | int            | NO   | PRI | NULL    | auto_increment |
| name               | varchar(100)   | NO   | UNI | NULL    |                |
| numMembers         | int            | YES  |     | 1       |                |
| logoImageUrl       | varchar(500)   | YES  |     | NULL    |                |
| links              | varchar(500)   | YES  |     | NULL    |                |
| description        | varchar(500)   | YES  |     | NULL    |                |
| location           | varchar(500)   | YES  |     | NULL    |                |
| locationLat        | decimal(30,15) | YES  |     | NULL    |                |
| locationLong       | decimal(30,15) | YES  |     | NULL    |                |
| genre              | varchar(100)   | YES  |     | NULL    |                |
| isLookingForMember | tinyint(1)     | YES  |     | NULL    |                |
+--------------------+----------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)

mysql>
```

Figure 7: The Band table

```
mysql> desc bandmembers;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| bandMemberId | int         | NO   | PRI | NULL    | auto_increment |
| isBandAdmin  | tinyint(1)  | YES  |     | NULL    |                |
| role         | varchar(45) | YES  |     | NULL    |                |
| dateJoined   | date        | YES  |     | NULL    |                |
| userId       | int         | YES  | MUL | NULL    |                |
| bandId       | int         | YES  | MUL | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

mysql>
```

Figure 8: The Bandmembers table

```
mysql> desc bandposts;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| bandPostId  | int          | NO   | PRI | NULL    | auto_increment |
| media       | varchar(400) | YES  |     | NULL    |                |
| title       | varchar(45)  | YES  |     | NULL    |                |
| description | varchar(400) | YES  |     | NULL    |                |
| bandId      | int          | YES  | MUL | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql>
```

Figure 9: The Bandposts table

```
mysql> desc events;
+--------------+----------------+------+-----+---------+----------------+
| Field        | Type           | Null | Key | Default | Extra          |
+--------------+----------------+------+-----+---------+----------------+
| eventId      | int            | NO   | PRI | NULL    | auto_increment |
| title        | varchar(45)    | YES  |     | NULL    |                |
| description  | varchar(400)   | YES  |     | NULL    |                |
| date         | date           | YES  |     | NULL    |                |
| startTime    | time           | YES  |     | NULL    |                |
| endTime      | time           | YES  |     | NULL    |                |
| location     | varchar(500)   | YES  |     | NULL    |                |
| locationLat  | decimal(30,15) | YES  |     | NULL    |                |
| locationLong | decimal(30,15) | YES  |     | NULL    |                |
| bandId       | int            | YES  | MUL | NULL    |                |
+--------------+----------------+------+-----+---------+----------------+
10 rows in set (0.00 sec)

mysql>
```

Figure 10: The Events table

```
mysql> desc invitations;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| inviteId   | int          | NO   | PRI | NULL    | auto_increment |
| message    | varchar(150) | YES  |     | NULL    |                |
| dateSent   | date         | YES  |     | NULL    |                |
| sentByBand | tinyint(1)   | YES  |     | NULL    |                |
| userId     | int          | YES  | MUL | NULL    |                |
| bandId     | int          | YES  | MUL | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

mysql>
```

Figure 11: The Invitations table

```
mysql> desc repertoire;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| repId    | int          | NO   | PRI | NULL    | auto_increment |
| songName | varchar(45)  | YES  |     | NULL    |                |
| runTime  | varchar(45)  | YES  |     | NULL    |                |
| genre    | varchar(45)  | YES  |     | NULL    |                |
| link     | varchar(150) | YES  |     | NULL    |                |
| bandId   | int          | YES  | MUL | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

mysql>
```

Figure 12: The Repertoire table

```
mysql> desc sets;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| setId    | int         | NO   | PRI | NULL    | auto_increment |
| songName | varchar(45) | YES  |     | NULL    |                |
| runTime  | varchar(45) | YES  |     | NULL    |                |
| eventId  | int         | YES  | MUL | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

