BrewsMuse
Project Overview

The Iron Yard Students

Andrea Coaker
Front End Engineering

Braden Holmes
Front End Engineering

Parker Tankersley
Back End Engineering

Colby Burke
Back End Engineering

Name of the App
BrewsMuse

Type of App
Web

Description

This is a mobile first web application that allows users in Charleston, SC to locate breweries, brew pubs, etc. that offers a large selection of craft beers and also feature live music. BrewsMuse is for the user that wants to kick back and relax with a good beer while listening to live, local music.

Vendor users can create accounts for their brew establishment and provide users with a list of what's currently on tap as well as a list of upcoming music events.

The general user will be able to search for an establishment by keyword or via map view. Search results will render a short listing view and a detailed individual vendor profile view. Users will also be able to check-in, like, and write a review and the information will then render to the vendors profile page.

Technologies:

- HTML
- CSS/SCSS
    - Bootstrap
- Javascript:
    - React
    - Backbone
- C#:
    - .NET Framework
    - SQLite
- API's
    - Google Maps

Features (MVP):

1. Customer Map View
2. Customer List View
3. Customer can see a vendor's profile view
4. Customer can see list of vendor's beers
5. Customer can see list of vendor's upcoming live music
6. Customer account creation
7. Vendor Account Creation
8. Vendor can edit their profile view

Roadmap (future requirements):

1. Customer can write a review
2. Customer can like a vendor
3. Customer can check-in

User Stories

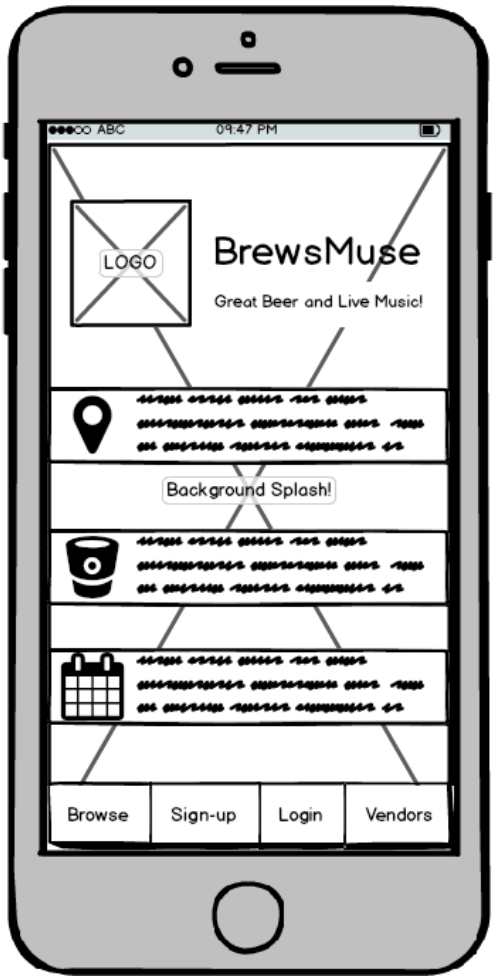
User Story: *As an anonymous* user, *I want* a welcome view *so that* I know what the site does and can navigate to other views.

Assumptions: Frontend to create welcome page, site summary and nav bar.

Functionality:

1. User will gain a general understanding about the site
2. Customer user will be able to navigate to the full vendor list/map view, create an account or login.
3. Vendor user will be able to navigate to the full vendor list/map view, create an account or login.

Wireframe:



User Story: *As an anonymous* customer, *I want to* see a list view and map view of nearby bars *so that* I know which bars have what beers and what musical acts.

Assumptions: Google Maps API provides map information. Frontend to use JSON request to fetch API data. Backend will use a POST request to store map data.

Functionality (by priority):

1. User will be able to see vendor's pin location on the map.
2. User will be able to click on pin location to view additional info on: vendor name, vendor address, link to vendor profile view, link to vendor website
3. User will be able to see list of all vendors
4. User will be able to filter by: type of bar, live music tonight, type of beers on tap

Wireframe:

