

Carve Connect Design Document

By: RJ Boucher, Dhruv Patel, Dylan Anderson, Christian Marcy,
Md Monir, Fred Budde, Brandon Wheeler
Team Lead: Sean Klinglesmith

High Level Description	2
Problem Solving Approaches	3
Technology Stack	3
Screens	4
Splash Screen	4
Existing User Sign In	5
New User Sign Up	6
Profile Create view	6
Profile Page edit view	7
Dashboard	8
Profile Page Personal view	9
Profile Page external view	10
User Search	10
Buddies List	11
Venues List Page	12
Venue Page	13
Carve Creation	13
Notifications Dropdown	14
Listing Page	15
Screen Navigation	15
Example Use Cases	16
Authentication and Security	17
Database Schema	17
Endpoints	19
Feature Goals	20
Mid-Assessment	20
Final Demo	20
Final Turn in	20
Planned Schedule	21
Front end	21
Back end	21
Testing	21
Project Manager	22
Closing remarks	22

High Level Description

The problem statement that we are trying to address arises from a need to facilitate connection between athletes and those who film. The purpose of Carve Connect is to provide a solution to that problem: through the use of a social media platform extreme sports athletes, filmographers, photographers, and fans of those sports are able to connect.

Upon arriving to the site the user will be given several options, take a tour to see what Carve Connect is all about, log in to Carve Connect, or sign up. The profile for the user contains relevant information including name, media, what sports they enjoy and whether they are filming or riding. The idea is that there are different types of profiles to enable connections between people with different skills.

Since this is a social media platform, the standard support for a “friend” will be provided through the use of a Buddy. A Buddy request can be sent to someone by clicking on their username, found through a user search, or if they have attended an open Carve with you. A user can also follow another user in the scenario that user really enjoys their content (they post interesting pictures, videos, tricks, etc.), but the relationship is not on a personal level. Being Buddies defaults following to be enabled and also allows for messaging.

A Carve is an event for a particular sport at a given venue at a specific time. For example, someone who is heading to a ski resort to snowboard creates a Carve, public so anyone who frequents that venue can see, Buddies’ Only for just for Buddies, or private.

Furthermore, the listings page would be roughly equivalent to a job/event search. The open Carves can have restrictions on the number of slots, so if you only want 1 photographer to go with you, you can do so, or if you are a filmographer and you need one athlete to film. Then it becomes open and anyone that finds it can apply. When trying to attend an open Carve you have to apply, the Carve creator has to approve that attendance, if successful you will be added to the Carve.

The second option of being notified of open Carves will stem from the venue page. Anytime a Carve is created the venue will be tagged in it. Users have the option of following a venue page, when they are following the venue, any Carves created at that venue will be sent to their feed.

Socially, users will get notified in their feed anytime a Buddy is attending a Carve, or if they create a Carve. A buddy can send an invite to you for a Carve that they create, if they want to try and convince you to go. Otherwise it will just show up in your feed and you can choose to attend if you'd like to.

The connections between people, finding someone to film you or finding an athlete to film, is the core of the problem that we are trying to address. These types of connections have been proven difficult in the past and is where Carve Connect aims to step in and resolve that problem. And since this is extreme sports, it's all about the shred sessions, so the Carve is half of the functionality of the platform. The other half is social connections, finding users through open Carves or by searching for them enables you to buddy request or follow them to view their content. Thus we have Carve Connect.

Problem Solving Approaches

The problem that we took on for the scope of this project is rather simple, provide a method of connection for athletes and those who document athletes through still images or video. Initially we had to come up with a high level concept to come up with a solution to the problem. We looked at the problem as one similar to current social media platforms such as Facebook, or along the lines of what LinkedIn tries to accomplish. Because of the parallels that we identified, we elected immediately to pursue a social media platform as our solution to the problem. Potentially a messaging application, a forum style website, or some form of job posting web interface could have been used. Upon deliberation and some quick research, the team arrived at the conclusion that a social media platform would indeed be the ideal choice for solving our problem.

The issue then arose of how exactly we were going to accomplish such a task. This initiated a design problem that we needed to identify a solution for. How were we going to host a website, and provide a robust enough back end to support it? Looking through various research avenues we began to identify Amazon Web Services as the primary hosting solution for our platform. Alternatives to it include Microsoft Azure, and various website hosting services such as WordPress. Upon further analysis and research, Microsoft Azure was found to be complicated and expensive so it was ruled it out. Looking at a website hosting provider such as WordPress, or others was also a viable solution, but again cost became an issue. Dr. Baliga offered us a solution to the cost woes, by offering to provide an EC2 instance hosted for us. Initially we were skeptical of it because when dealing with a social media platform, we would likely need a complex database. We looked into a database hosted by AWS, but the costs made it not likely to solve our problems. As a result, after discussion with Dr. Baliga he offered us a t2.medium instance of ec2, so that we can host a database, run a complex backend, and host the website on the front end all within the instance. Thus the decision was made to proceed with that infrastructure setup.

All that remained as a problem for us was to identify what tech stack we were going to proceed with on our instance. We considered several options for the front end, but quickly decided on React, due to the experience several group members had with it. As for the back end we looked for solutions that were easily compatible with React. Our two choices were python flask and node.js. Since the front end was using React.js, we elected to use node.js to keep the front end and back end in the same language. Beyond that the features of node.js allow for far more versatility and options for running the server. As a result the choice of our tech stack was made to use an ec2 instance, running a react front end, a mysql database, and a node.js backend. More details on the tech stack are in the section below.

Technology Stack

As mentioned above our choice of front end is to utilize React. It is highly robust and has a relatively easy workflow. The back end for the application will be run on node.js. It is an incredibly powerful, but lightweight backend that has a lot of features and functionality, ideal for making a social media platform. Mysql integration with the front end and back end were

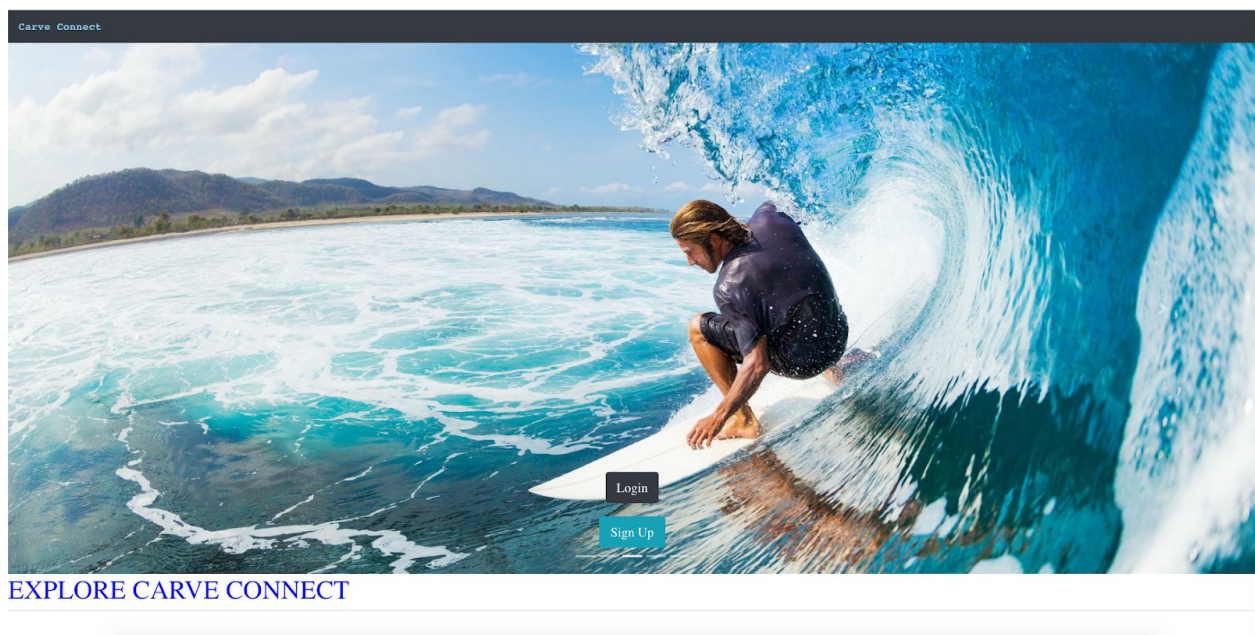
selected as it has the functionality we need to accomplish our goals. Express is the ideal solution for handling the communication between the database and the back end. The database schema includes tables for user profiles, Connections, Carves, venues and will be detailed later on in this document.