Figure 13: The Sets table

5 Task Management System Screenshots

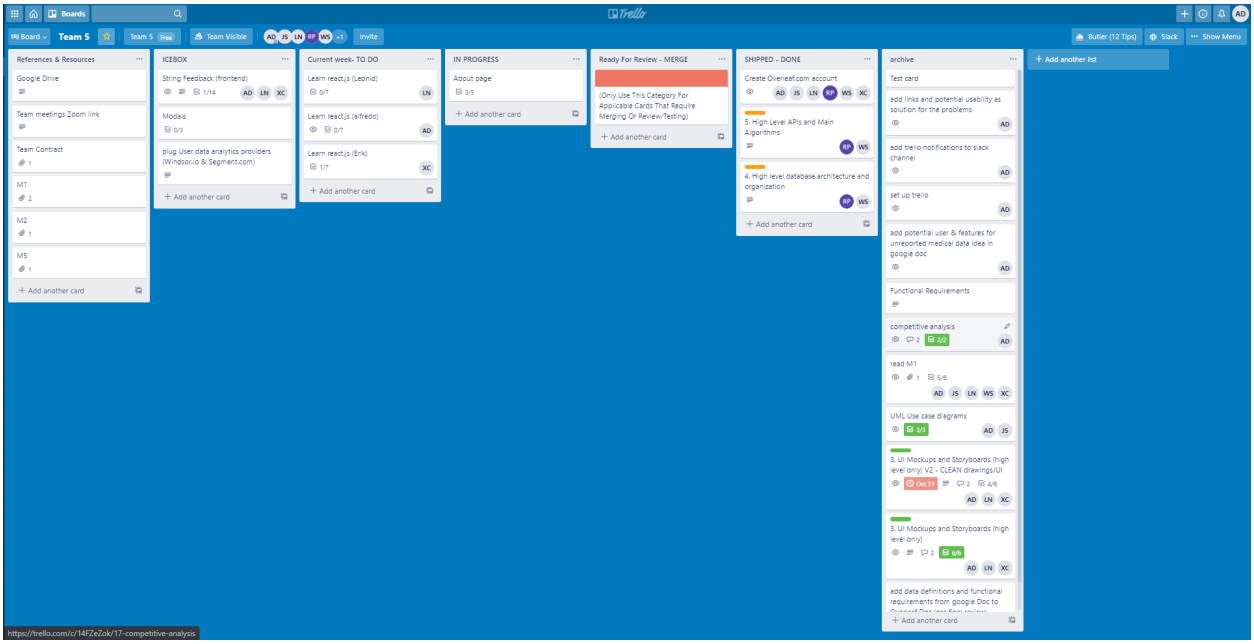


Figure 14: A screenshot of Trello, the task management system used

6 Team Member Contributions

6.1 Jainam Shah

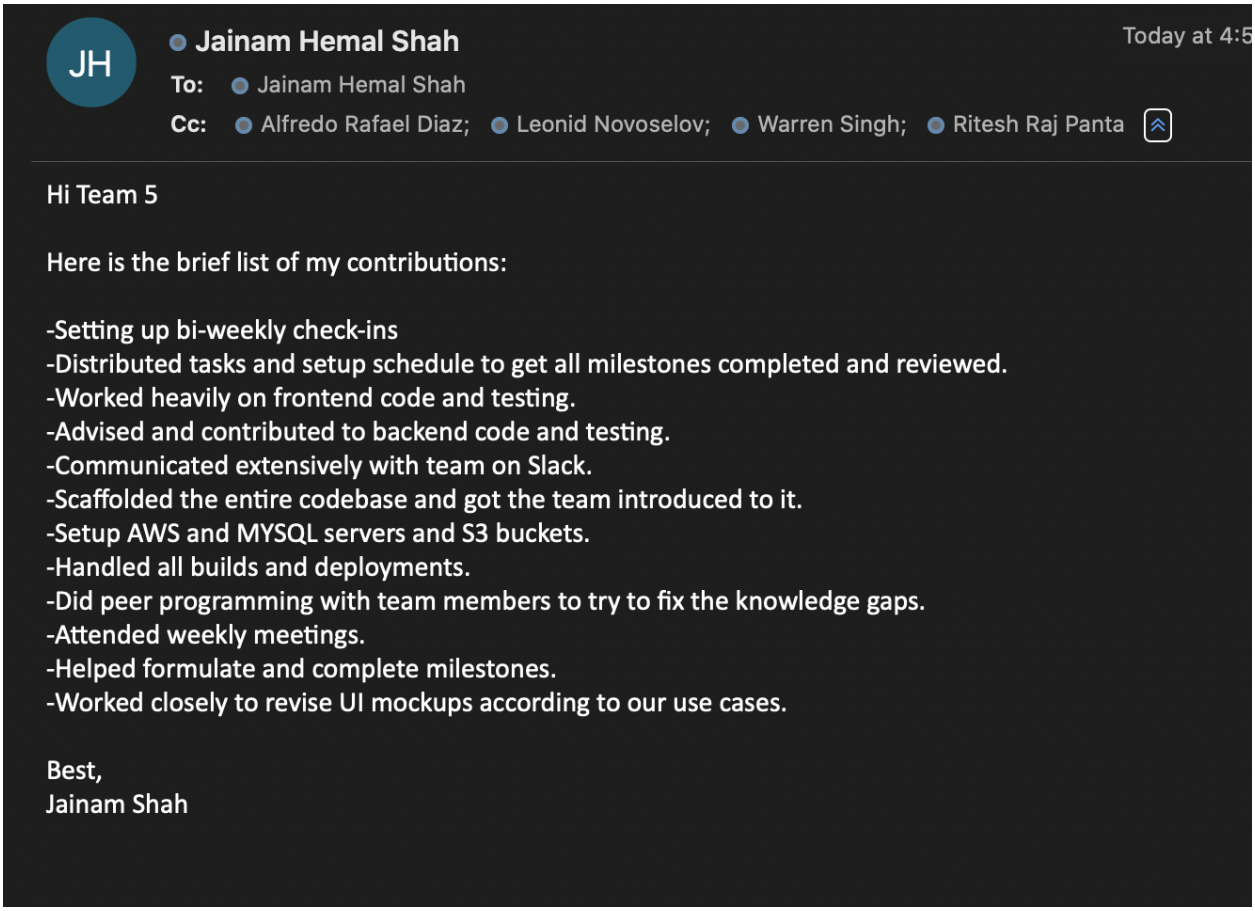
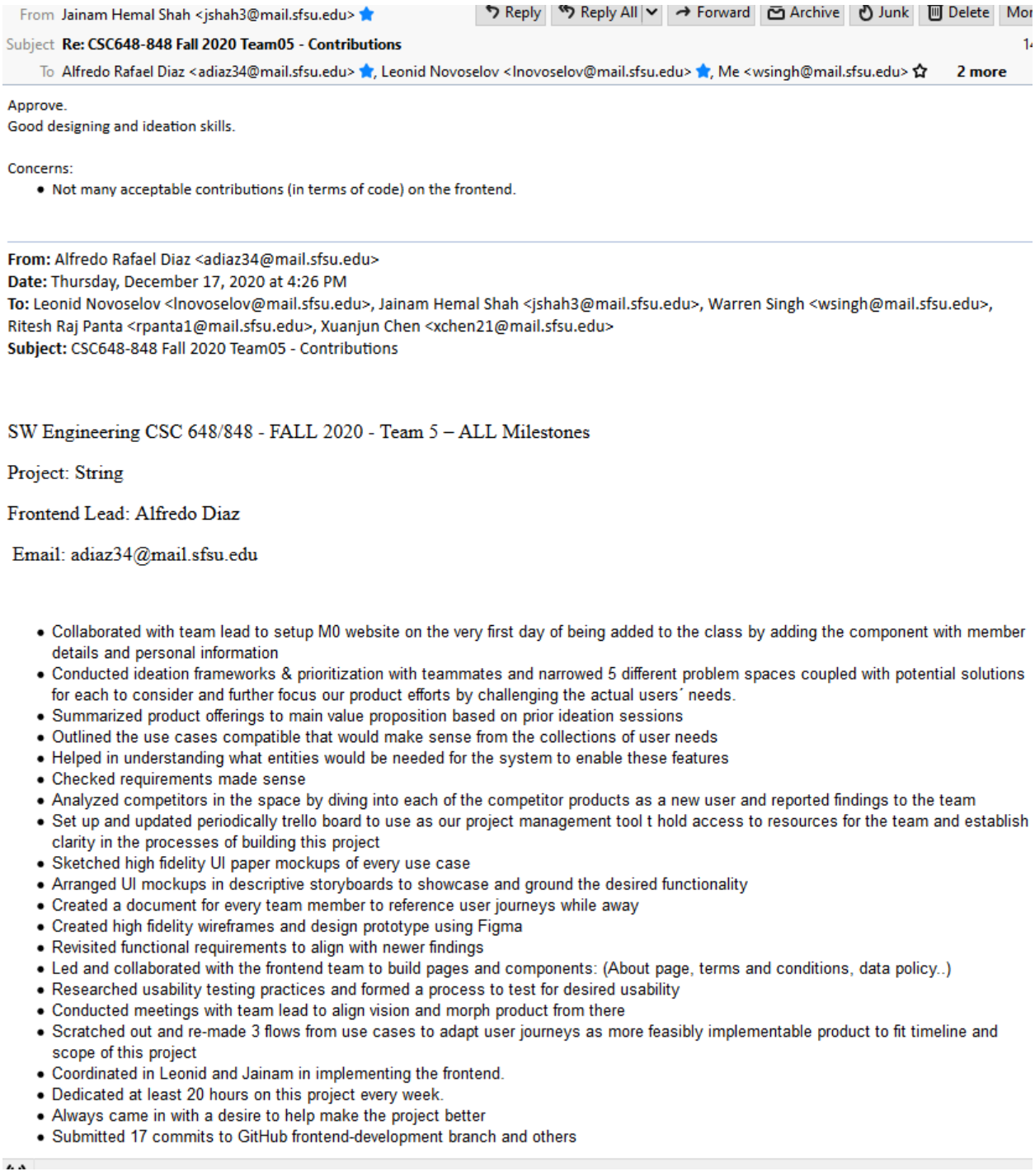


Figure 15: Jainam Shah's contributions - 135 commits

6.2 Alfredo Diaz



From Jainam Hemal Shah <jshah3@mail.sfsu.edu>
Reply | Reply All | Forward | Archive | Junk | Delete | Mor
Subject **Re: CSC648-848 Fall 2020 Team05 - Contributions**
To Alfredo Rafael Diaz <adiaz34@mail.sfsu.edu>, Leonid Novoselov <lnovoselov@mail.sfsu.edu>, Me <wsingh@mail.sfsu.edu> **2 more**

Approve.
Good designing and ideation skills.

