BARFLY APP PROPOSAL

Malcolm Cusack

Lealem Amedie

Table of Contents

Introduction

The modern-day bar is a fascinating crossroads of two long-lasting human practices brought together by one custom. The first practice is the consumption of alcohol and alcoholic beverages. We've done it for centuries and we'll likely continue to do so. The second practice is that of the taverner, whose living was made by offering these beverages (and sometimes food and/or shelter) for payment. It's easy to see how both practices have evolved. Today, buying and drinking alcohol is a cultural and monumental staple entwined with some of the most meaningful traditions of humankind, while bartenders have become masters of their craft and elevating their field to both a science and an art form. Clearly, both have significantly evolved, but what hasn't seemed to change is how the two interact in a bar setting.

It's no secret how frustrating it can be to order a single drink from a bar. From the long wait just standing around, to fighting for the bartender's attention, while getting pushed from every direction. It can be way more tiresome than it needs to be. Bartenders don't have it easy either. They deal with customers who can't decide on an order, customers that don't give their full order but instead ask for each drink individually, and customers that flood them with questions like "What's good here?" and "What do you recommend?". On a typical crowded night, these things can add up and cause significant dents on the businesses' efficiency and productivity, likely leading to either the bartender falling behind or longer lines and wait times. Don't forget that while all of this is happening, servers are still collecting drink orders and giving those tickets to the same bartender. Technology has come too far for such a casual activity to be so overwhelming, for both sides. Barfly is how we can bridge this disconnect by offering a mobile/web approach to streamline the ordering process, increasing the efficiency of the entire bar while increasing customer satisfaction.

Background

There has always been a need for a product like Barfly. The general problem with the customary service structure of bars is that none of the parties involved are fully catered to by it. Bars, their employees, and their customers all have different grievances to air with the current system. Bars themselves are a very hard business to sustain. According to Statista, the number of businesses in the bar and nightclub sector in the country has been decreasing since 2011 by an average of 1.3% yearly and all in a period of economic growth and expansion.[8] Their costs are rising faster than their

revenues so the margin of error for mom-and-pop owned bars is very slim.[1] The smallest increase in sales or cutting of costs could be the difference between the business staying afloat or sinking. When customers forget to pay, when they leave because the line is too long, or when employees botch an item's price and undercharge for it, these are all losses that some establishments can't afford. Now, with the devastation that Covid-19 has brought, the bar industry isn't expected to fully recover until 2026 according to IBISWorld.[3] With the added competition of at-home alcohol consumption, customers have seen the brighter side and know how effortless the process can be. If bars want to attract the number of clients that they were before the pandemic, their service structure will likely have to adapt.

For bar employees, it's all about performance. Bar employees and bartenders especially have all been trained to prioritize two things: speed and efficiency, both extremely important in a bar setting for reasons that are clear. No matter how these employees perform, there is one constraint that neither their talents nor training can overcome: the customer. Understandably, the customer's primary goal is their own self-enjoyment whereas employees are focused on speed and efficiency to encourage mass and not individual customer satisfaction. Since their goals here are unaligned, the burden to walk the incredibly thin line of keeping individual customers happy while keeping that bigger picture in mind falls on employees. Customers will ask questions when the answers are right in front of them. They will wait until they're at the front of the line to decide what they want, and they will ask for a recital of all 25 beers available and their costs, just to order a bud light. They will wait until the bartender finishes making their drink before listing out the rest of their order or they'll order as a group and pay separately. No matter how swamped they are, it is the job of the employees to accommodate these needs. In an efficient service structure, these needs wouldn't exist. Today's structure brings problems of all sizes to bar employees. They range from intense stresses to bothersome nuisances. Having 15 people shouting in your face is definitely a lot of pressure. And having to look through the 15 debit cards you're holding whenever a customer wants to tab out is definitely a nuisance. U.S. News & World Report found bartending to be the 4th most stressful job in the country in 2017, so some alterations to how they fulfill their responsibilities would be more than appreciated.[10]

As previously mentioned, the customer's primary goal is their own self-enjoyment. Unfortunately, the current service structure requires the customer to make sacrifices to achieve their goal, mainly in the form of long wait times for ordering. A study done by Tail Cocktails, a premium cocktail brand, found that the average UK bar consumer spends nearly 12 hours of a year waiting for their orders.[4] This wait takes away from the customer's valued time and plan. Customers also perceive wait times as much worse than they actually are, if their primary task at hand is waiting itself (as is the

case of waiting at a bar).[7] This can lead them to have negative opinions about an establishment. Another downfall for a consumer is the ease of losing one's friends in the crowd when placing an order or missing out on a desired location due to an overcrowded establishment. As far as payment goes, traveling to a bar just to find out you forgot your wallet can be a 'going out' night buzz kill. Luckily, most do not forget their phone and modern smartphones have wallet integration which allows for mobile/web-based ordering, exterminating the need for a physical wallet and giving customers the option to leave it at the safety of their homes. Being skipped in line or the total lack of one in bars is another commonly occurring annoyance. Customers may come across others who speak louder or wave cash in the air to get their drink before them. This service structure seems unfair with an easy fix. Consumers also want to explore different types of libations, to maximize what they get for their dollar, but bartenders don't always have the time to lend a hand nor do many bars have full drink menus after dinner hours. A mobile menu would allow bar-goers to explore everything an establishment has to offer.

