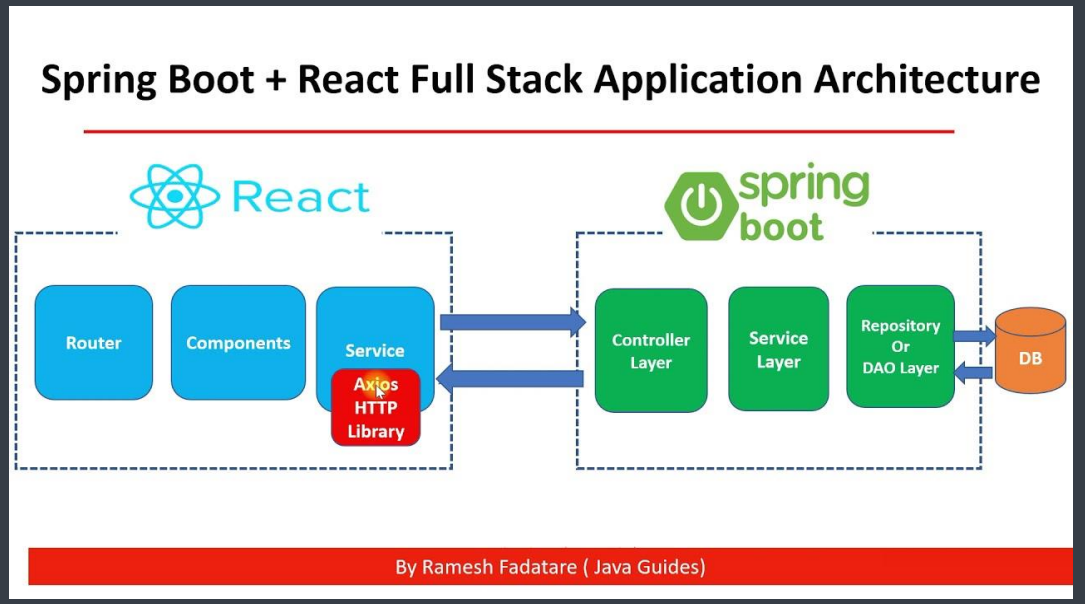
Logo generated by DALL-E

RETAPPED: DEFINED

- Social Media for beer enjoyers
  - Friends
  - Share drinks
  - Flavor profile
  - Rating system
- View key beverage information
  - Abv content
  - ingredients
- Locate alcohol vendors
  - Thirsty? Find the closet vendor

RETAPPED: ARCHITECTURE



SOURCE: JAVA GUIDES

RETAPPED: STORIES

- User Profiles
- Flavor Profile
- Search Component
- Maps
- Database

RETAPPED: REQUIREMENTS (USER PROFILE)

User Story: User Profile

As a new user, I want to create a new account so that I can have a user profile. As an existing user, I want to be able to access my user profile so that I can see what beers there are to drink and what my friends are drinking.

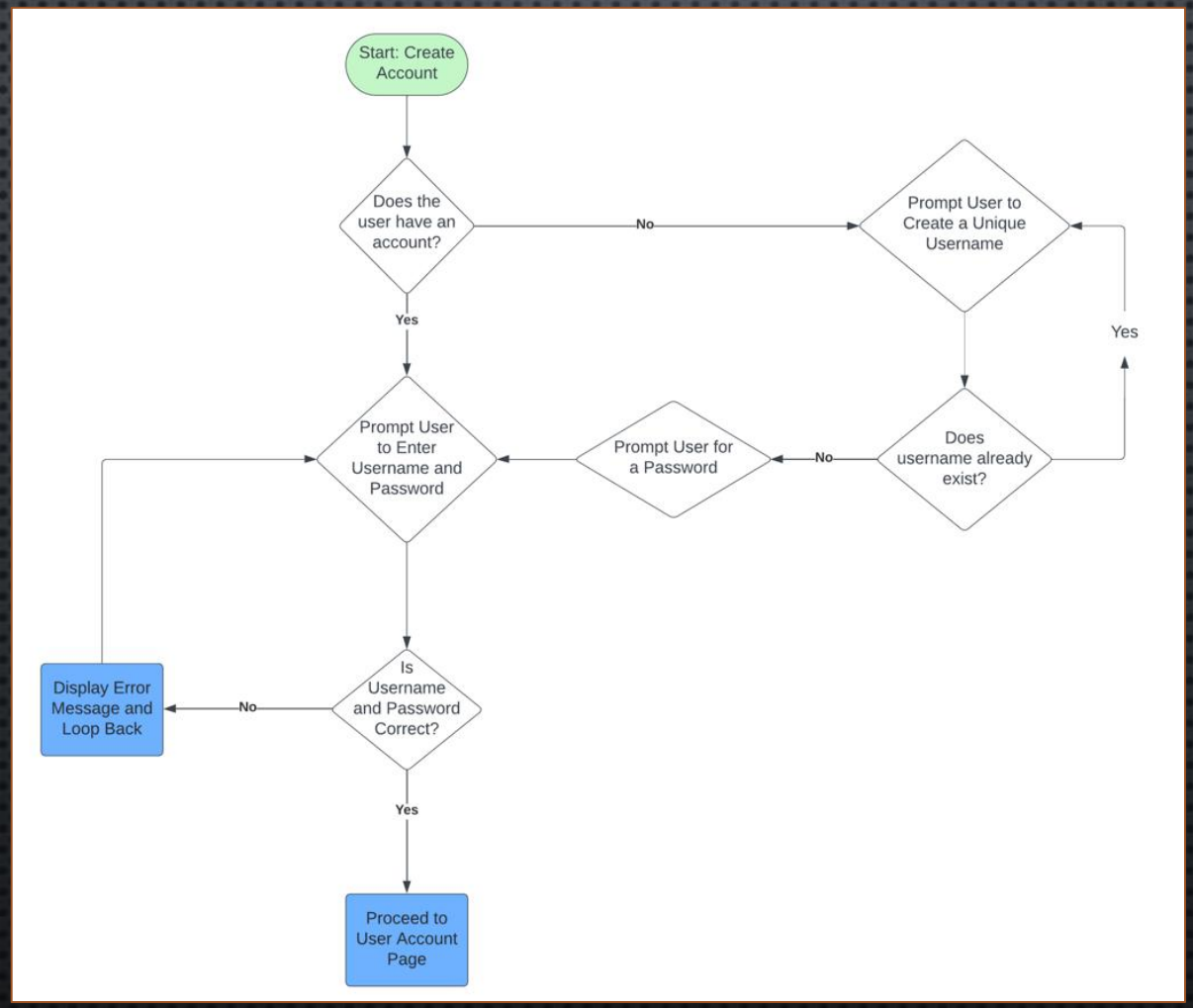
1.11 The Retapped system shall allow a user to create a new account with an input username and password.