Concerns:

- Not many acceptable contributions (in terms of code) on the frontend.

From: Alfredo Rafael Diaz <adiaz34@mail.sfsu.edu>
Date: Thursday, December 17, 2020 at 4:26 PM
To: Leonid Novoselov <lnovoselov@mail.sfsu.edu>, Jainam Hemal Shah <jshah3@mail.sfsu.edu>, Warren Singh <wsingh@mail.sfsu.edu>, Ritesh Raj Panta <rpanta1@mail.sfsu.edu>, Xuanjun Chen <xchen21@mail.sfsu.edu>
Subject: CSC648-848 Fall 2020 Team05 - Contributions

SW Engineering CSC 648/848 - FALL 2020 - Team 5 – ALL Milestones

Project: String

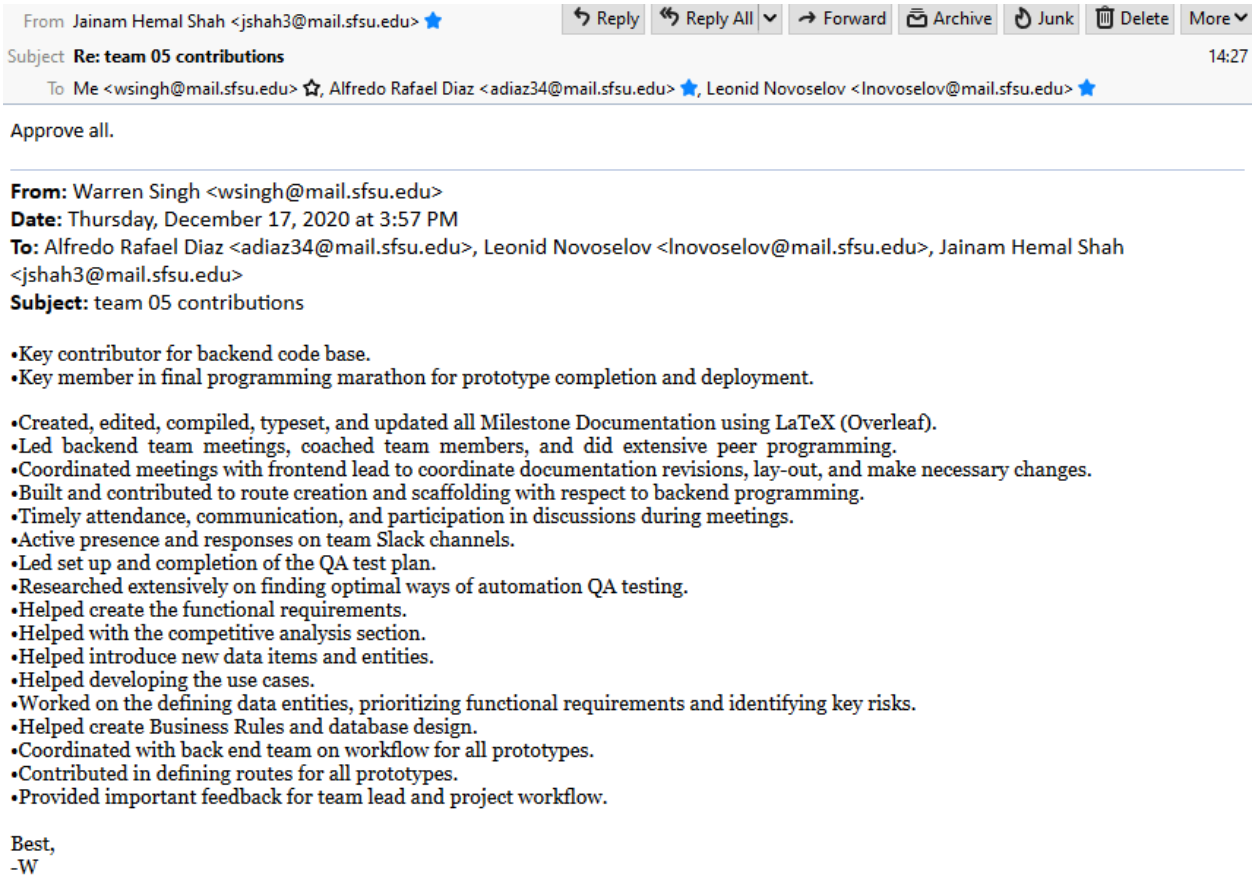
Frontend Lead: Alfredo Diaz

Email: adiaz34@mail.sfsu.edu

- Collaborated with team lead to setup M0 website on the very first day of being added to the class by adding the component with member details and personal information
- Conducted ideation frameworks & prioritization with teammates and narrowed 5 different problem spaces coupled with potential solutions for each to consider and further focus our product efforts by challenging the actual users´ needs.
- Summarized product offerings to main value proposition based on prior ideation sessions
- Outlined the use cases compatible that would make sense from the collections of user needs
- Helped in understanding what entities would be needed for the system to enable these features
- Checked requirements made sense
- Analyzed competitors in the space by diving into each of the competitor products as a new user and reported findings to the team
- Set up and updated periodically trello board to use as our project management tool t hold access to resources for the team and establish clarity in the processes of building this project