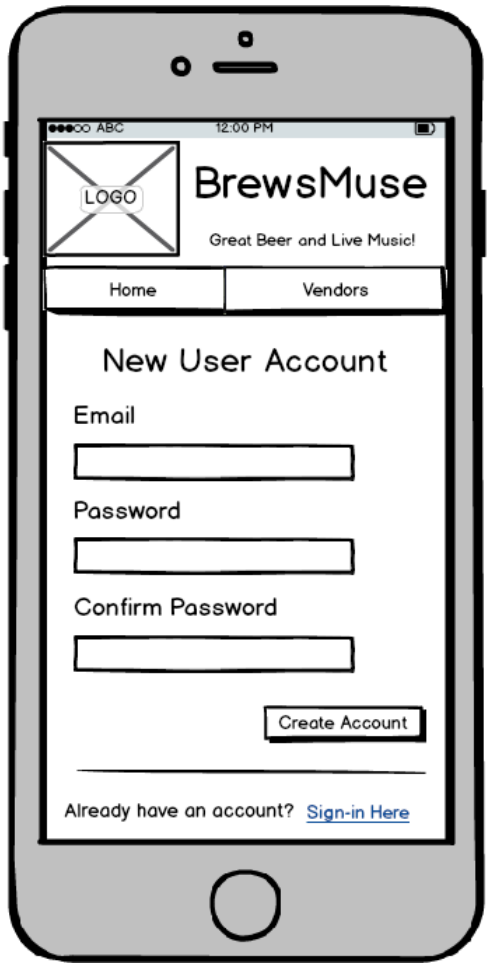
User Story: *As an anonymous* customer *I want to* have a registration page and a login page, *so that I can* register an account and login to my account.

Assumptions: The customer does not have an account set up. Backend has authentication that accepts POST request to save user data. Frontend has form view to capture user inputs.

Functionality:

1. Anonymous customer will be able to see registration view
2. Anonymous customer will be able to fill out name, email, and password fields
3. Customer will become authenticated
4. Authenticated customer will be able to see login view
5. Authenticated customer will be able to fill out email, and password fields
6. Customer will login

Wireframes:



User Story: *As a* vendor *I want to* have an account registration page and account login page, so *that I can* register an account and create my company profile page, then be able to login to that page.

Assumptions: The vendor does not have an account set up. Backend has authentication that accepts POST request to save user data. Frontend has form view to capture user inputs.

Functionality:

1. Vendor will be able to see registration view
2. Vendor will be able to fill out name, email, password, and location fields
3. Vendor will register and become authenticated
4. Vendor will be able to see login view
5. Authenticated vendor will be able to fill out email, and password fields
6. Vendor will login

Login Wireframe - See Registration Wireframe on Page 8

User Story: *As an* authenticated vendor *I want to* have a vendor account view, s*o that I can* view and update my profile information.

Assumptions: The vendor has an account set up. Backend has authentication that accepts POST request to save vendor data. Frontend has form view to capture user inputs.

Functionality:

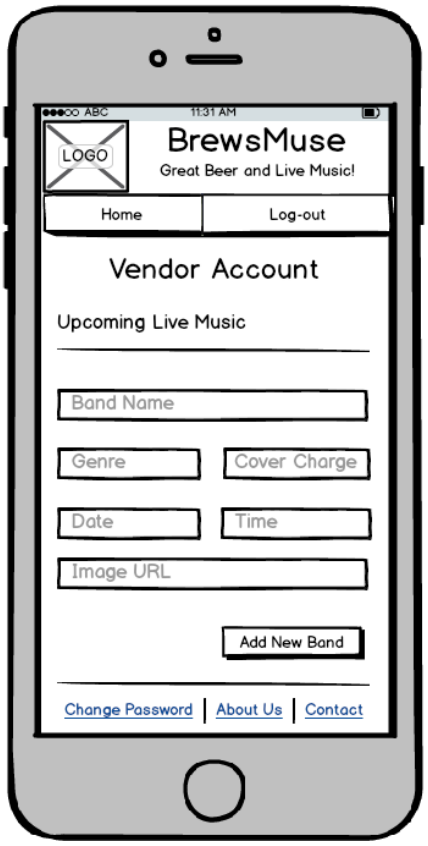
1. Vendor will be able to see profile view
2. Vendor will be able to edit name, email, password, location, bar name, location beer type, music schedule, and hours

Wireframes - See Next Page


ABC
11:31 AM
LOGO
BrewsMuse
Great Beer and Live Music!
Home
Log-out
Create New Account
First Name
Last Name
Company
Hours
Email
Phone
Address
City
ST
ZIP
Password
Website http://
Password
Create Account
Already have an account?
Sign-in Here