1.12 Each new user shall have a unique username.

1.2 Each user shall have a "friends list" that will include other users that they choose to "follow"

2.3 The Retapped system shall allow users to share beverage selections with other users.

USER PROFILE FLOWCHART



USER PROFILE CODE (1)

UserProfile.java 1

Users > ramen > OneDrive > Documents > WSU Dayton > Spring 2024 > (4) Intro to Software Engineering > Group Stuff > RetappedUserProfile >

```java
package com.retapped.RetappedUserProfile; // Declare a package for the user profile creation program.

import java.util.regex.Matcher; //Used to search for patterns.
import java.util.regex.Pattern; //Defines patterns.

import java.util.ArrayList; // import the ArrayList class

// Annotations to interact with the database.
@Entity
@Data
@Table(name = user)
@Table(name = friends)
// Construct a user profile based on the user's input in Main.
public class UserProfile {

    //global fields
    private String username; //username for the account; each is unique
    private String password; //case sensitive password for the account
    private ArrayList<String> friendLst = new ArrayList<String>(); // friend list in database.

    //default constructor
    //creates a new user profile with null values
    public userProfile() {
        this.username = null;
        this.password = null;
    }
```

```java
    //constructor for the profile
    //creates a profile with a unique username and a password
    public userProfile(String username, String password) {
        // Username and password requirements:
        // Each user must have a username and password by creating it with characters [a-z], [0-9], and special characters (@, !, etc).
        // The user will not be able to create them with invalid characters such as `[]` and `\`.

        // Define patterns to search for in usernames and passwords.
        Pattern letter = Pattern.compile(regex:"[a-zA-Z]");
        Pattern digit = Pattern.compile(regex:"[0-9]");
        Pattern special = Pattern.compile (regex:"[!@]");
        Pattern invalid_special = Pattern.compile(regex:"[\\[\\]/\\/]");

        // Create matcher variables for the username.
        Matcher UN_hasletter = letter.match(username);
        Matcher UN_hasDigit = digit.match(username);
        Matcher UN_hasSpecial = special.match(username);
        Matcher UN_hasInvalid = invalid_special.match(username);
```

```java
        // Create matcher variables for the password.
        Matcher PW_hasletter = letter.match(password);
        Matcher PW_hasDigit = digit.match(password);
        Matcher PW_hasSpecial = special.match(password);
        Matcher PW_hasInvalid = invalid_special.match(password);

        //Check to make sure the input is not null.
        //Next, do the following code:
        if(username != null && password != null){
            //Check if username is valid and print a message if it isn't.
            if(!UN_hasletter.find() == true || !UN_hasDigit.find() == true || !UN_hasSpecial.find() == true || !UN_hasInvalid.find() == false){
                System.out.println(x:"INVALID USERNAME!");
            }
            // Check if password is valid and print a message if it isn't.
            else if(!PW_hasletter.find() == true || !PW_hasDigit.find() == true || !PW_hasSpecial.find() == true || !PW_hasInvalid.find() == false){
                System.out.println(x:"INVALID PASSWORD!");
            }
            // Set the username and password if both are true.
            else{
                this.username = username;
                this.password = password;
            }
```