Clearly, all parties in the bar setting face the drawbacks of the current system, but somehow, the industry-wide status quo has persisted. However, the market has seen some attempts to solve these problems. Ontapp is a mobile application developed in England that functions as an in-venue ordering app.[6] The app allows for a streamlined transaction between the bar and the customer effectively, addressing some of the key concerns just discussed. The app itself however, is not available in the US and various other countries. It also assigns more work to the bars as they are charged with delivering the customers' orders to their respective locations (in contrast to how Barfly will have set order pickup spots). Customers themselves are more likely to prefer Barfly since they wouldn't be forced to enter their payment information to browse bars and their menus. Users of Ontapp must enter their payment information to create an account to access any of the app's services. Customers are also more likely to prefer Barfly because of the web application format of the product, whereas users would have to hesitantly install Ontapp on their device to use it.

Noble is a mobile application that allows users to order/pay for food and drinks online at registered event venues.[5] It advertises that customers can skip the line for concession stands, order from their phones, and grab their orders when ready, but the product's scope is limited to certain locations. Noble offers its services to businesses in the hospitality industry in which food/drinks are not the primary commodities such as theatres, concerts, and arenas. Its objective is to alleviate some of the responsibilities that come with serving food/drinks, so that the business can focus more on the primary service that they provide. In addition, it requires the business to buy kits and equipment before their customers are able to access Noble's services.

TendedBar is a solution that promises to integrate into any venue no matter the magnitude of the venue's needs.[9] TendedBar functions like an automated bar and the technology itself is comparable to an alcoholic vending machine unit. These units are installed according to the venues' needs and are then filled up with the venues' choice of alcohol. Customers can then walk up to units, place their order, and watch it pour into their cups. TendedBar advertises that their technology can take pressure off of venue staff during high volume shifts and as well as having a maximum transaction time of 5 minutes for the customer. However, this solution could be expensive for many venues as they must pay for the units up front. As for the customer, the notion of lines and waiting still exists in this use-case as customers have to line up to use the units. One could argue that the venue could simply buy more units, but again that could prove to be too costly for them.

Drizly is an app that lets users get various types of alcohol delivered to their home address.[2] Instead of changing a bar's service structure, Drizly eliminates it entirely. Its selling point is that users can order their drinks from the comfort of their own homes. However, the scalability of this service is limited by the availability of delivery drivers and its scope is limited to less than 30 states as the remaining territories do not allow for alcohol delivery. Not to mention that it holds no long-term applicability as it deprives the customer of the bar environment. DoorDash, GrubHub, and UberEats are food delivery apps that fall under the same umbrella of Drizly's use case as well as limitations.

Some bars implement their own software that acts as their point of sale and service. They make use of QR codes and electronic waiting lists to provide a more seamless and practical digital experience for the customer. For the less tech-savvy bars that can't afford to outsource producing the software, this is not a very realistic route to go. The countless mom-and-pop owned bars face the threat of being left in the dust as the rest of the industry makes these technological advances.[1] This isn't very convenient for the customer either as they don't have a centralized system that consolidates all their favorite bars.

Work Schedule

At Barfly LLC, we are choosing to use AGILE Scrum as our development/ productivity lifecycle framework. Using Scrum allows us to keep track of our goal, by organizing tasks into weekly sprints (time period to finish an increment of work) that are a part of bi-weekly epics (multiple sprints). Sprints are composed of stories, which are sprint goals. Each story is a TODO that is cleared when its definition of done is

completed. Sprints also have backlogs that are filled with uncompleted stories. Stories from the backlog need to be completed with higher priority in future sprints. At the beginning of each epic and sprint, the Barfly team will have a planning session to map out all stories necessary for completion. This planning session will also be our weekly standup. Weekly stand-ups are a team check-in where members ask questions, update each other on current work status and stay accountable. There will be a retrospective at the end of each epic where the team will reflect on the epic's sprints, ensuring each goal has been completed.

Epics and sprints will consist of front-end, backend, and middleware development. Everyone in the group will be responsible for different tasks during each sprint. Each member will contribute to all aspects of the project. At the end of each epic, we will conduct testing to make sure development has been successfully completed. Below is a table depicting each of our epic goals and their deadlines.

Epic Description	**Start Date**	**End Date**	**Duration**
Creating initial database & query testing, front end skeletons for both apps, creating connection of orders between both apps.	1/24/21	2/7/21	14 days
Authentication for both apps, including distinguishing between Admin, employees and users. Implement Payment protocols.	2/7/21	2/21/21	14 days
Implementing order queue, nightly summary, employee order selection, bar creation steps (import data, menu, location, employees, admin ect. Bar app UI refinement.	2/21/21	3/7/21	14 days
Implementing location services, QR code logic, ability for users and employees to upload profile photos, create store of photos for menu items. Refine user app UI.	3/7/21	3/21/21	14 days
Creating bar layout and spot selection logic. Create custom drink/ food orders ability (tie in mixers/liquors)	3/21/21	4/4/21	14 days