- Sketched high fidelity UI paper mockups of every use case
- Arranged UI mockups in descriptive storyboards to showcase and ground the desired functionality
- Created a document for every team member to reference user journeys while away
- Created high fidelity wireframes and design prototype using Figma
- Revisited functional requirements to align with newer findings
- Led and collaborated with the frontend team to build pages and components: (About page, terms and conditions, data policy..)
- Researched usability testing practices and formed a process to test for desired usability
- Conducted meetings with team lead to align vision and morph product from there
- Scratched out and re-made 3 flows from use cases to adapt user journeys as more feasibly implementable product to fit timeline and scope of this project
- Coordinated in Leonid and Jainam in implementing the frontend.
- Dedicated at least 20 hours on this project every week.
- Always came in with a desire to help make the project better
- Submitted 17 commits to GitHub frontend-development branch and others

Figure 16: Alfredo Diaz's contributions - 16 commits

6.3 Warren Singh



From Jainam Hemal Shah <jshah3@mail.sfsu.edu>
Reply | Reply All | Forward | Archive | Junk | Delete | More
Subject **Re: team 05 contributions** 14:27
To Me <wsingh@mail.sfsu.edu>, Alfredo Rafael Diaz <adiaz34@mail.sfsu.edu>, Leonid Novoselov <lnovoselov@mail.sfsu.edu>

Approve all.

From: Warren Singh <wsingh@mail.sfsu.edu>
Date: Thursday, December 17, 2020 at 3:57 PM
To: Alfredo Rafael Diaz <adiaz34@mail.sfsu.edu>, Leonid Novoselov <lnovoselov@mail.sfsu.edu>, Jainam Hemal Shah <jshah3@mail.sfsu.edu>
Subject: team 05 contributions