In summary:

React native(front end), node.js(backend), mysql (DB), express (BE/DB communication).

Screens

Splash Screen



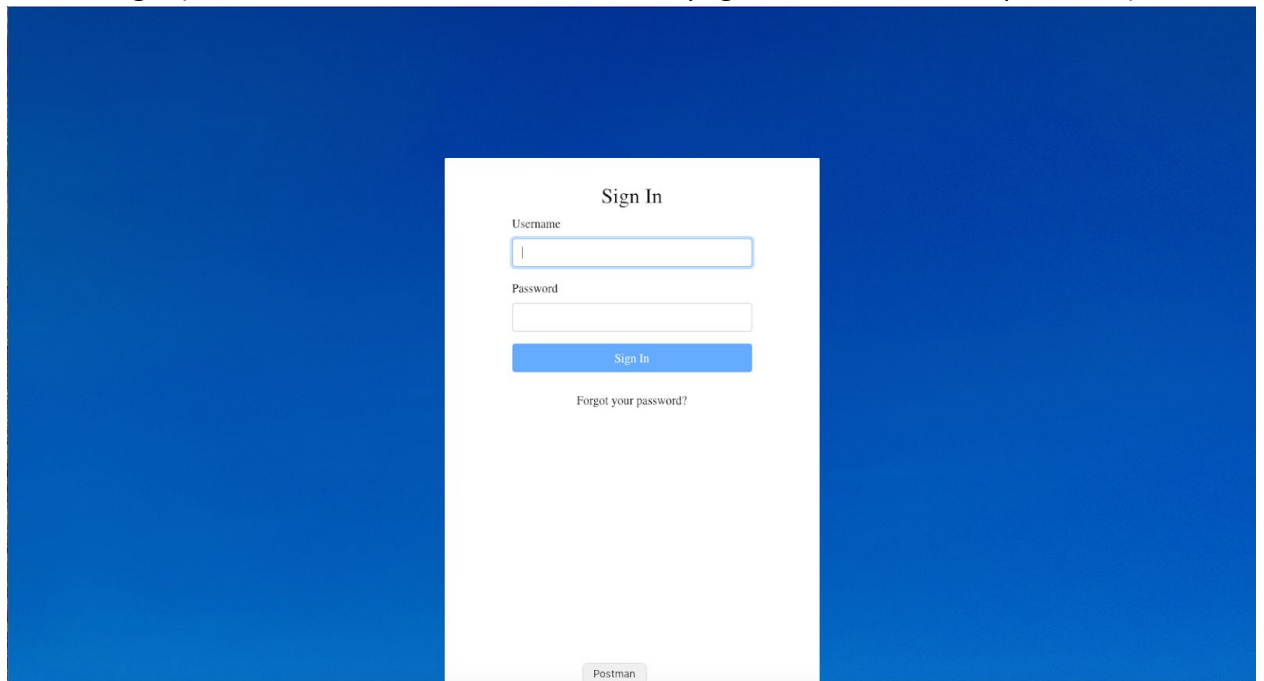
Upon arriving at the website, users are met with the splash screen. This page in the future can be used to show highlights of Carve Connect and everything it has to offer. This mockup alludes itself to that. The main transaction that occurs here is beginning a login session, or opening up the sign up/ create profile session. The only potential endpoints to look at for this screen would be any related information for highlighting the functionality of Carve Connect as a whole, such as: GET - /users/featured_users (It will return the list of featured users to show). Currently the splash screen does not make use of any endpoints.

Existing User Sign In

User's who already have an account must enter their details to reach their profile. Simple login transaction will occur on this screen. The user provides input, and then that input is compared against the user accounts in the system. First the username is checked against the usernames within the user table, if it is confirmed to be correct, it checks the password for that entry against what the user provided in the password field on the screen. If it fails to identify

the user, or the password fails, the failed login message will appear. The endpoint for this is the users table.

POST - /login (return the information from the users page like username and password)



Sign In

Username

Password

Sign In

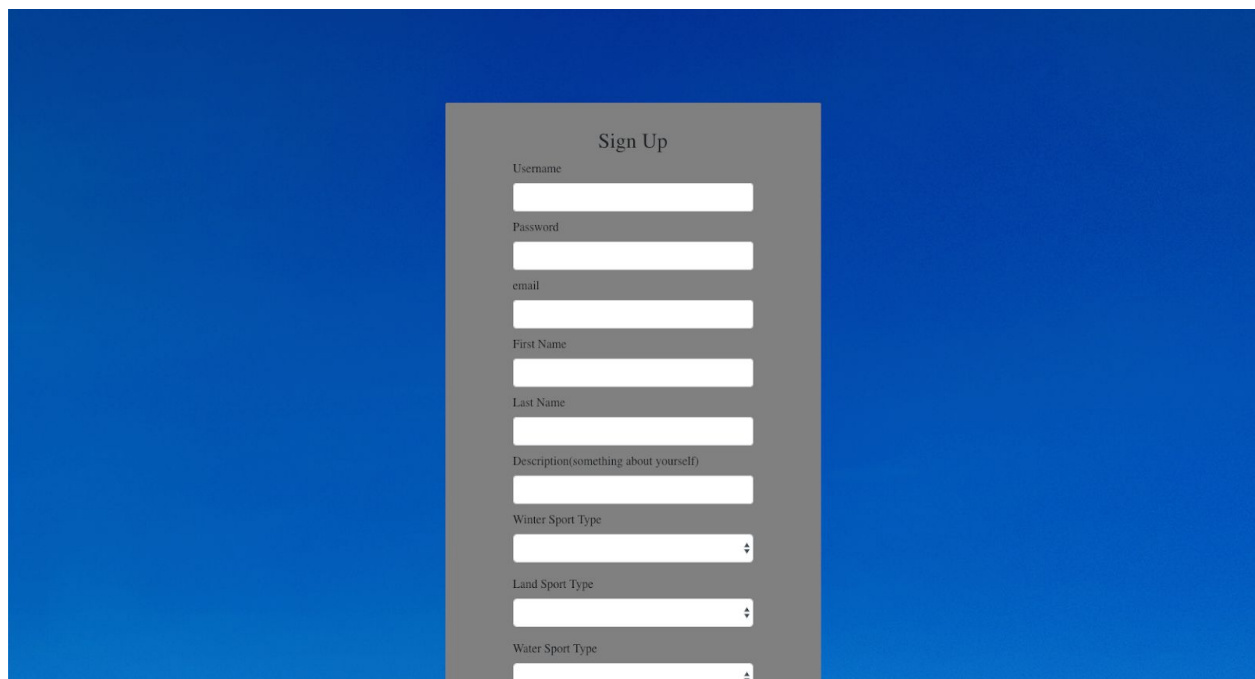
[Forgot your password?](#)