Create the ability to scale bar apps to multiple bars. Each with their own db and moderately custom front end. Each bar goes through the ‘setup process.’	4/4/21	4/18/21	14 days
Beta test app with bars, the public, friends and family. Diligent Testing	4/18/21	4/25/21	7 days
Improve both apps from feedback given from earlier epic. Could include new features, eliminate features, different UI/UX.	4/25/21	5/2/21	7 days
Full app demo	5/2/21	5/9/21	7 days

Proposal Statement

Barfly will allow users to order a drink or food from the establishment directly through their phones. By accessing a specific establishment by scanning a QR code, the user will be directed to a specific bar’s web page. The site will depict general information about the bar and display a categorized menu. This menu will consist of predefined drinks and food set by that specific bar and once a user selects then creates an order, they will be sent to a checkout page and choose where in the bar they’d like to pick up their order. Once an order has been submitted and payment is confirmed, the user will be sent a confirmation. When the order is ready, the user will be notified and can proceed to pick it up at the selected spot. The process is simple and eliminates nearly all the opportunity cost of placing an order at a bar by cutting out the unnecessary clutter associated with it.

Bars/restaurants will have to have their own enterprise version of Barfly (Barfly-E). Once a bar signs up for our service, it will undergo a quick setup process and create or fill in the bar’s name, menu items, payment method, staff info, bio, and other optional details. Once completed, the bar’s page will be automatically generated with their data. The main page will be a queue of order tickets to be completed by bartenders. Tickets will have information on what is ordered and by whom, with a corresponding confirmation number tied to whoever ordered it for the items to be picked up by the customer. Orders can be claimed by the employee selecting an outstanding customer’s ticket. Then, employees just update the order with the current status associated with it. At the end of the night, the tips will be forwarded to the assigned

employee or mathematically pooled between all employees, depending on the bar's configuration settings. This system allows for bars to receive optimum customer satisfaction rates by limiting the wait times customers perceive even if the time to make the order itself stays constant, a phenomenon The Washington Post has examined thoroughly. It also provides them with exposure and opportunities to gain new clientele.

Functional Requirements

Barfly Functional Requirements

Category	**Requirement**	**Priority**
Authentication	Account creation User login functionality Continue as guest option	Must Must Should
Bar Selection	Bar search functionality QR code scanning Bar selection Location-based Bar selection	Must Should Should
Bar Info Display	Name, Bio Photos	Must Should
Order Selection & Checkout	Display interactive Menu with selectable entries Add/remove selected orders to/from cart Display order summary and cost Allow for order revision and editing Payment functionality Select where to pick up order Communicate order to Barfly-E	Must Must Must Must Must Could Must
User settings	Display current profile, info, preferences Profile, password, preference update functionality	Must Should

Barfly-E Functional Requirements

Category	Requirement	Priority
Authentication	Account creation User login functionality	Must Must
Bar Setup	Menu, staff credentials & bio creation Bar QR code generation	Must Should
Order Queue	Queue of incoming orders to be completed by employees. Choose order → start process → complete order → deliver/handoff order.	Must
Confirmation	An order will display a random phrase, an ID and photo of who is picking up the order.	Must
Settings	Ability to change menu, bar layout, credentials, employees, payment, and ect…	Must
Analytics	Nightly, Monthly, yearly summaries of bar monetary performance and inventory consumption.	Should

Non-Functional Requirements

Requirement	Priority
Usability (Easily accessible services)	Must
Marketability (Good design, logo, aesthetic)	Must
Insightful (little to no learning curve)	Must
Scalable (both vertically and horizontally)	Must

Accessibility (available on/for different platforms)	Must
Compliance (FDIC regulations, NIST guidelines)	Must
Adaptability (Suitable for different establishments)	Should
Extendability (Allow for easy addition of new features)	Should
Robustness (Operates well even with some missing data)	Could
Regular Testing	Could
Redundancy (Backup the data)	Could

Performance Requirements

Because of the nature of the industry that Barfly exists in, overall speed and efficiency are prioritized objectives. And the nature of the service industry aside, customers know just what they're looking for in a digital experience and have gotten used to their high expectations being met. For Barfly to meet these expectations, firstly, it must have shorter load and user-response times. The app must be optimized for fast API calls as well as frequently expected queries and data access patterns. Caching should also be implemented to reduce user latency. Another requirement for Barfly is a lower memory consumption and a minimal application footprint. It must restrain the amount of physical memory it occupies by its optimized codebase and limit its number of nodes. Furthermore, these criteria must be addressed roughly consistently throughout platforms, networks, and even various load conditions.