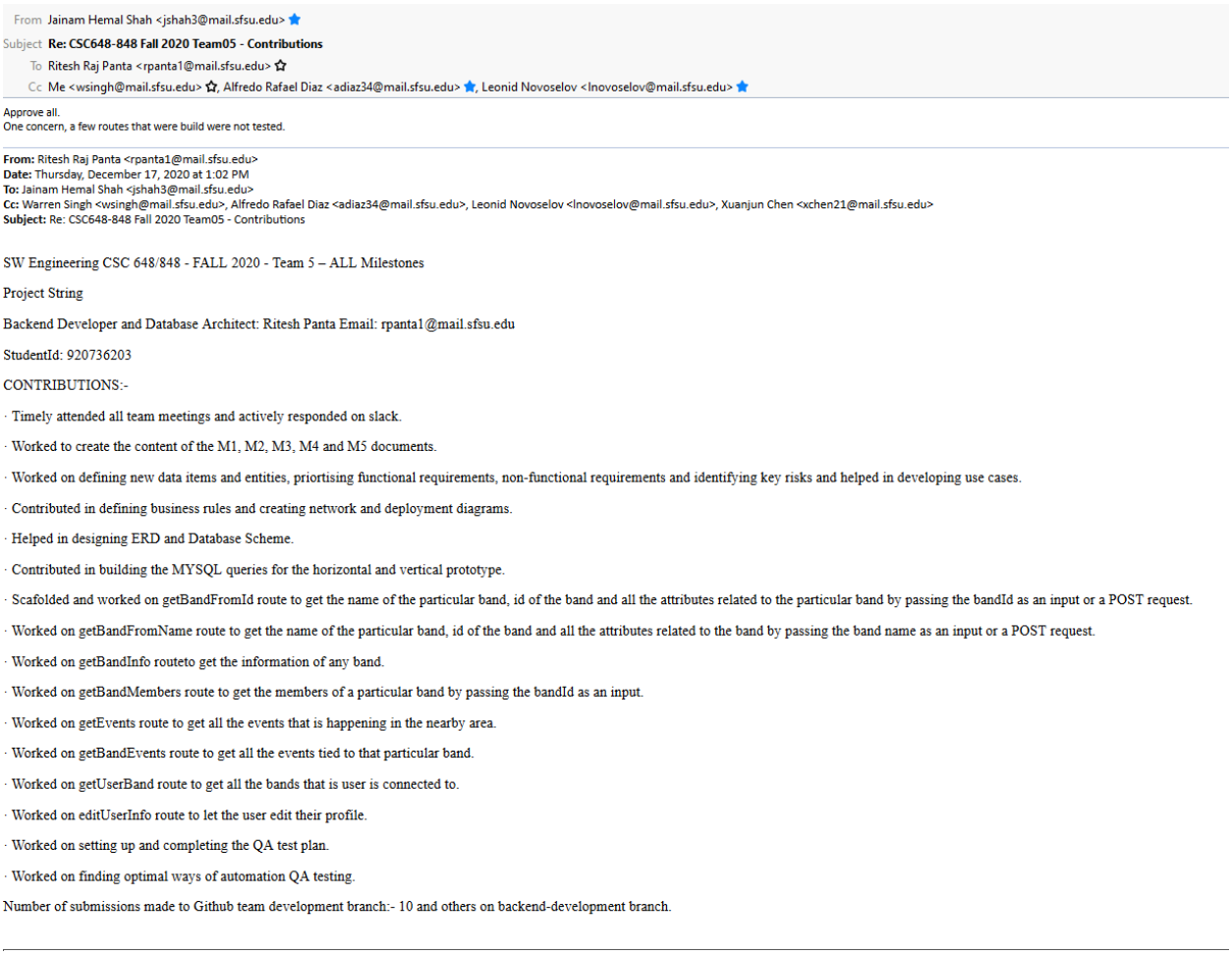
•Key contributor for backend code base.
•Key member in final programming marathon for prototype completion and deployment.

•Created, edited, compiled, typeset, and updated all Milestone Documentation using LaTeX (Overleaf).
•Led backend team meetings, coached team members, and did extensive peer programming.
•Coordinated meetings with frontend lead to coordinate documentation revisions, lay-out, and make necessary changes.
•Built and contributed to route creation and scaffolding with respect to backend programming.
•Timely attendance, communication, and participation in discussions during meetings.
•Active presence and responses on team Slack channels.
•Led set up and completion of the QA test plan.
•Researched extensively on finding optimal ways of automation QA testing.
•Helped create the functional requirements.
•Helped with the competitive analysis section.
•Helped introduce new data items and entities.
•Helped developing the use cases.
•Worked on the defining data entities, prioritizing functional requirements and identifying key risks.
•Helped create Business Rules and database design.
•Coordinated with back end team on workflow for all prototypes.
•Contributed in defining routes for all prototypes.
•Provided important feedback for team lead and project workflow.

Best,
-W

Figure 17: Warren Singh's contributions - 73 commits

6.4 Ritesh Panta



From Jainam Hemal Shah <jshah3@mail.sfsu.edu>
Subject **Re: CSC648-848 Fall 2020 Team05 - Contributions**
To Ritesh Raj Panta <rpanta1@mail.sfsu.edu>
Cc Me <wsingh@mail.sfsu.edu>, Alfredo Rafael Diaz <adiaz34@mail.sfsu.edu>, Leonid Novoselov <lnovoselov@mail.sfsu.edu>

Approve all.
One concern, a few routes that were build were not tested.

From: Ritesh Raj Panta <rpanta1@mail.sfsu.edu>
Date: Thursday, December 17, 2020 at 1:02 PM
To: Jainam Hemal Shah <jshah3@mail.sfsu.edu>
Cc: Warren Singh <wsingh@mail.sfsu.edu>, Alfredo Rafael Diaz <adiaz34@mail.sfsu.edu>, Leonid Novoselov <lnovoselov@mail.sfsu.edu>, Xuanjun Chen <xchen21@mail.sfsu.edu>
Subject: Re: CSC648-848 Fall 2020 Team05 - Contributions

SW Engineering CSC 648/848 - FALL 2020 - Team 5 – ALL Milestones

Project String

Backend Developer and Database Architect: Ritesh Panta Email: rpanta1@mail.sfsu.edu

StudentId: 920736203

CONTRIBUTIONS:-

· Timely attended all team meetings and actively responded on slack.

· Worked to create the content of the M1, M2, M3, M4 and M5 documents.

· Worked on defining new data items and entities, priortising functional requirements, non-functional requirements and identifying key risks and helped in developing use cases.