Postman

New User Sign Up

Those that are viewing the website for the first time and wish to become a part of the community must make an account in order to create a profile. The idea here is to create a new entry in the users table. A query is run to check if the username exists, if it does not, the new entry in the table is created, assuming that the password matches the specifications for password that we define. As of now simple alphanumeric, 4 characters will suffice. More on the password creation information in the security section that follows. Endpoints required:

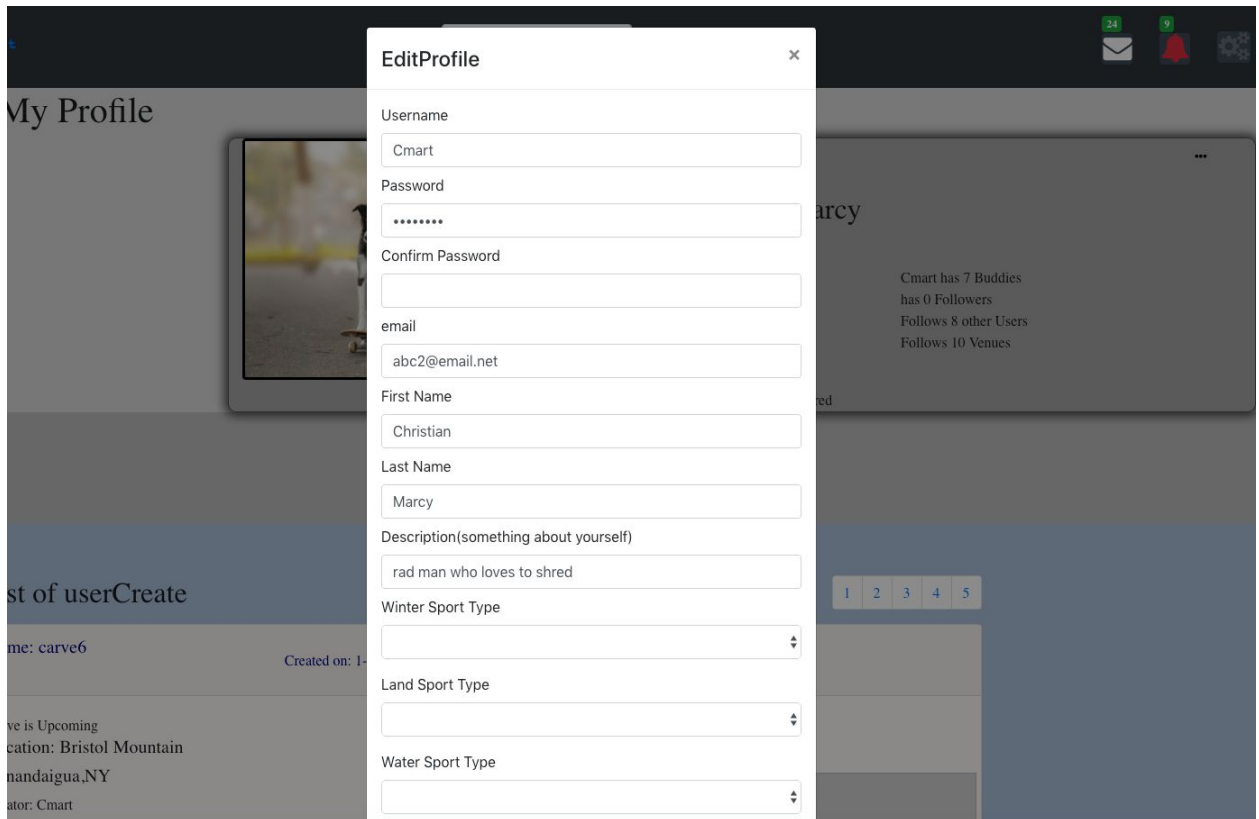
POST -- /users (send the username and password in post body for basic security)



Profile Page edit view

Make changes to the profile. This screen is almost identical to the profile creation screen. The only difference is that since you currently have a profile, your information is present in the fields. This enables you to edit the fields, or make different selections and save them, updating the database and one that is complete, it will update your profile as well. The profile information will be the current information stored for the user. This screen will enable the user to customize their profile. The endpoints needed for this are

PUT - /users (Get the users information in a more secure way)

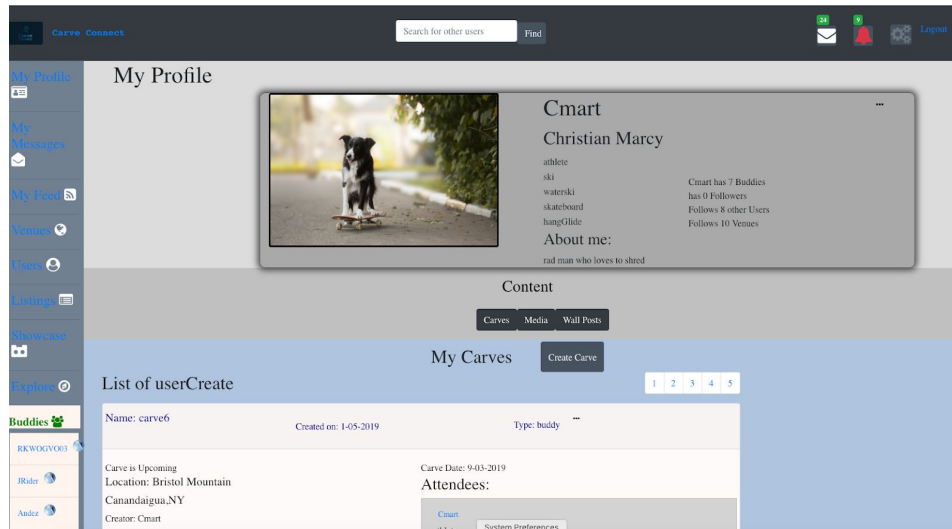


Dashboard

This is the primary screen available to the user. Utilizing the React Framework allows users to stay on one page and have the content change by what they select on the side. The dashboard contains the buddy list which is detailed more below. There's also links to my profile, explore page, venue list page, my feed, carve listings page, media showcase page, and my messages. At the top of the dashboard is the navbar containing messages, notifications, and settings drop-down menus. By default, when logging in, the user is presented with their profile information.