Interface Requirements and Tools to be Used

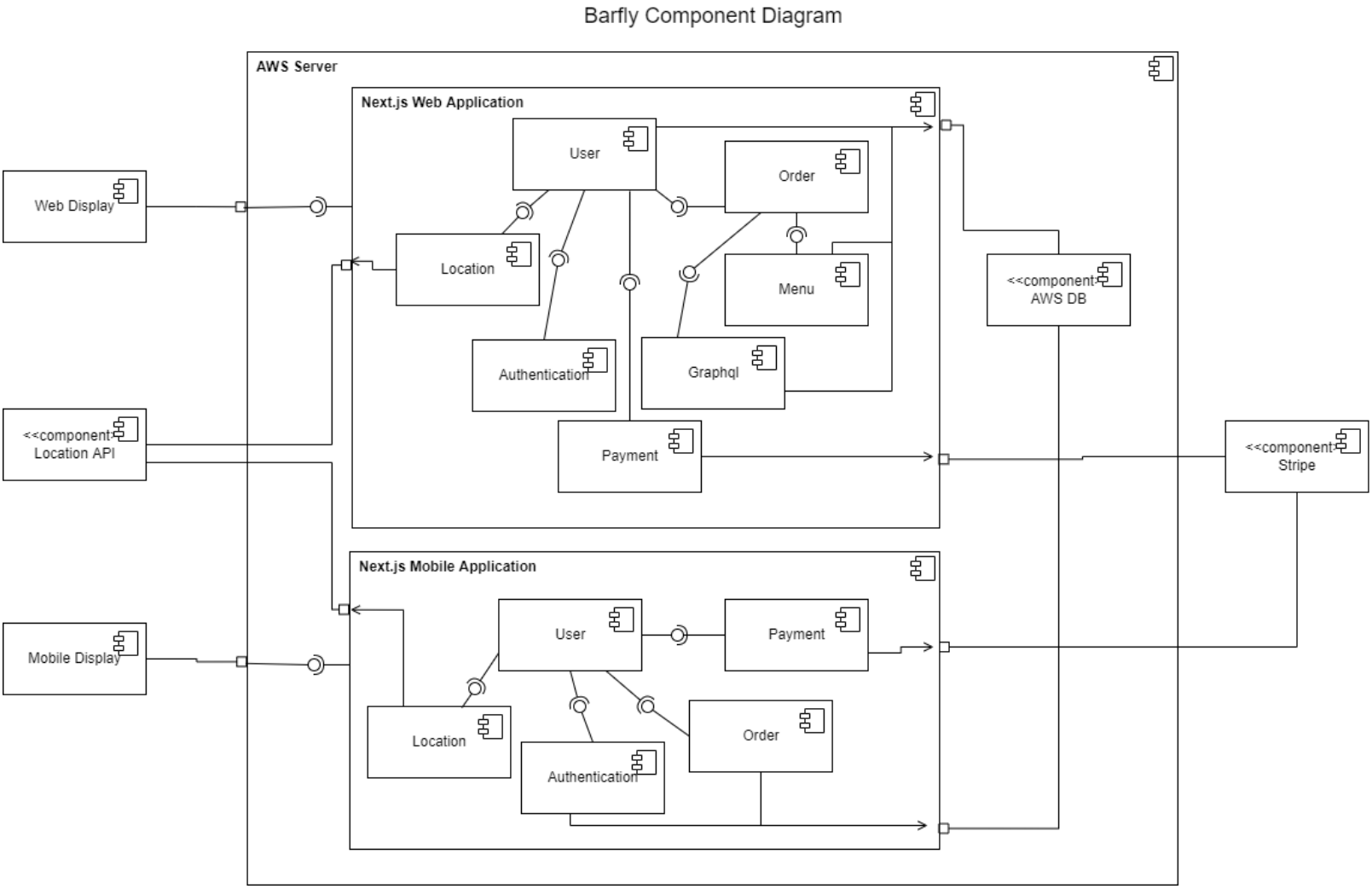
Barfly has quite a modern stack and interfaces with a good deal of technologies. The front-end stack includes both a React.js and React Native app. React.js allows developers component and state-based web development. Writing React involves writing JSX (javascript based HTML), programming using javascript or typescript, and styling using CSS or SCSS. React Native supplies the ability to write a codebase for both IOS and Android native apps. Native React uses a different version of JSX that compiles into native code. Many web apps involve lots of state management. We will

accomplish state management by using React Context API, which creates a data layer for the whole front end to access. The plan is to enclose both in a Next.js app, which allows for faster app performance, by running immediately on the client-side browser. React apps are created with Node.js, this is a great advantage because it's very easy to import node packages that skip the need to reinvent the wheel. Barfly will be interfacing with Stripe API. Stripe is a payment platform with a uniquely amazing API to interface with. Stripe allows for accepting payments and will be implemented with both apps. We will also interface with Google's location API to locate where a user is. This is very important for directing an end-user to a specific bar's website. Both apps will be using materialUI, this is a React component library that eases implementation and design costs. MaterialUI library has many types of components from styled buttons to full tables that are easy to implement and integrate with our data and standing components. Testing Barfly can be automated using the selenium library. React by default creates a single page application. We are implementing a library called React Router Dom which allows our app to have as many pages as we desire. For example /menu/beer or /signup would be page examples.