ABC
11:31 AM
LOGO
BrewsMuse
Great Beer and Live Music!
Home
Log-out
Update Account
First Name
Last Name
Company
Hours
Email
Phone
Address
City
ST
ZIP
Website http://
Update Account
Change Password
About Us
Contact


ABC
11:31 AM
LOGO
BrewsMuse
Great Beer and Live Music!
Home
Log-out
Vendor Account
Beers on Tap
Beer Name
Brewery
Type
ABV%
Image URL
Add New Beer
Change Password
About Us
Contact


ABC
11:31 AM
LOGO
BrewsMuse
Great Beer and Live Music!
Home
Log-out
Vendor Account
Upcoming Live Music
Band Name
Genre
Cover Charge
Date
Time
Image URL
Add New Band
Change Password
About Us
Contact

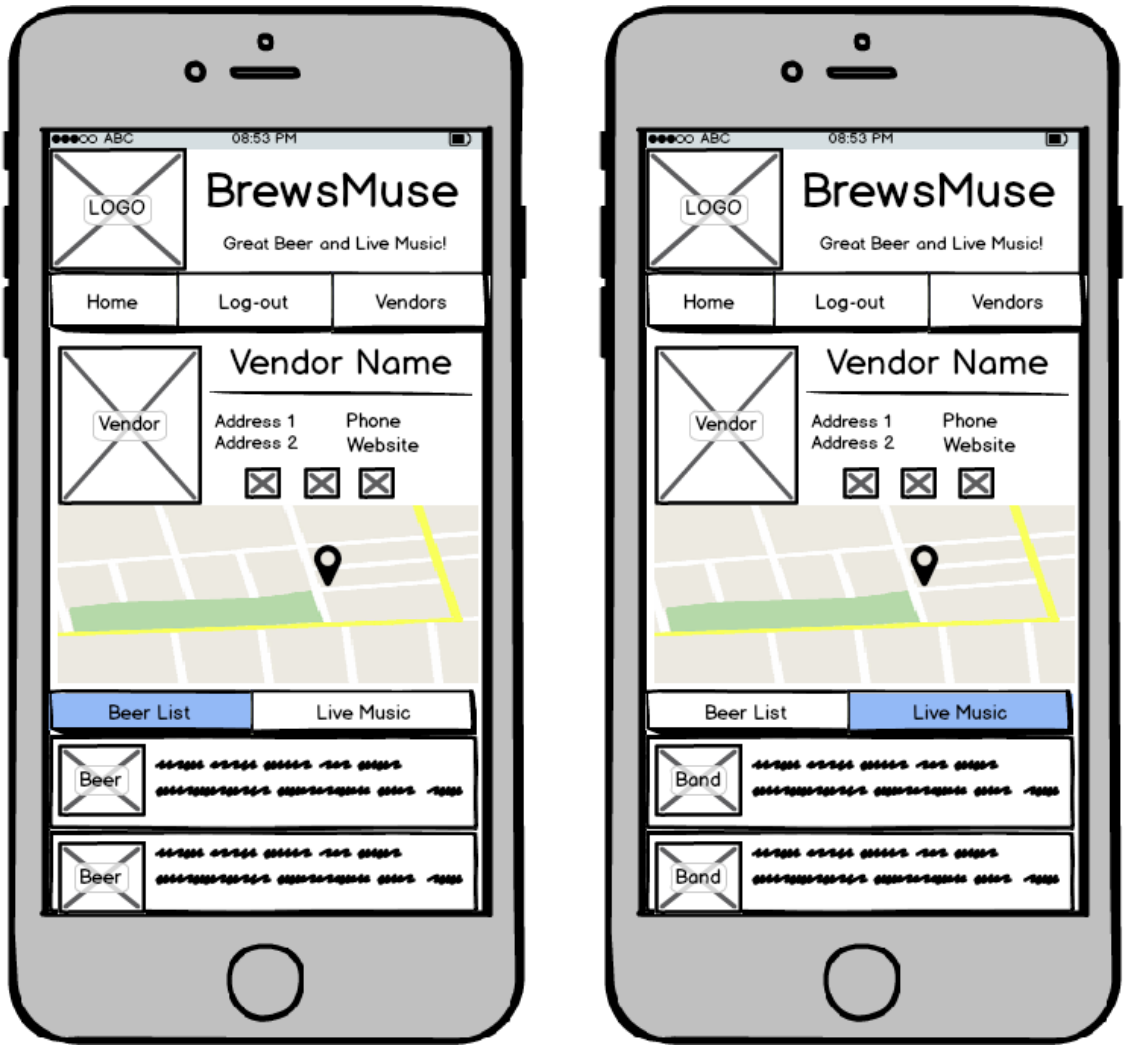
User Story: *As an* authenticated customer, *I want to* be able to click on a vendor *so that I can* see its full profile view.