Personal view

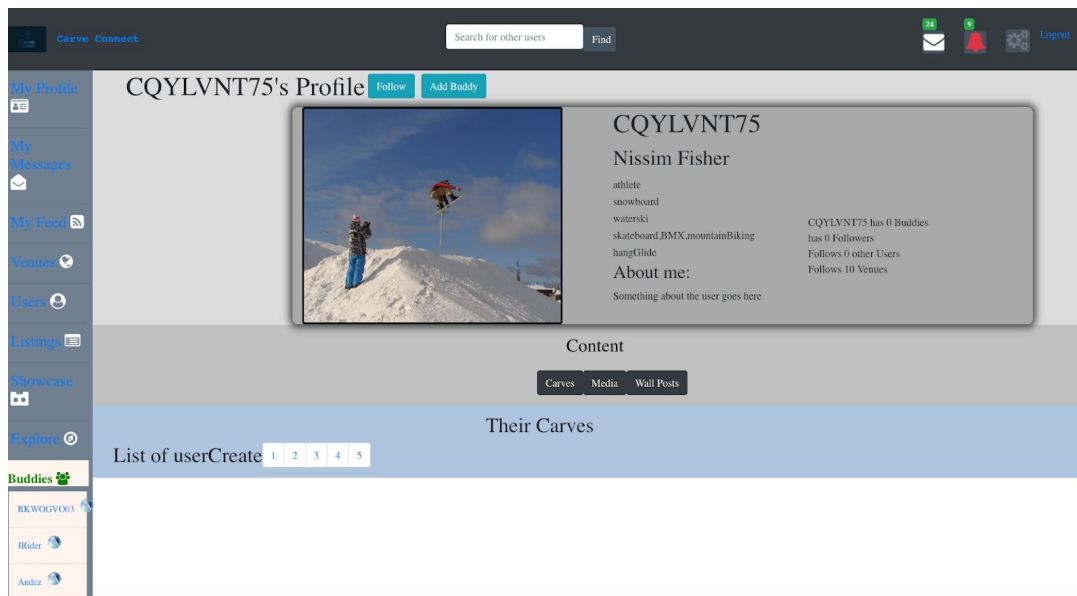
View your profile with access to edit it. All of the relevant information will be displayed on this screen. The main feature here is that this is your own profile view. It also gives you the option to edit your profile and change your information on this page. The data needed is profile type, sport, username, first name and last name. The endpoints needed are: GET - /users/:user_id(Only get the relevant information that the user needs like name but excludes password ect.)



Profile Page external view

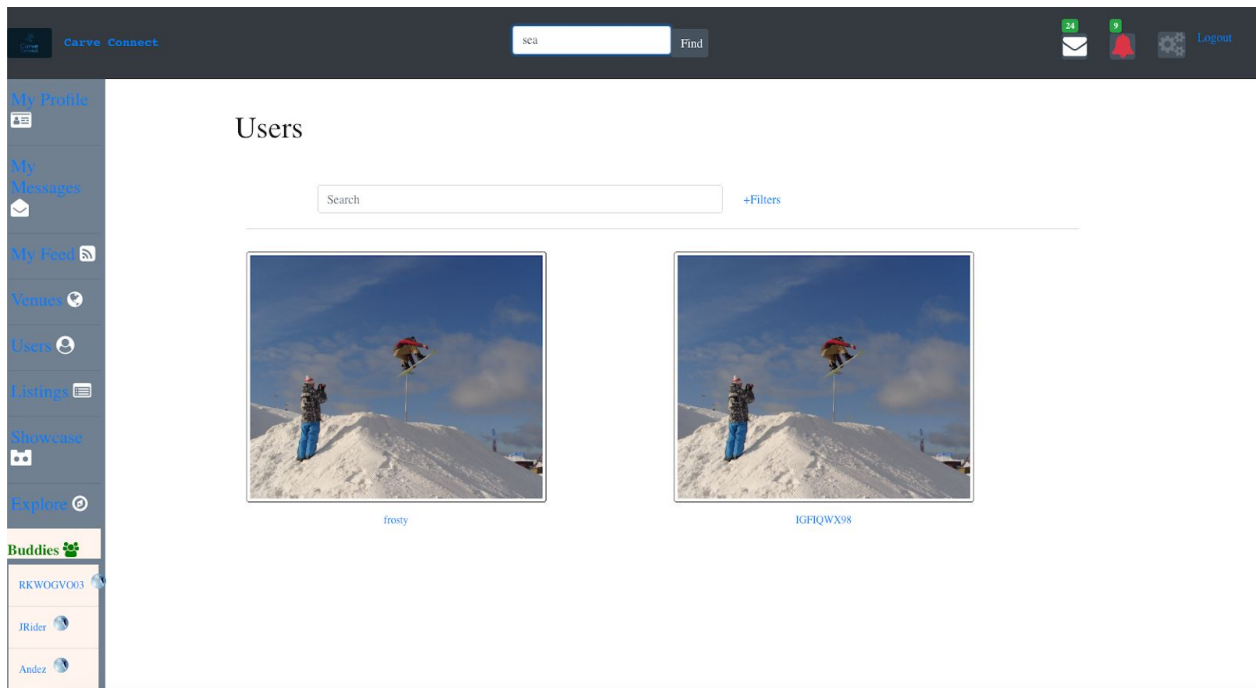
Allows other users to view your profile. This page contains all of your relevant information. It is the view that someone besides you will have of your profile page. If they are not your buddy, you can add them. If they are your buddy, that button will not be visible. Eventually privacy settings will decide what on this screen is visible publicly. The data here is the user's profile information, without the user's login information. The endpoints needed are:

- GET - /users/:user_id(This is the same as above but since we removed the personal info we can also use it for other people viewing the pages)



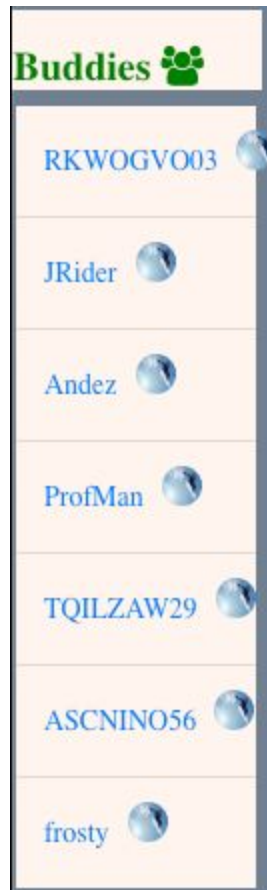
User Search

Utilizing the search bar at the top, users are able to search by username, first name, or last name to try and find someone. The endpoints needed for this are: GET - `/users?search=searchTerm` (Where searchTerm is a username, first name or last name).



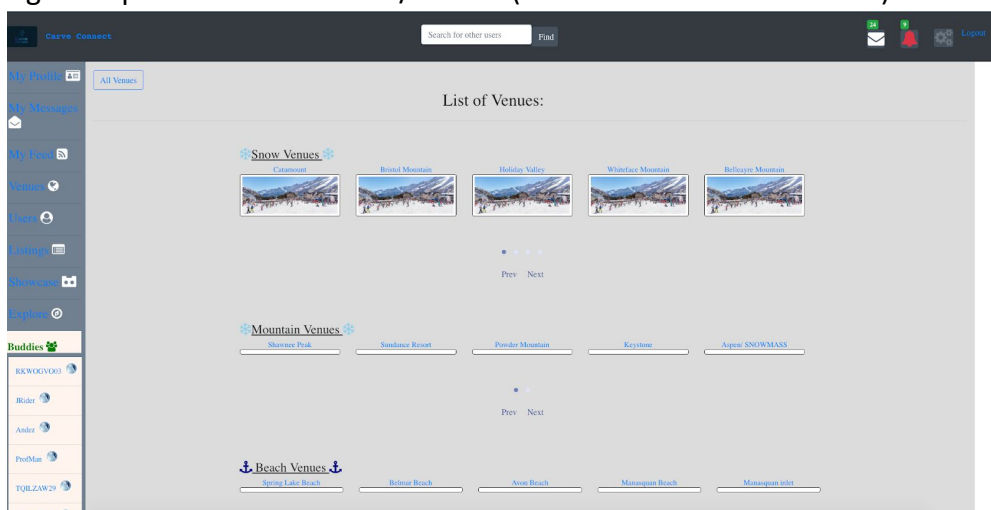
Buddies List

Shows list of buddies. The username of your “buddies” will show up in this list. If the username is clicked, it will open your buddies profile. The endpoints needed for this are: GET - `/follows/buddies`