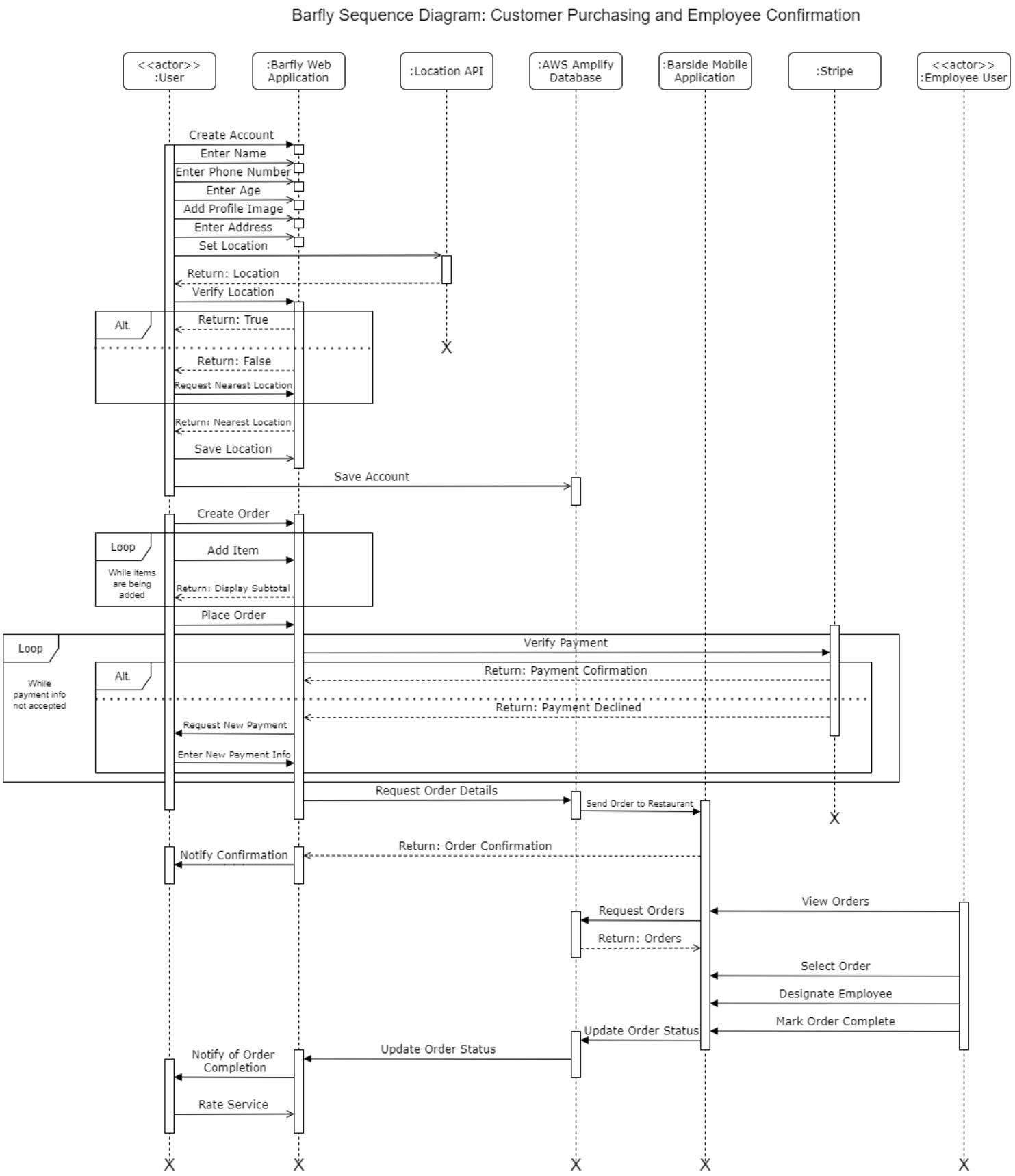
Barfly's backend duties are carried by AWS Amplify. Amplify is a serverless backend suite that is coupled with dynamoDB, authentication, storage, graphQL querying, CI/CD, Lambda functions, automatic scaling, testing, and hosting. DynamoDB is a noSQL that stores all our data in tables that are very easy to create using a scheme file or a GUI. Amplify authentication takes care of many security issues. The feature also allows for a GUI to manage all users and permissions throughout the site. Storge is much like dynamoDB, but solely for larger files like images and videos. The storage feature is great as it speeds up querying. GraphQl queries can be accomplished on the front-end the same way any web server would. GraphQl has advantages over SQL as you can very easily subscribe to a specific table. For example, both our apps will need to subscribe to any change in the orders table. GraphQl is more secure and sends fewer queries than SQL. Continuous integration and deployment is an amazing feature. For example, if I push code up to our master GitHub branch, Amplify sees this and rebuilds and deploys both the current frontend and backend version of the app. The lambda functions feature can be easily created and integrated for any type of computational needs. You can even set triggers on dynamoDB that will automatically run a function. Lambda functions can be written in most any language, we will program using Node.js or Python. On top of everything Amplify scales for you, if the load on AWS servers goes up, more will be allocated for Barfly. Finally Amplify comes with a free SSL certificate that links to your app's domain for deployment and hosting.