Live Share Java: Ready Ln 40, Col 1 Spaces: 4 UTF-8 CRLF Java

USER PROFILE CODE (2)

```
14  public class UserProfile {
73      //getter for the username
74      public String getUsername(){
75          return username;
76      }
77
78      //Check to see if the Username entered exists.
79      public boolean usernameExists(String inputUsername){
80          //Return true if the inputted username matches a username in the database.
81          if(inputUsername == username){
82              return true;
83          }
84          // Return false otherwise.
85          else{
86              return false;
87          }
88      }
89
90      //check to see if the password entered is the correct password.
91      public boolean isPasswordCorrect(String inputPassword){
92          //Return true if the inputted password matches the password in the database and corresponds with the inputted
            username.
93          if(inputPassword == password){
94              return true;
95          }
```

```
96          // Return false otherwise.
97          else{
98              return false;
99          }
100     }
101
102     //Function: Add and remove a friend
103     public updateFriendList(int update, String friendName){
104         // If update = 1, then add friendName to friend list.
105         if(update == 1){
106             friendLst.add(friendName);
107         }
108
109         // If update = 0, then remove friendName from friend list.
110         if(update == 0){
111             friendLst.remove(friendName);
112         }
113     }
114 }
```

RETAPPED: REQUIREMENTS (FLAVOR PROFILE)

User Story: Flavor Profile

I AM A USER WHO WANTS TO BE GIVEN BEVERAGE SUGGESTIONS BECAUSE I WANT TO KNOW WHAT BEVERAGES I SHOULD TRY IN THE FUTURE BASED OFF WHAT I LIKE.

1.31 EACH USER SHALL HAVE THE ABILITY TO LOG WHEN THEY HAD A BEVERAGE FROM THE RETAPPED BEVERAGE DATABASE

1.32 EACH USER SHALL HAVE THE ABILITY TO RATE A BEVERAGE THAT THEY ARE LOGGING

1.4 EACH USER SHALL HAVE A "BEVERAGE HISTORY" THAT CATALOGUES ALL OF THEIR LOGGED BEVERAGE AND THE DATE THAT THEY WERE INPUT

RETAPPED: REQUIREMENTS (FLAVOR PROFILE)

1.5 Each user shall have the ability to add preferred beverages to a list of preferences

1.6 The Retapped system shall develop a "Flavor Profile" for each user based on their logged beverages and preferences

2.1 The Retapped system shall recommend beverage options to users based on their Flavor Profile

FLAVOR PROFILE CODE

```java
//check the score of the user's rating
//if it is high, add points to the beverage type's score
//if it is low, remove points from the beverage type's score
public void checkScore(){

    //if the score is high, add points to the beverage type
    if(this.userRating >= 7)
        flavorProfile.addPoints(beverageType:null, this.userRating);
    else if(this.userRating <= 4)
    flavorProfile.removePoints(beverageType:null, this.userRating);
```

```java
public void addPoints(BeverageType beverageType, float rating){

    //variable to determine what to add to the score
    float scoreAddition = 0;

    //if the rating is 7, add one to the score
    if(rating >= 7 && rating < 8)
        scoreAddition = 1;

    //if the rating is 8, add 2
    else if(rating >= 8 && rating < 9)
        scoreAddition = 2;

    //if the rating is 9,10 add 3
    else if(rating >= 9)
        scoreAddition = 3;

    //check that the score will not go over 10
    float newScore = 0;
    float currentScore = beverageType.getUserScore();
    if((currentScore + scoreAddition) > 10)
        newScore = 10;
    else
        newScore = (currentScore + scoreAddition);

    //set the score
    beverageType.setUserScore(newScore);
```

```java
public void removePoints(BeverageType beverageType, float rating){

    //variable to determine what to add to the score
    float scoreSubtraction = 0;

    //if the rating is 3-4 , subtract one from the score
    if(rating <= 4 && rating > 3)
        scoreSubtraction = 1;

    //if the rating is 2-3 , subtract one from the score
    else if(rating <= 3 && rating > 2)
        scoreSubtraction = 2;

    //if the rating is 0-2 , subtract one from the score
    else if(rating <= 2)
        scoreSubtraction = 3;

    //check that the score will not go over 10
    float newScore = 0;
    float currentScore = beverageType.getUserScore();
    if((currentScore - scoreSubtraction) < 0)
        newScore = 0;
    else
        newScore = (currentScore - scoreSubtraction);

    //set the score
    beverageType.setUserScore(newScore);
```

RETAPPED: REQUIREMENTS (SEARCH)