· Contributed in defining business rules and creating network and deployment diagrams.

· Helped in designing ERD and Database Scheme.

· Contributed in building the MYSQL queries for the horizontal and vertical prototype.

· Scafolded and worked on getBandFromId route to get the name of the particular band, id of the band and all the attributes related to the particular band by passing the bandId as an input or a POST request.

· Worked on getBandFromName route to get the name of the particular band, id of the band and all the attributes related to the band by passing the band name as an input or a POST request.

· Worked on getBandInfo routeto get the information of any band.

· Worked on getBandMembers route to get the members of a particular band by passing the bandId as an input.

· Worked on getEvents route to get all the events that is happening in the nearby area.

· Worked on getBandEvents route to get all the events tied to that particular band.

· Worked on getUserBand route to get all the bands that is user is connected to.

· Worked on editUserInfo route to let the user edit their profile.

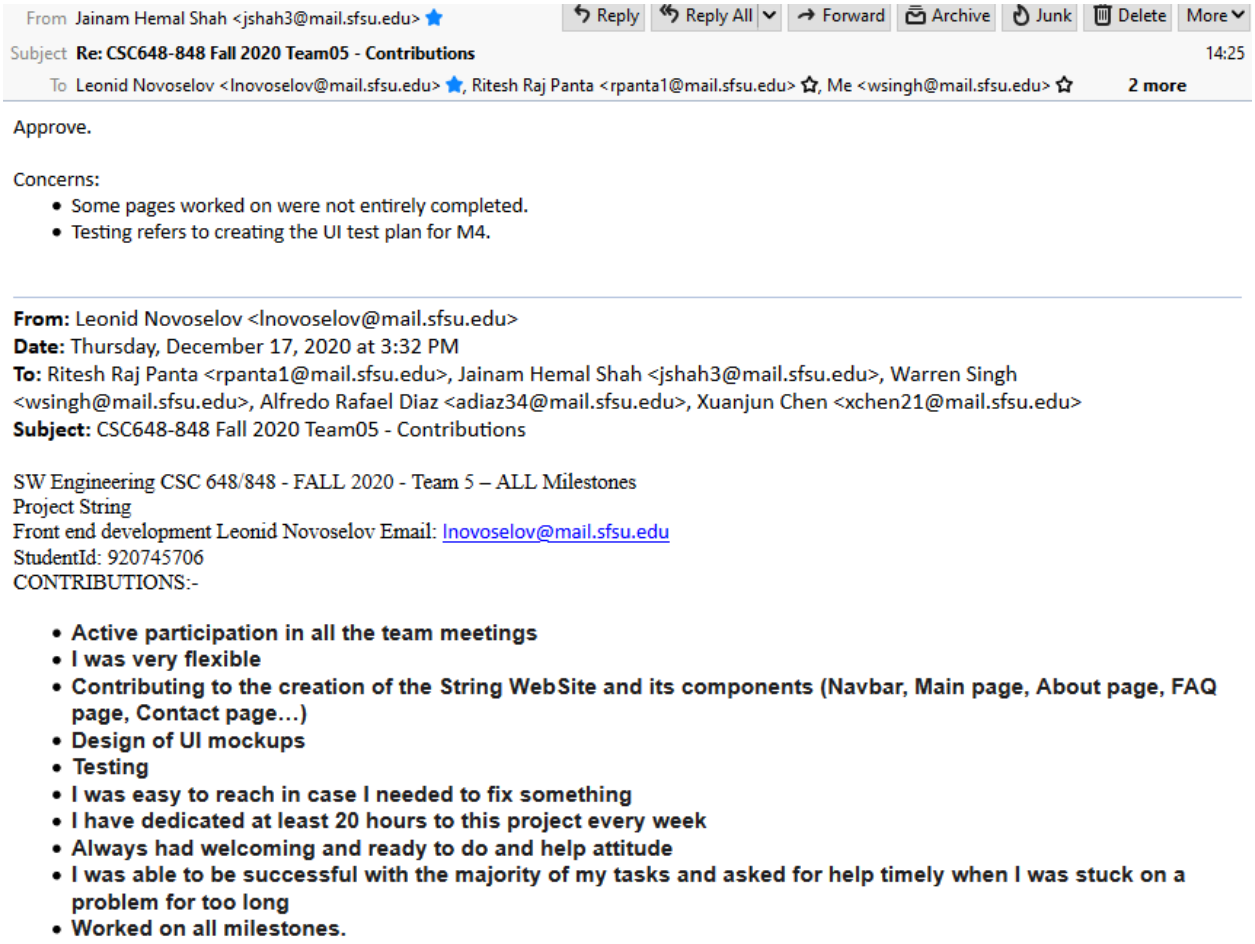
· Worked on setting up and completing the QA test plan.