Architectural Design Documents

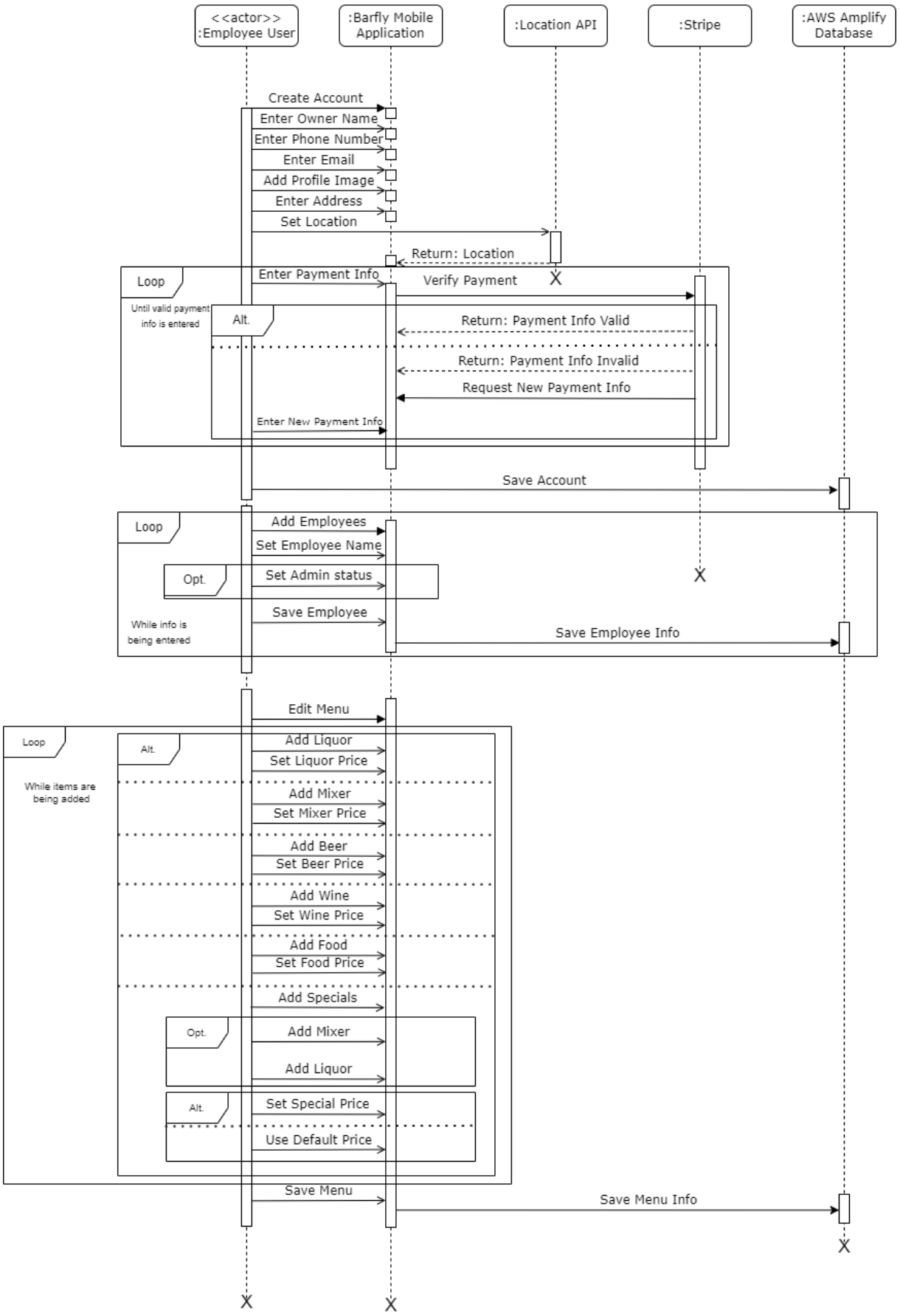
Component Diagram



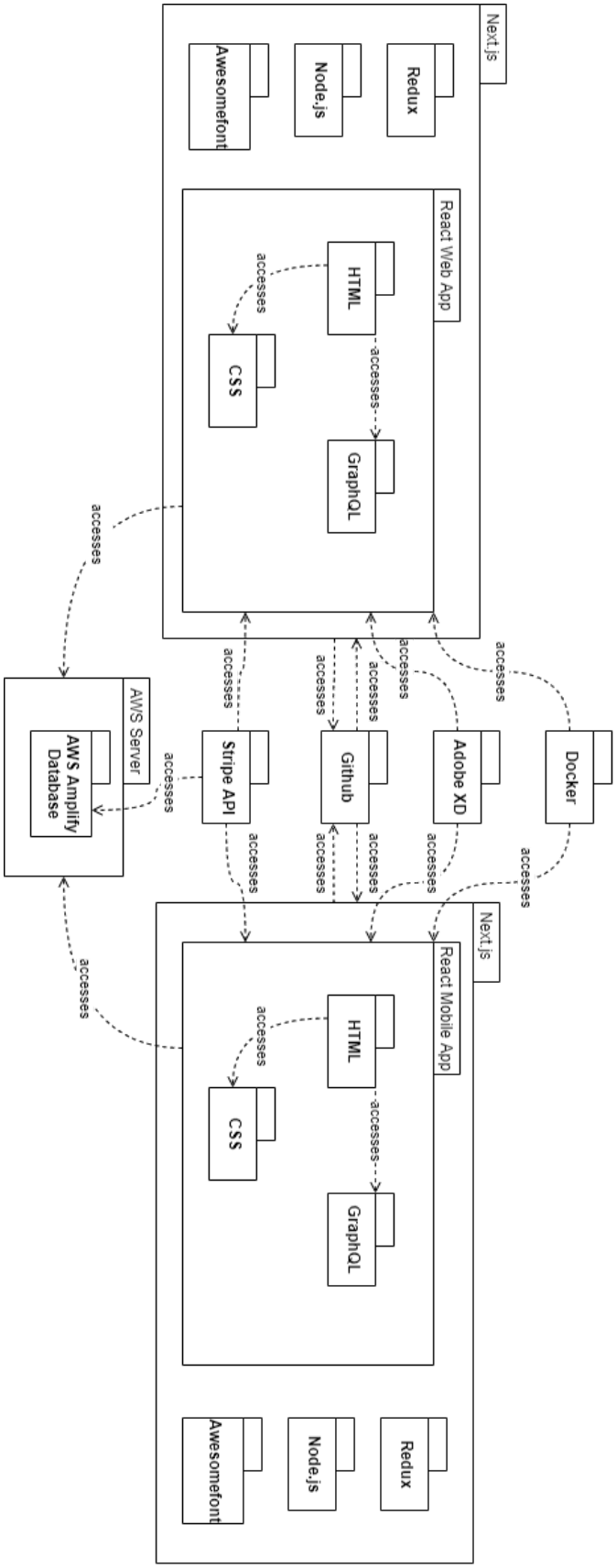
Sequence Diagrams



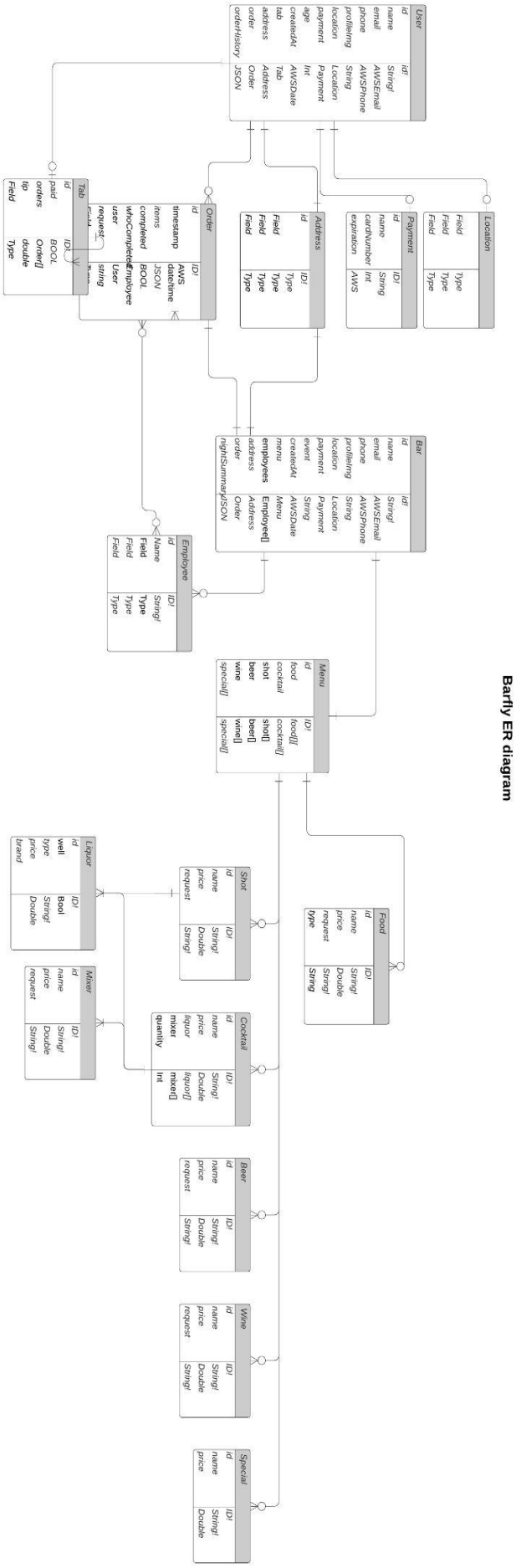
Barfly Sequence Diagram: Employee Account Creation and Bar Info Setup



Package Diagram



ER Diagram


Barfly ER diagram
User
Location
Payment
Address
Tab
Order
Employee
Bar
Menu
Food
Liquor
Shot
Mixer
Cocktail
Beer
Wine
Special

Design Patterns

The design patterns that we used for Barfly can be seen in the UML documents above. The first is the observer pattern, this pattern essentially is to have a one-to-many relationship between an object or in our case a React component and all of the observers. Web apps use this pattern all the time with the use of state in an application. For instance, if a user logs into Barfly, the entire aspect of the app that the user would see is notified of the current logged in user. Another example of state within Barfly is constructing an order of drinks and food. As soon as a user chooses a beverage, the order summary, and checkout pages state update instantly. Barfly's core functionality includes order subscription between both bar side and end user apps. Each observes the orders table and the state of each app changes when a new order is created, and completed. Resulting in notifying the user their drink is ready. Another design pattern Barfly will implement is the decorator pattern. This pattern essentially has a base class with properties that can be expanded upon using the decorator class. Barfly also uses the decorator design pattern when adding events onto existing anonymous functions. We created a decorator function called prevDef to prevent refresh upon events like submitting a form. Another example of Barfly using the decorator is within the data layer. Barfly uses React Context API that encompasses the entire app within this data layer. Giving access to global state and properties. Contex API using decorators to connect React components to data within the data layer. Additionally, Barfly will also implement the state design pattern. The state design pattern helps to program objects so that their behavior, as well as functionality, is dependent on their internal state. With Barfly, this pattern may indirectly determine how/what a user orders by deciding which bars a user can access (as some might be in an inactive state), what menu items are currently available. The pattern will also be used when users place their order, when assigning orders to employees and when reporting the status of the placed order to the user.