User Story: Search

I AM A USER WHO WANTS TO SEARCH FOR A BEER TO ADD TO MY PROFILE TO RATE AND FURTHER DEVELOP MY FLAVOR PROFILE

2.2 THE RETAPPED SYSTEM SHALL SHOW AN AVERAGE RATING FOR EACH BEVERAGE BASED ON RATINGS FROM USERS WHO HAVE LOGGED AND RATED IT

2.5 THE RETAPPED SYSTEM SHALL CONTAIN A SEARCH SYSTEM ALLOWING USERS TO SEARCH FOR BEVERAGES IN THE RETAPPED BEVERAGE DATABASE

SEARCH BAR CODE

```
    WRose226
    @GetMapping("/grabABeer")
    public List<Beer> grabABeer() { return service.grabABeer(); }
}
```

```
import axios from 'axios';

// default URL from the controller to pass to axios
const BEER_API_BASE_URL : string  = 'http://localhost:8080/grabABeer';

// creates a search bar that calls GET to grabABeer using axios. Displays as a table
1+ usages  Bruk04
class beerFetchingService
{
    1+ usages  Bruk04
    getBeer() : any
    {
        return axios.get(BEER_API_BASE_URL);
    }
}

1+ usages  Bruk04
export default new beerFetchingService();
```

```
1   import ...                                                          2
3
    1+ usages  Bruk04 +1
4   export default function SearchBarComponent() {
5
6       const [beers : any[] , setBeers] = useState( initialState: []);
7
8       useEffect( effect: () : void  => {
9           getBeer();
10      }, deps: []);
11
        1+ usages  Bruk04
12      const getBeer = () : void  => {
13
14          beerFetchingService.getBeer().then((response) : void  => {
15              setBeers(response.data)
16              console.log(response.data);
17          });
18      };
19
```

RETAPPED: REQUIREMENTS (MAP)

User Story: Map

I AM A USER WHO WANTS TO FIND WHERE I CAN BUY A SPECIFIC BEER BECAUSE IT IS HIGHLY RATED ON RETAPPED, OR WAS RECCOMENDED BY A FRIEND

2.4 THE RETAPPED SYSTEM SHALL ALLOW USERS TO FIND STORES NEAR THEM WITH SPECIFIC BEVERAGES IN STOCK USING LOCATION SERVICES

3.3 THE RETAPPED SYSTEM SHALL USE LOCATION SERVICES TO LOCATE A USER IF THEY ALLOW IT

MAP CODE – DEFAULT DISPLAY

```
/* CONST VARIABLES */
const googleMapsApiKey = process.env.REACT_APP_GOOGLE_MAPS_API_KEY // Receive
api from .env file in src/main/frontend
const center = { lat: 39.77989349393462, lng: -84.06510220325708 } // default
location to WSU
```