Venues List Page

List of all current Venues. Clicking on a venue in the list will open up its corresponding venue page. Endpoints needed: GET - /venues (return the names of venues)



Venue Page

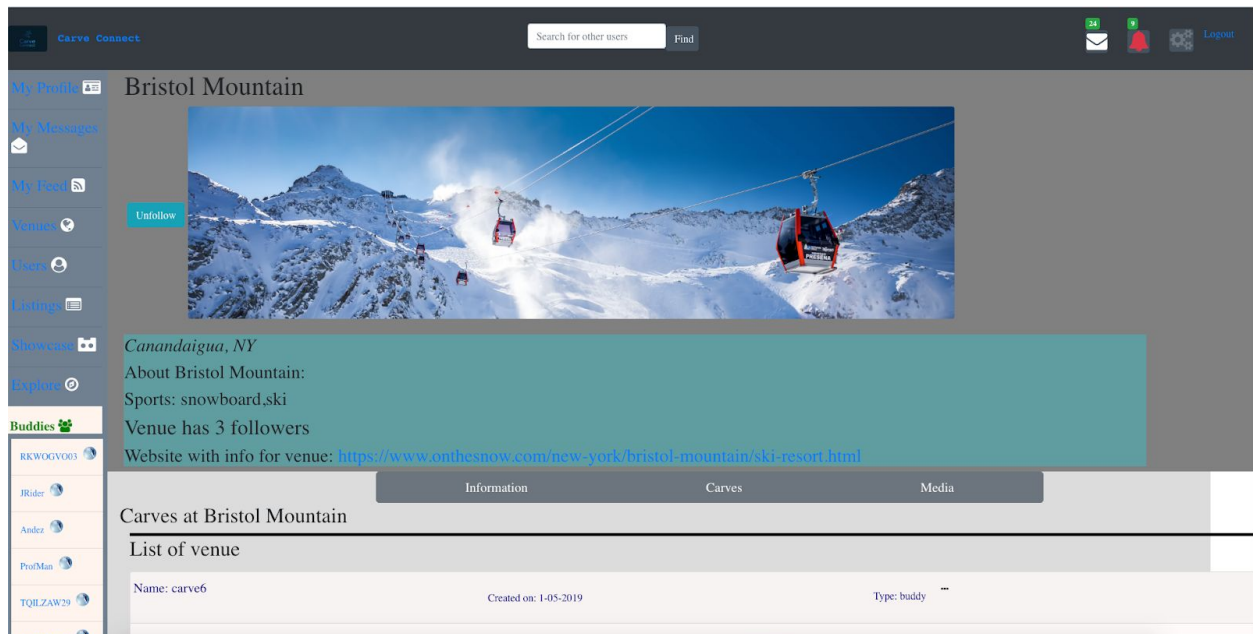
Page for venue information (such as local conditions). It has detailed information for the venue and what is offered in terms of runs. It will contain any Carves occurring at that venue, and also contains a way to showcase photos or video taken at the venue, done through linking pictures or video and tagging the given venue. When a Carve is created, it will be added to this list. The Carves in this list, are shared to anyone following the venue, being placed in their feed. The endpoints needed are: GET - /venues/:venue_id(get the specific venue information)

GET - /utilities/darksky/:venueId

GET - /utilities/darksky/past/:venueId

GET - /venues/:venueId/carves

GET - /media/venue/:venueId



Carve Creation

Allows user to create a Carve. This would act as a pop-up when a user chooses to create a Carve. Here the user has toggled the need for Filmers. A carve is upcoming until it has been completed. Once completed, it will have a way to embed videos. In either case, comments to

the carve will be enabled alongside a like/ dislike system for the carve itself and each of its subsequent comments. The endpoints needed to create a Carve are:

POST- /carves (create a new Carve and store it in the table)

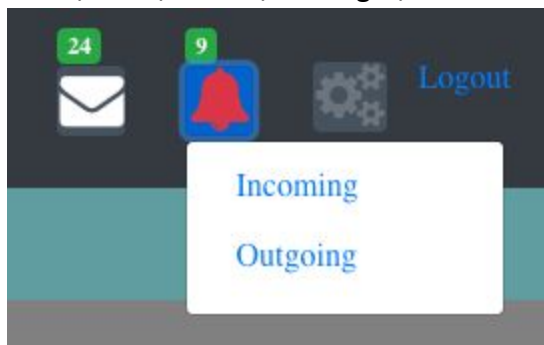
The screenshot shows a web application interface. On the left is a sidebar with navigation links: My Profile, My Messages, My Feed, Friends, Groups, Settings, and Buddies. The main content area is titled 'User's Feed' and shows a 'List of users' with details for a user named 'carve1'. Overlaid on this is a 'New Carve' form. The form fields are: Name (with a placeholder 'Carver's Carve'), Carve Type (dropdown menu), Sport Type (dropdown menu), Venue (dropdown menu), Date of Event (calendar icon and date input), Description (text area), PhotoSlots (input field), and AthleteSlots (input field). At the bottom right of the form are 'Close' and 'Create' buttons. The background also shows a 'Users' Followed Carves' section with a grid of carves.

Notifications Dropdown

Users will have access to a notifications menu displaying things urgent to the user's attention. For the current design, once a user logs in a bell icon will take place in the navigation bar. Buddy requests and Carve requests will appear here. Messages received, carve attendance requests will also be listed here.