· Worked on finding optimal ways of automation QA testing.

Number of submissions made to Github team development branch:- 10 and others on backend-development branch.

Figure 18: Ritesh Panta's contributions - 16 commits

6.5 Leonid Novoselov



From Jainam Hemal Shah <jshah3@mail.sfsu.edu>
Reply | Reply All | Forward | Archive | Junk | Delete | More
Subject **Re: CSC648-848 Fall 2020 Team05 - Contributions** 14:25
To Leonid Novoselov <lnovoselov@mail.sfsu.edu>, Ritesh Raj Panta <rpanta1@mail.sfsu.edu>, Me <wsingh@mail.sfsu.edu> **2 more**

Approve.

Concerns:

- Some pages worked on were not entirely completed.
- Testing refers to creating the UI test plan for M4.

From: Leonid Novoselov <lnovoselov@mail.sfsu.edu>
Date: Thursday, December 17, 2020 at 3:32 PM
To: Ritesh Raj Panta <rpanta1@mail.sfsu.edu>, Jainam Hemal Shah <jshah3@mail.sfsu.edu>, Warren Singh <wsingh@mail.sfsu.edu>, Alfredo Rafael Diaz <adiaz34@mail.sfsu.edu>, Xuanjun Chen <xchen21@mail.sfsu.edu>
Subject: CSC648-848 Fall 2020 Team05 - Contributions

SW Engineering CSC 648/848 - FALL 2020 - Team 5 – ALL Milestones
Project String
Front end development Leonid Novoselov Email: lnovoselov@mail.sfsu.edu
StudentId: 920745706
CONTRIBUTIONS:-

- **Active participation in all the team meetings**
- **I was very flexible**
- **Contributing to the creation of the String WebSite and its components (Navbar, Main page, About page, FAQ page, Contact page…)**
- **Design of UI mockups**
- **Testing**
- **I was easy to reach in case I needed to fix something**
- **I have dedicated at least 20 hours to this project every week**
- **Always had welcoming and ready to do and help attitude**
- **I was able to be successful with the majority of my tasks and asked for help timely when I was stuck on a problem for too long**
- **Worked on all milestones.**

Figure 19: Leonid Novoselov's contributions - 13 commits

6.6 Xuanjun (Eric) Chen

7 Post-Project Analysis: Lessons Learned

One challenge was the sometime lapse in timely communication via Slack. Sometimes, team members would not reply or respond to messages, causing confusion from leads regarding if team members saw messages, or which tasks were being completed by whom.

The only fix for this in the future would be to more firmly establish Slack communication protocols: To mark seen messages, and to reply within a certain time frame (say, 12 hours). This is more of a cultural norm, so it is a bit harder to nail down, but certainly some attention and definite guidelines could be laid down from the very beginning.

Such guidelines could be along the lines of:

1. If you see a message, mark it as read
2. Reply to messages within 12 hours, even if only to say that you are working on it
3. Be clear as to how far along with a task you are, using percentages

And so on.

The main challenge was the lack of overall technical knowledge on the team, which contributed to picking up of slack by more experienced/knowledgable team members.

Next time, I would implement earlier check-ins regarding technical knowledge: while it was good to provide time to the team to learn the necessary tech stack, beginners frequently cannot estimate their own proficiency of knowledge and will tend to overestimate their ability, or get stuck very early on. Therefore, while of course newcomers to particular frameworks or tech stacks should be given time to assimilate the new knowledge, there must be early and frequent checkpoints for them to demonstrate that knowledge in a practical manner.

For example, if a team member had to learn react, one easy way to implement this would be to assign them a specific youtube web tutorial (the team lead has skipped through and knows what the end product should look like and do), and then ask them to screenshare the resulting react project after they had gone through and built the project along with the video, and discuss with team lead in a 1:1.

For another example, if a team member had to learn how the backend worked, one way to approach this would be to send them some tutorial videos/articles (or ask them to research on their own), and then ask them in a 1:1 meeting to draw a diagram and explain succinctly how the different parts of the backend communicated and worked with each other.

For this project in particular, assigning the team a very early (week 3?) baby prototype could have been used as a quick 'smoke test' for the team's technical ability. Such a baby prototype could have consisted of a

single webpage which had 1 input form and retrieved data from a backend database. Basically, the simplest possible prototype that would have tested key knowledge before critical moments.

Such smoke tests do not just check for technical ability – they help build it before mission critical workloads become too heavy. Judicious use of check ins and coaching can build up team member's technical ability and allow them to surpass previous limits in technical knowledge and coding skill.

Since this was a class and not an industry environment, it was perhaps inevitable that some team members would be in a position to contribute more as far as technical ability is concerned. Thus, it is best to find this out early rather than having key tasks dropped at crucial moments due to lack of technical knowledge or skill.