Node.js

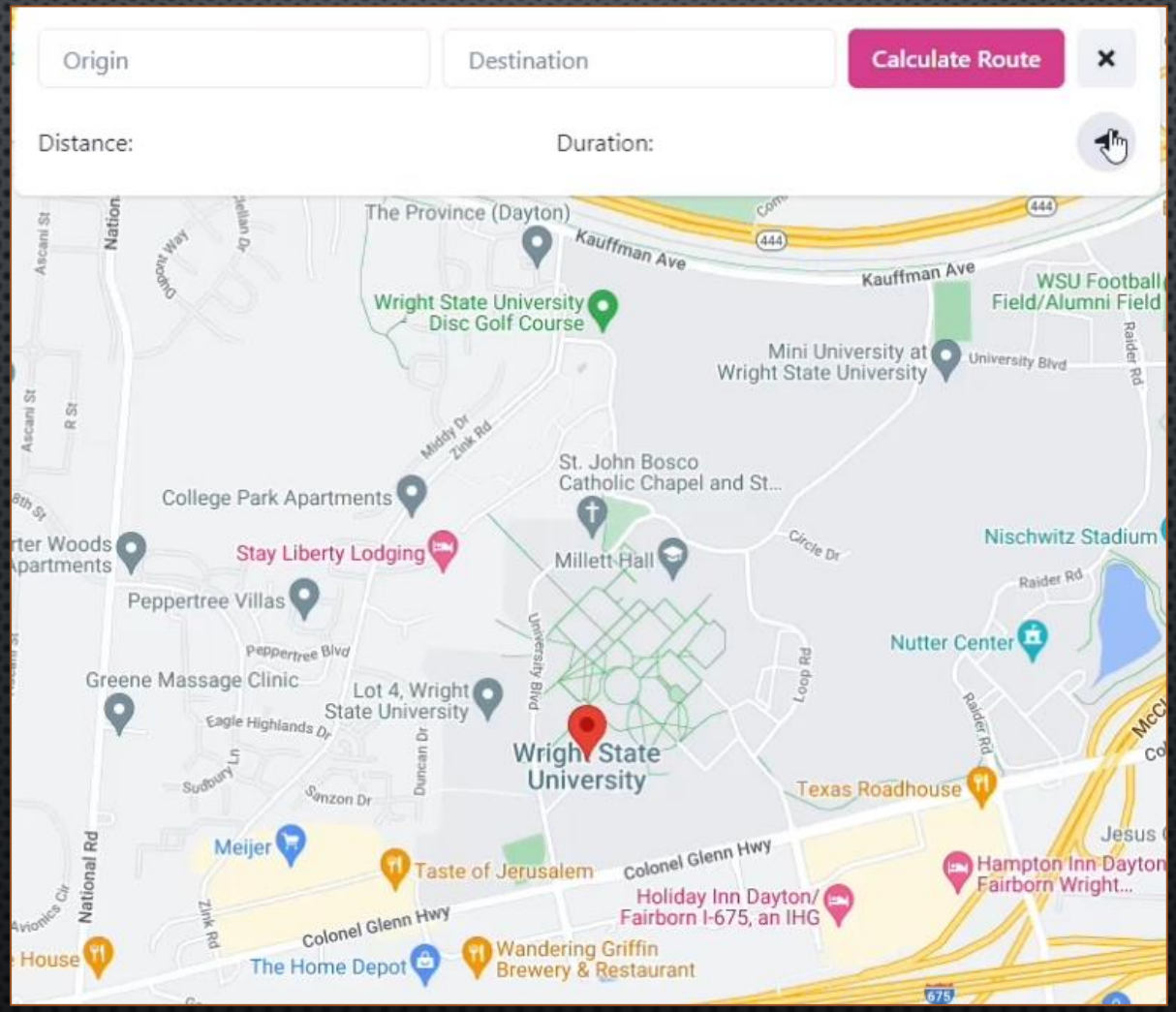
```
/* MAIN COMPONENT */
function App() {
  // Loads in google maps script to display map with libraries
  (react-google-maps/api)
  const { isLoaded } = useJsApiLoader({
    googleMapsApiKey: googleMapsApiKey,
    libraries: ["places", "routes", "core", "maps", "geocoding", "marker"],
  })

  // React hook: Initial Map State
  const [map, setMap] = useState(null)
  const [vendors, setVendors] = useState([]) // Initailize vendors state

  // Ensure the script and map are loaded and calls fetchNearbyVendors
  useEffect(() => {
    if (isLoaded && map) {
      fetchNearbyVendors()
    }
  })
```

@react-google-maps/api // @React

```
/* UI LAYOUT */
return (
  <Flex
    position="relative"
    flexDirection="column"
    alignItems="center"
    h="100vh"
    w="100vw">
    <Box position="absolute" left={0} top={0} h="100%" w="100%">
      {/* Google Map Box */}
      <GoogleMap
        center={center}
        zoom={15}
        mapContainerStyle={{ width: "100%", height: "100%" }}
        options={{
          zoomControl: false,
          streetViewControl: false,
          mapTypeControl: false,
          fullscreenControl: false,
        }}
        // Set Map to the center variable Lat/Long
        onLoad={(map) => setMap(map)}>
        <Marker position={center} />
        {directionsResponse && (
          <DirectionsRenderer directions={directionsResponse} />
        )}
      </GoogleMap>
    </Box>
    <Box
      /* Interactive Box */
      p={4}
      borderRadius="lg"
      m={4}
      bgColor="white"
      shadow="base"
      minW="container.md"
      zIndex="modal">
```