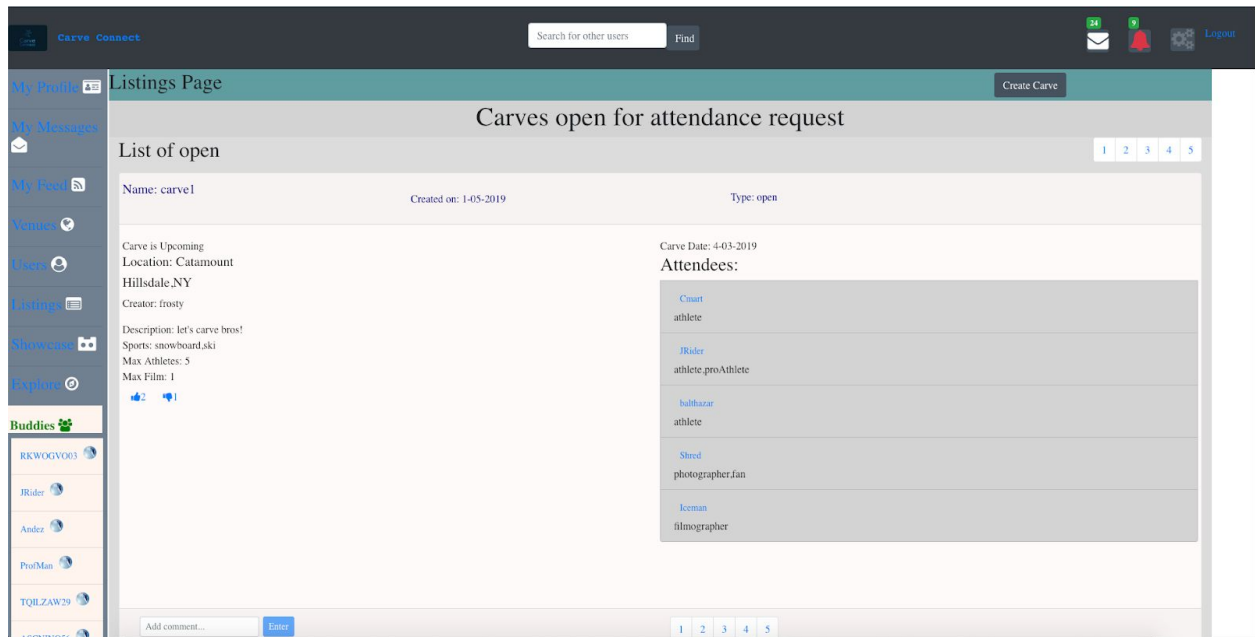
GET - /users/:userId/messages

GET - /users/:userId/messages/notifications



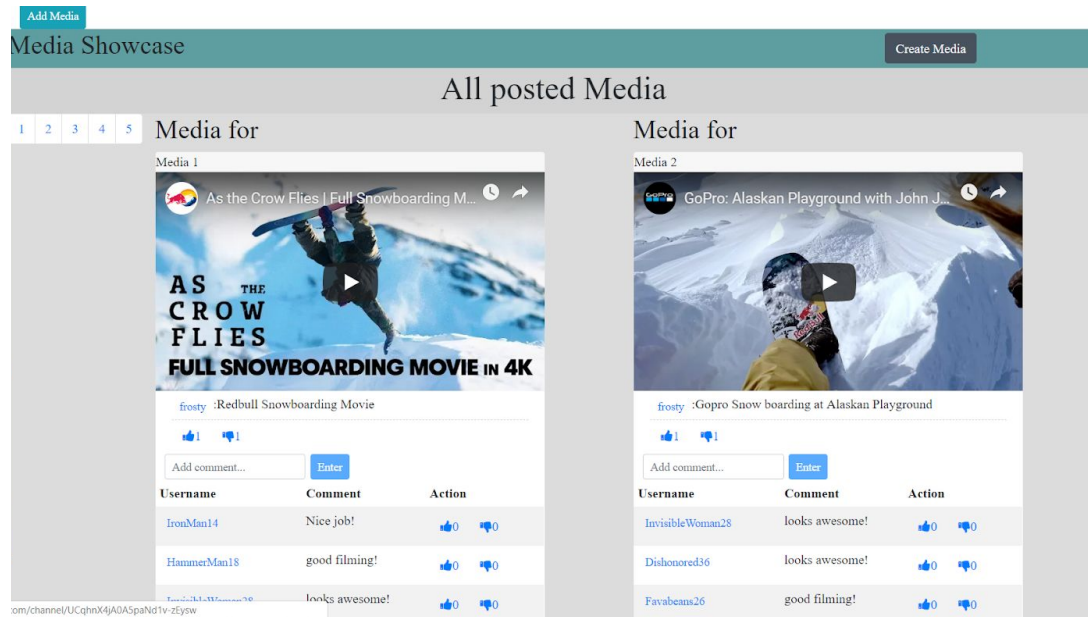
Listing Page

All open Carves in the system are posted here as a central hub to look for potential jobs. Empty Carves can be posted here if the user doesn't specify when or where and is just looking for someone. The endpoints needed are: GET - /carves/open (Show the different spots you can go to)



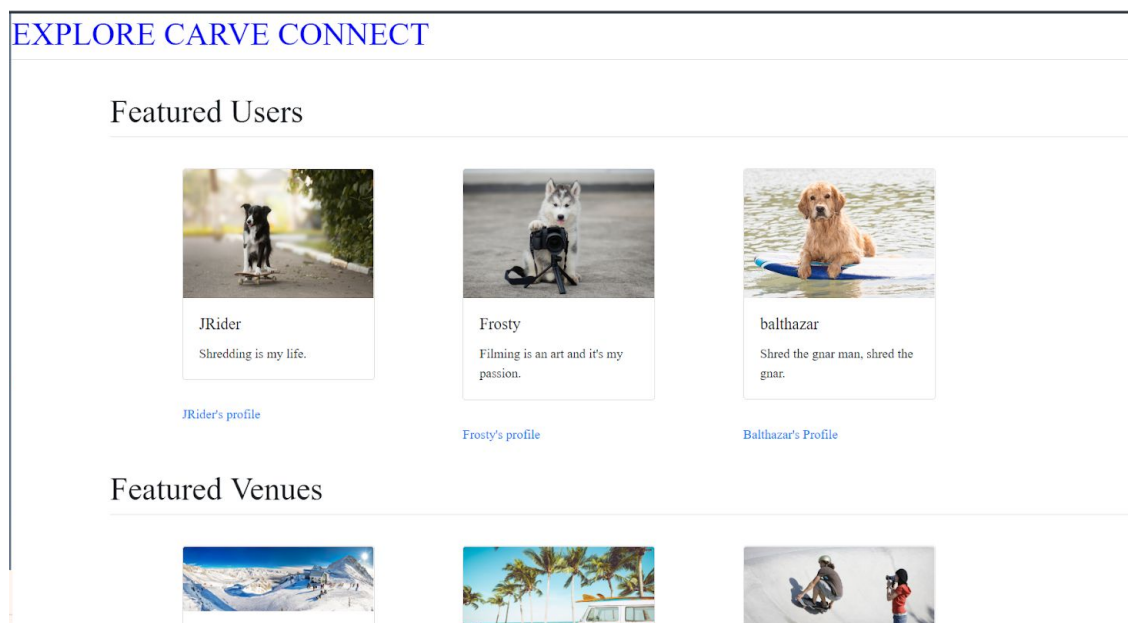
Showcase Page

The showcase page is a collection of all media that users have posted to the site. This is where you go if you want to see some cool tricks or whatever video content other users have posted. Endpoints needed: GET - /media



Explore Page

The explore page gives you a small snapshot of what the site has to offer. Featured users and featured venues are shown here



Screen Navigation

When a user visits the web application they will land on the Home or Splash page. If this is the user's first time on the website they can follow the Signup button to be prompted to fill

out a new username and password. Following the creation of a new account, they will be brought to the create profile screen to fill out details about themselves landing them at their freshly made dashboard screen. If they are already registered users, they can bypass this process by following the Login button to be immediately brought to their dashboard screen. The dashboard screen will act as the epicenter for all functions/routes that the application provides.

A goal of Carve Connect is to bridge together users who possess different skills to create something meaningful. Users are able to achieve this goal through the medium of Buddies. The idea is that you find a user by searching for their username, first name or last name, once found you proceed to do a Buddy request with them, in which you send a small message that identifies who you are to them. If they accept, you are both now Buddies and can do Carves together. A Carve is an event that is at a venue, that exists within the system, on a given date between 1 or more users.

Example Use Cases

- Login - navigate to login button, click login, enter username and password, hit enter
- Create account - navigate to the sign up button, enter the relevant information on the signup page
- Edit profile - navigate to the top of your profile page, click edit profile, edit information you want to edit
- Find user - navigate to search page, type in username/first name/ last name into search bar, hit search and results will appear based on users in the database.
- View user- if a user is found through search, clicking on their name takes you to their profile.
- Add buddy- navigate to a user's page and click add buddy to send a buddy request message
- Find venue- browse the list of available venues, sorted by sport
- View venue- if found through navigation, click the venue name to be taken the venue page.
- Follow venue - navigate to a venue's page and click follow venue
- Create a Carve - on the dashboard page there will be a button for the user to create a Carve. The dialogue box will open and the appropriate fields will be filled in to create a Carve at a venue on a given date.
- Invite buddy to Carve- the Carve creator navigates to the Carve's page, they click invite to Carve, they then enter the username of the person they want to invite to the Carve
- Ask to attend a Carve - if a Carve shows up in your feed, you click a button to ask to attend it
- Allow attendance to a Carve- if someone asks to attend your Carve, you can accept or decline them, buddies will be automatically accepted .