Assumptions: Vendor profile has been created and single vendor record can be fetched. Frontend has created a view that can display the vendor's data. Backend has to handle a GET request to query the database of the vendor's profile.

Functionality:

1. Authenticated customer will be able to see profile view
2. Authenticated customer will be able to see vendor data such as location and beer selection, music schedule and hours

Wireframe:



User Story: *As an authenticated customer*, *I want to* be able to click on a vendor's beer tab OR live music tab *so that I can* see a list of beers and musical acts.

Assumptions: Frontend will create a profile view with tabbed sections showing a vendor's beer list and live music list. Vender has the ability to PUT information to profile.

Functionality:

1. User will be able to see beer list and scroll through upcoming live shows.
2. User will be able to see live music list and scroll through upcoming list of bands.

Wireframe:

Phase 2 User Stories

User Story: *As an* authenticated customer, *I want to* be able to like a vendor *so that I can* know which establishments are best.

Assumptions: There is a customer account in which they vote or like a vendor. Frontend has created a like button that will send POST request. Backend will handle POST requests and GET information for likes per bar.

Functionality:

1. Authenticated customer will be able to push like button
2. Authenticated customer's like will be pushed to database for each vendor

User Story: *As an* authenticated customer, *I want to* be able to write a review of a vendor *so that other users* can see my opinion on that vendor.

Assumptions: There is a customer account in which they can write a review of a vendor. Frontend has created a review form and reviews will render to the vendor's profile page. Backend must send a PUT request to display user reviews.

Functionality:

1. Authenticated customer will be able to enter text and submit review from the vendor's profile page
2. Authenticated customer's text will be saved to database and render to vendor profile page

User Story: *As an* authenticated customer, *I want to* be able to check-in at a vendor *so that other users* know how many people are at that establishment that night.

Assumptions: There is a customer account in which they check-in at a vendor. Frontend has created a check-in button and will render a count of the number of check-ins. Backend POST request will add to the count of people in the establishment.

Functionality:

1. Authenticated customer will check in at vendor location
2. Authenticated customer will be able to see the number of other users checked-in
3. Authenticated user will see capacity

Data Model


Bands
Id
CoverCharge
Description
Genre
Name
OwnerId
Rating
VendorId
Bars
Id
Address1
Address2
ApplicationUserId
City
Hours
ImageURL
Name
OwnerId
OwnerName
Rating
State
VendorURL
ZipCode
VendorPhone
UserTokens
UserId
LoginProvider
Name
Value
AspNetUsers
Id
AccessFailedCount
ConcurrencyStamp
Discriminator
Email
EmailConfirmed
LockoutEnabled
LockoutEnd
NormalizedEmail
NormalizedUserName
PasswordHash
PhoneNumber
PhoneNumberConfirmed
SecurityStamp
TwoFactorEnabled
UserName
Signature
Beers
Id
AlcoholContent
Brewery
Name
OwnerId
Price
Rating
Type
VendorId
UserRoles
UserId
RoleId
Roles
Id
ConcurrencyStamp
Name
NormalizedName
RoleClaims
Id
ClaimType
ClaimValue
RoleId
UserClaims
Id
ClaimType
ClaimValue
UserId
UserLogins
LoginProvider
ProviderKey
ProviderDisplayName
UserId

Routes:

Client Routes:
GET"#/"
GET"#/vendors/
GET"#/vendors/{id}"

API Routes:

GET"~/api/accounts/"
POST"~/api/accounts/register/"
POST"~/api/accounts/login/"
POST"~/api/accounts/logout/"
GET"~/api/vendors/"
POST"~/api/vendors/"
PUT"~/api/vendors/"
DELETE"~/api/vendors/"
GET"~api/vendors/{id}/beers/"
POST"~/api/vendors/{id}/beers/
PUT"~/api/vendors/{id}/beers/
DELETE"api/vendors/beers"
GET"~api/vendors/{id}/bands/"
POST"~/api/vendors/{id}/bands/"
PUT"~/api/vendors/{id}/bands/"
DELETE"~/api/vendors/{id}/bands/