@chakra-ui/react

MAP CODE – DEFAULT DISPLAY

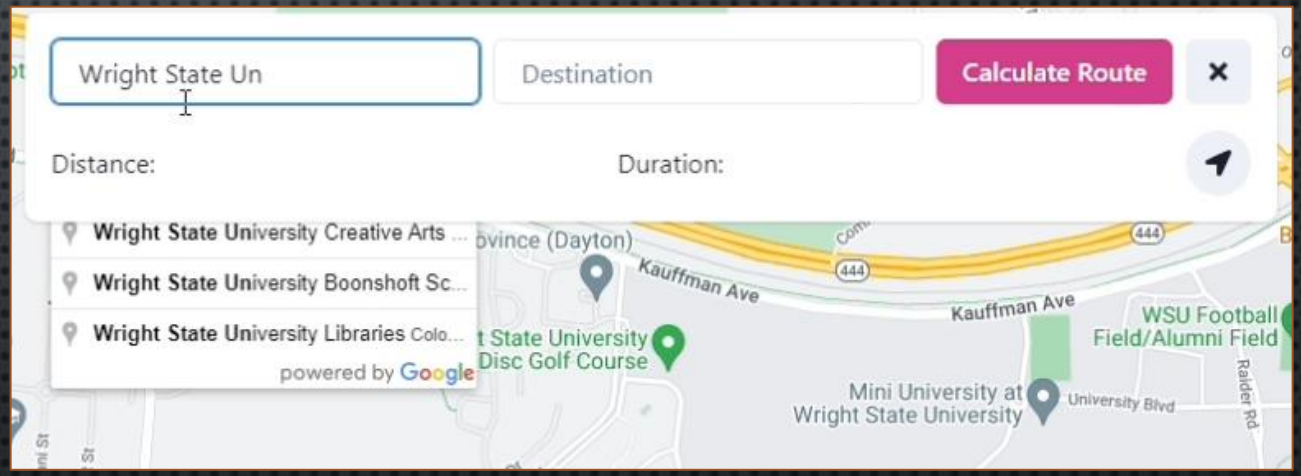


Output

MAP CODE - AUTOFILL

```jsx
<Box flexGrow={1}>
  <Autocomplete>
    <Input type="text" placeholder="Origin" ref={originRef} />
  </Autocomplete>
</Box>
<Box flexGrow={1}>
  <Autocomplete>
    <Input
      type="text"
      placeholder="Destination"
      ref={destinationRef}
    />
  </Autocomplete>
</Box>
```

@chakra-ui/react // @react-google-maps/api (Places)



Output

MAP CODE - ROUTE

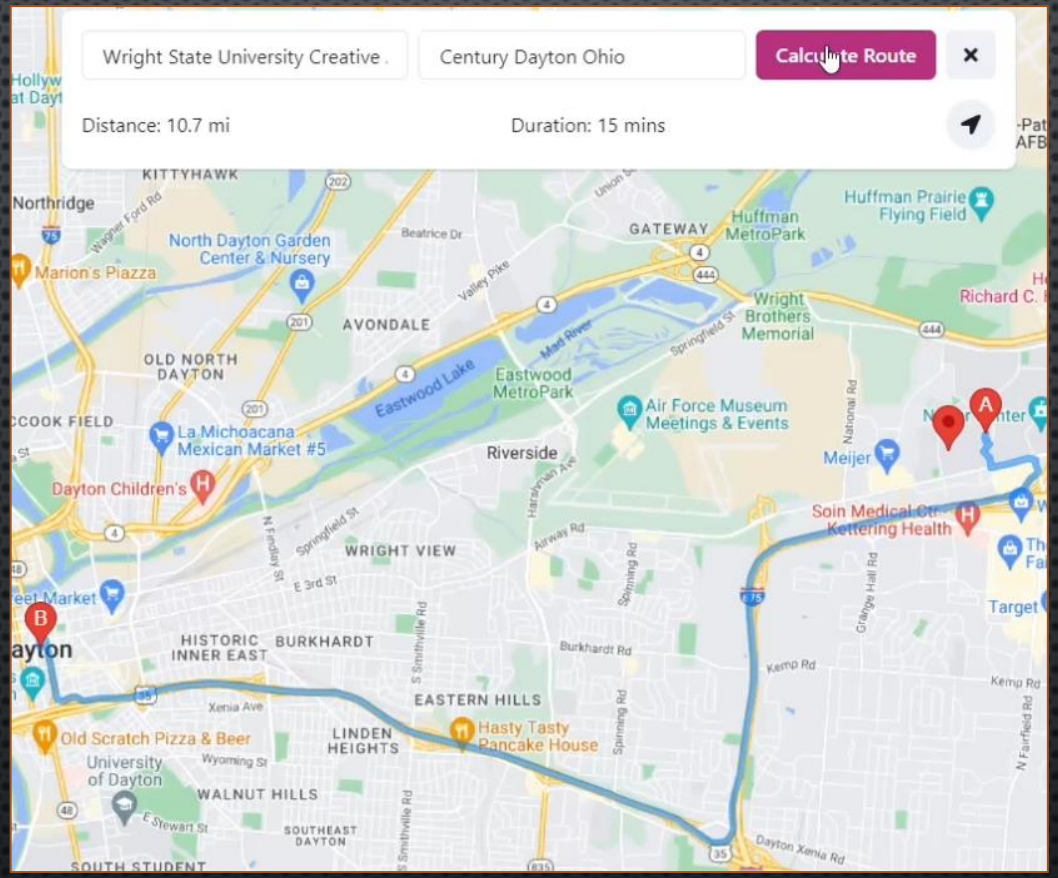
```
async function CalculateRoute() {
  if (originRef.current.value === "" || destinationRef.current.value === "")
  {
    return
  }
  // eslint-disable-next-line no-undef
  const directionsService = new google.maps.DirectionsService() //
  Initialize directionService object using Gmaps API
  const results = await directionsService.route({
    // wait for directionService and then Initialize results
    origin: originRef.current.value,
    destination: destinationRef.current.value,
    // eslint-disable-next-line no-undef
    travelMode: google.maps.TravelMode.DRIVING, // Mode: Walk, Bu
    Etc.
  })

  // Update the states of each hook
  setDirectionsResponse(results)
  setDistance(results.routes[0].legs[0].distance.text)
  setDuration(results.routes[0].legs[0].duration.text)
}
```