-
- Respond to invite to attend a Carve - user is notified if a buddy invites them to a Carve, once the dialogue box opens you can choose to accept or decline the invitation.
 - View buddy- if the buddy's username is clicked in the buddy list, you will be taken to the buddy's profile page
 - View buddy's Carve- anytime a buddy creates a Carve, it will automatically show up in your feed when you log in, or if you are online when it is created it will pop up instantly into your feed.
 - View feed - navigate to the Dashboard feed posts will be visible there.
 - Post job request - navigate to the listings page and there will be a button to create a Carve there. Carves done this way will be open Carves, you specify how many slots of athlete/photographer you want, can specify venue if you wish, can specify date if you wish. It is open ended to promote a connection between you and potential users to Carve with.
 - Find an open Carve - navigate to the listings page to search/ browse for open Carves until one suits your fancy.
 - Request open Carve attendance - once a suitable open Carve is located, click on it to submit a request to attend the Carve.
 - Choose to approve an open Carve attendance request - if someone sends you a request to attend an open Carve, you receive a notification. Upon clicking on the notification, you are taken to the approve/reject screen. Since it is open, multiple users may choose to try and attend, and the amount of approvals is dependent on the size of the open Carve.
 - Logout - navigate to the top of the page and click the logout button
 - Delete profile - navigate to the settings page, then click account, then click delete profile, click yes on the acknowledge prompt

Authentication and Security

The app's user authentication will be performed using a simple password authentication until more security is required. The idea is that a login transaction will be created, and will perform a check of the password against the user's account. If it passes the login transaction will success and enables the user access. The password can have constraints and requirements that are updated as we tighten the security that we are using.

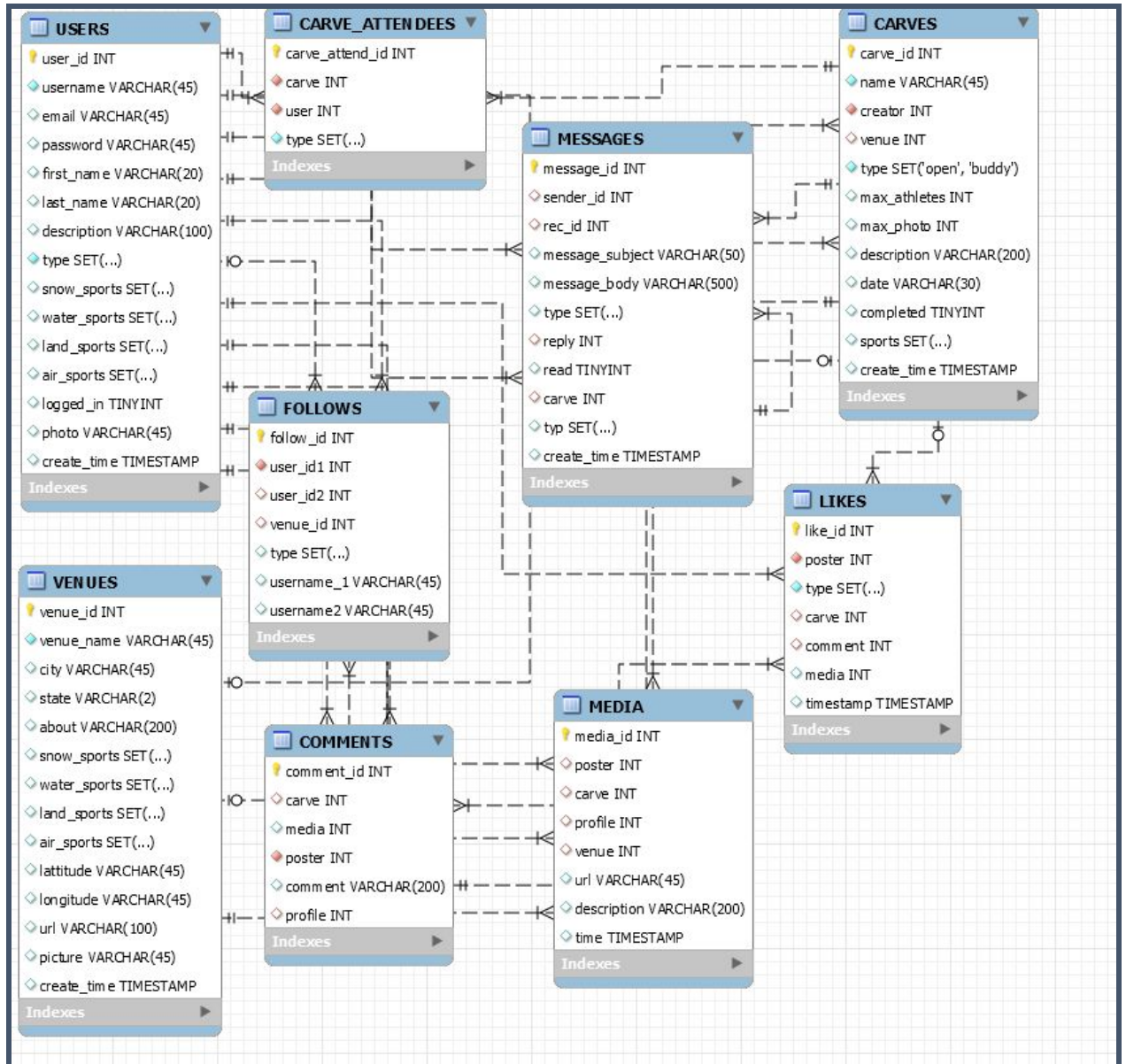
Over time encryption algorithms will be used to make it more secure, but since this is more of a development proof of concept it will not be going explicitly live. Therefore tighter security is not a priority, but can be implemented if things change. Looking at Google Oauth as an alternative, or addition is going to be considered later in development.

Database Schema

Our platform is a web based platform involving numerous users and various ways in which those users can interact with one another. Since the solution to our problem revolves around connecting people, or users, it was determined that a relational database would be the

preferred way to handle it. The key to this schema is everything is centered around independent users. Every user will have a profile, a list of buddies and connections, and a list of venues that they follow. What this means is that when a user opens a transaction to make a “Buddy”, if the buddy accepts, the DB table BUDDY gets updated that the user id of the buddy will be stored as a key value pair. Following a venue would be similar except that it’s a one way acceptance meaning if you choose to follow a venue, then that value pair is made. The key is designing the relationships between the entities that we are using within the application, and designing views to support access to those relationships. The simple 8 table database is quite robust in its functionality. It allows for creating a new user, looking up a user, searching through the user table, and potentially enable sport or type based content. The venue table is also straightforward, eventually enabling an interface for a company to create their venue page, update the venue, or to use any of the information stored there. The follow venue relationship is intended to push the content for a venue to the users that follow it. Much in the same way content generated by a buddy will be pushed to the users that they are buddies with. A venue has multiple Carves associated with it. Again as mentioned anyone following the venue gets notified of the Carve.

The EER diagram of our database schema is shown below:



One thing to note with this schema is that the relationships are all 1 way. User and venue do not have any foreign keys contained within them. This allows for easy insertion into those tables. Any of the relationship tables will require foreign keys into them, but that will not be an issue due to the entries having been created already. Whenever a relationship is established between users or between a user and a venue, the appropriate entry is created in the corresponding table. The id of that user or venue is a foreign key, that way the relationship can be identified by a query for all entries for that id. A Carve is no different in that respect, except that it is 1 Carve creation user, 1 venue location, and potentially 1 or more users that choose to attend. Comments are enabled for the Carve and exist as again a many comment to 1 Carve relationship. Once a Carve is completed, videos/ photos can be embedded into the Carve via the embed table, again as a many embed to 1 Carve meaning multiple photos/ videos can

be added to it. The last table is the message table. This exists between 2 users, and functions as both a means of communication and as the format for a buddy request transaction. It also carries the field of reply message as an id of the previous message id that it is in reply to. A social media platform is all about multiple entity types being able to connect to one another. The relationships between the entities can best be accomplished using MYSQL as a database, the relationships outlined in the schema above show that our specific entities will have the relationships needed for the functionality we desire. Our MYSQL database schema as shown should allow us to complete our project and is robust enough to allow for our feature set to be completed.

The follows table has a one to one relationship. The user table has one to one relationship with venues table. Carves table has one to one relationship with venues and one to many with users table via carve attendees table. Media table has one to one relationship with profile and venues table. Comments table has many to many relationship with media, carve and profile. Finally likes are many to one to comment, carve, and media.

Endpoints

Comprehensive list of the endpoints that we are going to need for the project. The details of how these are used relative to each screen, is available in the screens section.

1. POST - /user/login -- Returns user_id if they exist or null if they exist in the database
2. POST - /user/signup -- Inserts a row in the database of a new user with necessary fields
3. GET - /user/:user_id -- Gives specific user back to front end
4. PUT - /user/:user_id -- takes info and updates row of associated user
5. GET - /venues/ -- list of venues
6. GET - /follow_user/:user_id -- Gives buddies of the user id back
7. POST - /follow_user -- shows all of the follow users that are in the Database
8. GET - /venues/:venue_id -- takes id of specific venue and returns specific venue info
9. GET - /users/search-- get the user that are being searched on the front end
10. GET - /carves -- return all of the carves that are on the Database
11. GET - /carves/:venue_id-- get the carves that are at the specified venue
12. GET - /carves/:carve_id -- get the information at the specified carve
13. POST - /carves -- update the carves table on the Database
14. PUT - /carves/:carve_id -- add a new carve to the carves table with the carve_id
15. '/users' -- return all of the users
16. '/venues' -- return all of the venues
17. '/carves' -- return all of the carves
18. '/follows' -- return all of the follows
19. '/messages' -- return all of the messages
20. '/comments' -- return all of the comments
21. '/media' -- return all of the media
22. '/likes' -- return all of the likes
23. '/login' -- return the login

-
24. '/carveAt' -- return where the carve is at
 25. '/utilities'
 26. '/carves/:carveld/users' -- return the users that are at the specified carveld
 27. '/carves/:carveld/carveAttendees' -- return the carveAttendees that are at the carveld
 28. '/carves/:carveld/comments' return the comments that are at the carveld
 29. '/carves/:carveld/media' -- return the media that is at the carveld
 30. '/carves/:carveld/likes' -- return the likes that are at the carveld
 31. '/users/:userId/carves' -- return the carves that the user is associated with
 32. '/users/:userId/follows' -- return the users that the userId follows
 33. '/users/:userId/messages' -- returns the messages that the userId has
 34. '/users/:userId/comments' -- returns all the comments at the userId
 35. '/users/:userId/media' -- returns the media that the userId has
 36. '/users/:userId/likes' -- returns the likes that are at the userId
 37. '/users/:userId/carveAttendees' -- get all the carveAttendees the userId has
 38. '/venues/:venueId/media' -- return the media at the venue
 39. '/venues/:venueId/comments' -- return the comments at the venueId
 40. '/venues/:venueId/carves' -- return the carves that the venueId
 41. '/venues/:venueId/follows' -- return the people who follow the venueId

Feature Goals

Mid-Assessment

PAGES: Profile(all views), Carve(open and buddy), User Search, Listing, Venue List, Venue profile

INTERFACES: notifications, buddy request, follow user, CARVE: edit, attend, accept attendance, invite to Carve, accept Carve invitation, complete Carve; search for user, follow venue

ENDPOINTS: /users, /venues, /carves, /carves/listing, /user/follow_user, /user/buddy_request, /user/notifications

MYSQL PROCEDURES: login, create_user, get_profile_personal, get_profile_other, buddy_request, accept_buddy_request

MYSQL TABLES: user, venue, follow_user, carve, follow_venue

Final Demo

PAGES: carve-past, message,

INTERFACES: comment carve, message, like/dislike carve, embed photo/video to Carve

ENDPOINTS: /user/message, /carve

MYSQL PROCEDURES: embed_media, comment_carve, like_carve

MYSQL TABLES: message, embed, comment

Final Turn in

Fix whatever is broken in all phases of the project.

Planned Schedule

Front end

Team members: RJ Boucher, Christian Marcy

Goals for the mid assessment are: create the pages required for the mid assessment goals, talk to and verify functionality of the back end endpoints and subsequently the database, apply the data flow/ user interaction needed to enable functionality of the mid assessment pages.

Goals for the final demo are: establish the entire database required for desired features, develop mysql procedures for the associated endpoints, write the back end code to allow for front end functionality for desired final demo goals.

Goals for the final turn in are: fix any issues with the database, develop any missing mysql procedures, write back end code for any missing features, debug existing code where required.

Back end

Team members: Dylan Anderson, Dhruv Patel, Fred Budde, Md Monir

Goals for the mid assessment are: establish the basic database required for desired features, develop mysql procedures for the associated endpoints, write the back end code to allow for front end functionality for desired mid assessment goals.

Goals for the final demo are: establish the entire database required for desired features, develop mysql procedures for the associated endpoints, write the back end code to allow for front end functionality for desired final demo goals.

Goals for the final turn in are: fix any issues with the database, develop any missing mysql procedures, write back end code for any missing features, debug existing code where required, clean up the code, add comments for more readability.

Testing

Brandon Wheeler

Goals for the mid assessment are: basic use case testing for features defined above for mid assessment, oversee/ assist in creation of trello work cards for the mid assessment.

Goals for the final demo are: advanced use case testing for features defined above for final demo, oversee/ assist in creation of trello work cards for the final demo.

Goals for the final turn in are: make sure it works, check off feature list and confirm they have been added, full testing to confirm functionality with minimal bugs.

Project Manager

Sean Klinglesmith

Goals for the mid assessment are: update server with improved infrastructure capabilities, assist other teams where needed.

Goals for the final demo are: oversee team development, push team to adhere to timeline goals.

Goals for the final turn in are: verify everything is finalized, review entire application and validate tests to confirm ready to submit.

Closing remarks

Our ambitious goal of creating a social media platform for athletes and those who film them to connect is underway. This design document is our guide to making that become a reality moving forward. A lot of work lies ahead, and the design put forward in this document will guide us and keep us on track moving forward. Everything in this document should make is

way into the final project turn in, but there will in all likelihood be minor differences between the design mockups, endpoints, and database schema. The more that gets developed and pushed, the more testing can be done, and the more changes/ fixes we are going to identify. As of now, this is our design outline for the project.