@react-google-maps/api (Directions)

```
{/* Calculate route button*/}
<ButtonGroup>
  <Button colorScheme="pink" type="submit" onClick={CalculateRoute}>
    Calculate Route
  </Button>
  <IconButton
    aria-label="center back"
    icon={<FaTimes />}
    onClick={ClearRoute}
  />
</ButtonGroup>
/HStack>
HStack spacing={4} mt={4} justifyContent="space-between">
<Text>Distance: {distance} </Text>
<Text>Duration: {duration} </Text>
```

@chakra-ui/react

MAP CODE - ROUTE



Output

RETAPPED: REQUIREMENTS (DATABASE)

User Story: Database

I AM A USER WHO WANTS TO ACCESS THE DATABASE SO I CAN STORE AND ACCESS MY ACCOUNT AND BEVERAGE INFORMATION

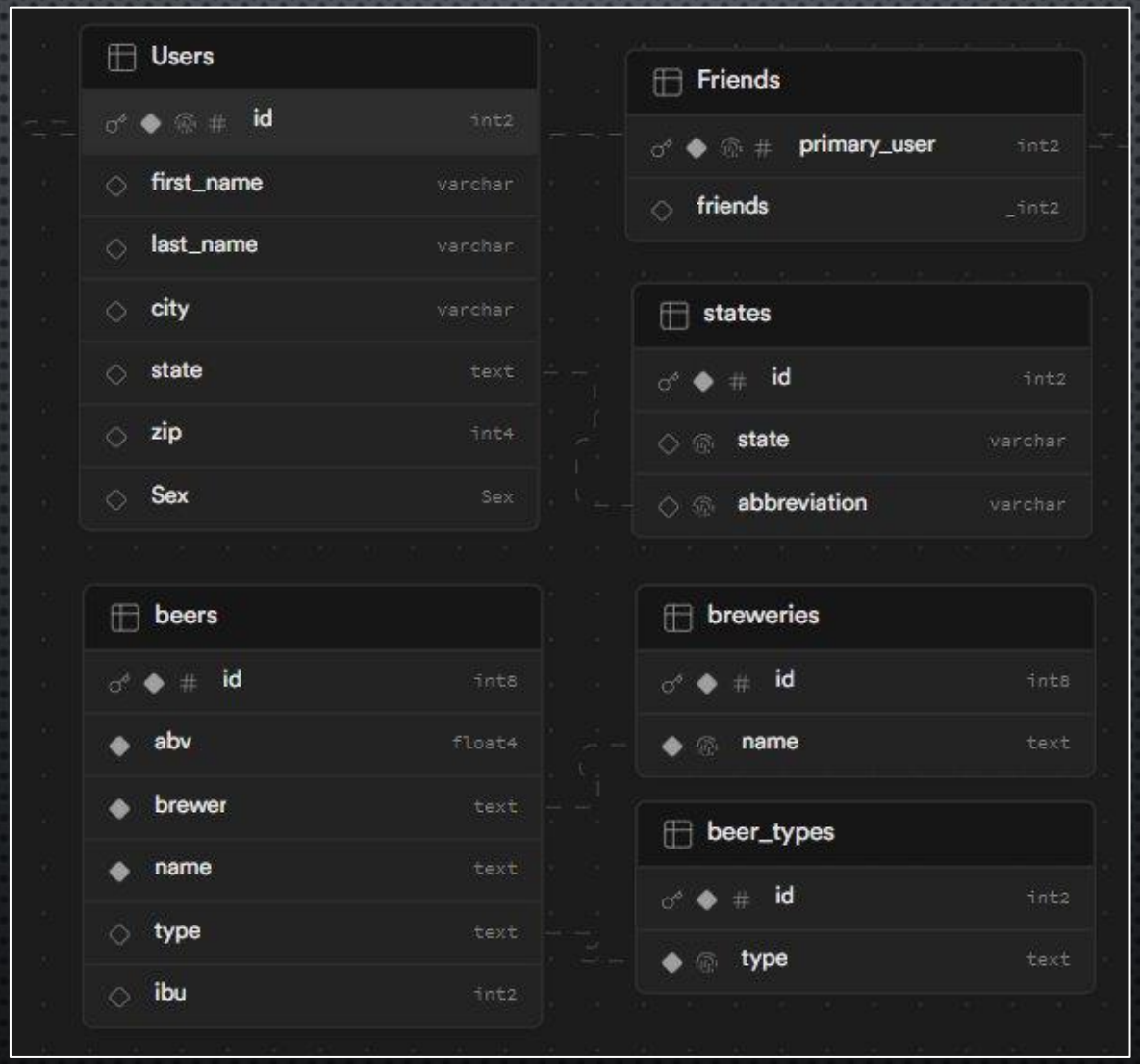
3.1 THE RETAPPED SYSTEM SHALL HAVE A WEB SERVER THAT USES DATA FROM AND STORES DATA IN A DATABASE

3.21 THE RETAPPED SYSTEM SHALL STORE USER ACCOUNT INFORMATION, INCLUDING IN THE DATABASE

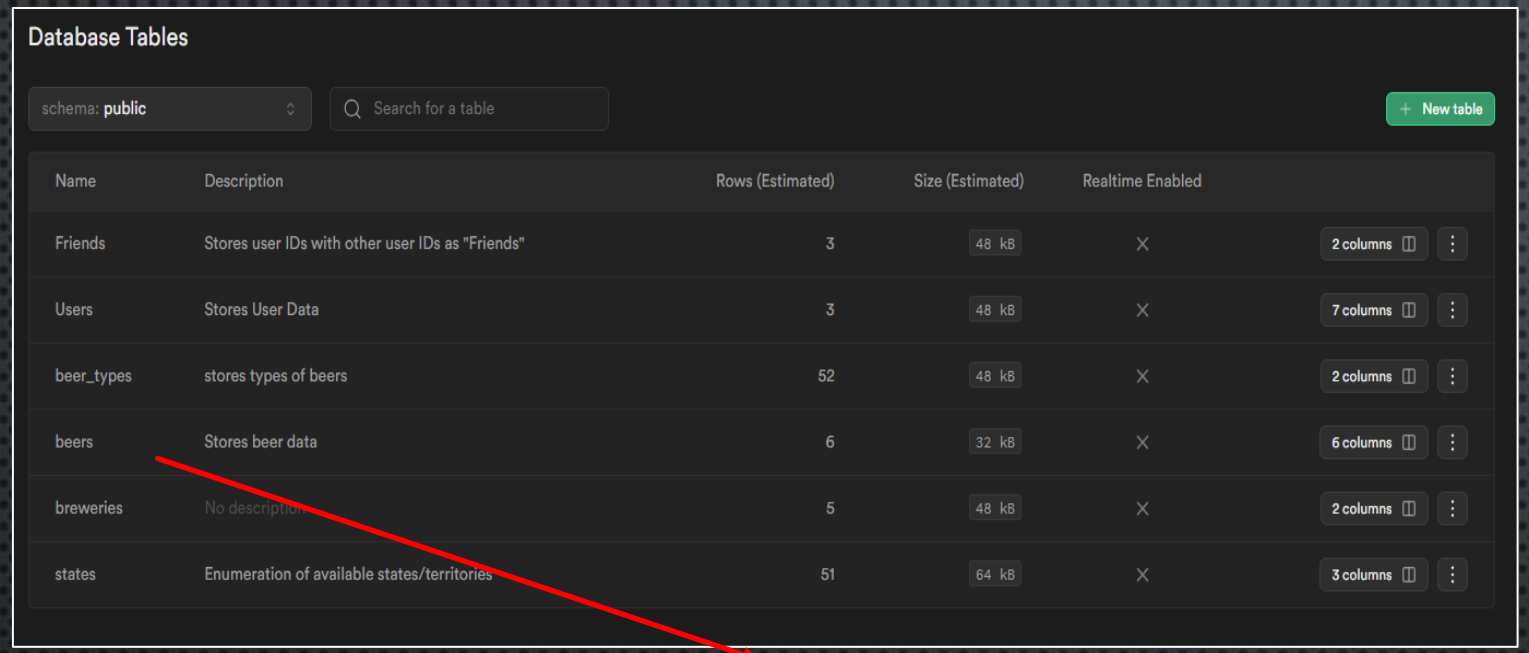
3.22 THE RETAPPED SYSTEM SHALL STORE BEVERAGE INFORMATION IN THE DATABASE

3.23 THE RETAPPED SYSTEM SHALL STORE BEVERAGE TYPES IN THE DATABASE FOR USE IN CREATING A FLAVOR PROFILE FOR A USER AND MAKING BEVERAGE RECOMMENDATIONS TO A USER

DATABASE - SCHEMA



DATABASE - TABLE



Beers Table:

id int8	abv float4	brewer text	name text	type text	ibu int2
1	4.1	Anheuser-Busch	Busch Light	Lager	5
2	6.5	Rhinegeist	Juicy Truth	India Pale Ale	75
3	4.5	Rhinegeist	Beer For Humans	Easy Hop Ale	7
4	4.8	Rhinegeist	Cheetah	Lager	6
5	4.2	Rhinegeist	Cincy Light	American Lager	NULL
6	7.2	Rhinegeist	Truth	India Pale Ale	75

RETAPPED: DEMO


RETAPPED

QUESTIONS?

References

- HTTPS://WWW.NPMJS.COM/PACKAGE/@CHAKRA-UI/REACT
- HTTPS://WWW.NPMJS.COM/PACKAGE/@REACT-GOOGLE-MAPS/API
- HTTPS://CONSOLE.CLOUD.GOOGLE.COM/
- HTTPS://SUPABASE.COM/
- HTTPS://REACT.DEV/
- HTTPS://MAVEN.APACHE.ORG/