Design Tradeoffs

To give Barfly the qualities it needed, certain tradeoffs had to be made at the design level. Firstly, the use of AWS and its authentication allows Barfly to provide a faster and more secure login experience for users but also tightens the coupling of their system with Barfly, which wouldn't happen if Barfly had its own authentication functionality. Using AWS' server also comes with issues such as limited control and possible downtimes. Secondly, designing Barfly-E to allow bars to generate their bar layout introduces a myriad of problems as it would consume heavier hours to implement and it would likely require developers to help with the initial building of said layout but it does happen to be an essential segment of functionality. Another design trade off is the

use of a data layer compared to passing data and state from parent to child components. A data layer gives the advantage of easily accessible global variables that reduce the memory consumption of the app. Whereas passing data from parent to child can become bulky and slow the application, while also causing potential variable conflicts. The pro to passing data parent to child is easier readability and faster coding. There's no way to get around not passing data this way, but implementing a data layer is a necessary trade off for the long term scalability of the app. Barfly is using a web component library for styling called materialUI, while using this library eases design and implementation time. We don't have full customization of every aspect of our product that comes from creating every component from scratch. The team also had some trouble deciding on how authentication and certain functionality should work on Barfly-E. The first proposed method was for establishments to use just one tablet and have a single login credential for the business as a whole. This allows for a somewhat centralized usage system for the bar while also being a simpler approach, development wise. However, having just one login for the bar adds certain redundant steps that employees must go through (mostly selecting themselves as the active personnel from a dropdown list of employees) since the app won't know which user it is accommodating. The second proposed method was for employees to have separate logins and use Barfly-E from their own devices. Although it introduces some complexity in the implementation logic, the Barfly team decided on this approach because it's likely more convenient for employees as it allows for a more customized experience and avoids the problems that would come with having multiple people trying to use one device.

Results

By the end of April 2022, we successfully created both Barfly and Barfly-E apps. During the process we achieved all functional, non-functional, performance and interface requirements listed above. We were even able to show our MVP to MSU's Blackstone for additional feedback and potential next steps for making Barfly into an actual company. We are especially proud of achieving full functionality from an end user signing up and ordering a drink from Barfly. To receive that same order and complete it using Barfly-E downloaded to an Ipad. We learned a world of information creating both a React web and React native apps, complemented by a sturdy well performing graphQl API serving data from dynamodb. Building Barfly has been by far one of the best project experiences to date. We will now have a great project to showcase our skills. We plan on improving Barfly and bringing the apps to market someday soon. Our teamwork turned an idea into full fledged apps.

References

1. Dao, D. Q. (2020, June 8). *Bars must change for good after coronavirus*. Food & Wine. Retrieved November 5, 2021, from www.foodandwine.com/fwpro/bars-drinks-industry-change-coronavirus.

2. Drizly. (n.d.). *Drizly, On-demand alcohol delivery*. Drizly. Retrieved November 2, 2021, from drizly.com.

3. IBISWorld. (2021, July 23). *Industry market research, reports, and Statistics*. IBISWorld. Retrieved November 5, 2021, from www.ibisworld.com/united-states/market-research-reports/bars-nightclubs-industry.

4. Leader, A. (2019, November 11). *Long Queue Times 'drives customers away'*. The Morning Advertiser. Retrieved November 3, 2021, from www.morningadvertiser.co.uk/Article/2019/11/08/Long-queue-times-could-drive-customers-away-research-reveals.

5. Noble. (n.d.). *Noble, mobile ordering for entertainment*. Noble, the Mobile App for Ordering Drinks at Venues. Retrieved November 2, 2021, from getnoble.co

6. Ontapp. (n.d.). *Ontapp, the new in-venue ordering app for bars, clubs, cafes, restaurants and events.* Ontapp.app. Retrieved November 2, 2021, from www.ontapp.app.

7. Swanson, A. (2019, April 26). *What really drives you crazy about waiting in line (it actually isn't the wait at all)*. The Washington Post. Retrieved November 18, 2021, from www.washingtonpost.com/news/wonk/wp/2015/11/27/what-you-hate-about-waiting-in-line-isnt-the-wait-at-all.

8. Statista. (2021, February 21). *Topic: Bar and nightclub industry in the U.S.* Statista. Retrieved November 5, 2021, from www.statista.com/topics/1752/bars-and-nightclubs.

9. TendedBar. (n.d.). *Tendedbar is the first fully automated bar designed primarily for high-volume venues.* TendedBar. Retrieved November 2, 2021, from tendedbar.com.

10. Williams, G. (2017, March 12). *The most stressful jobs in the U.S. in 2017 - US News Money*. U.S. News. Retrieved November 2, 2021, from money.usnews.com/careers/company-culture/slideshows/the-most-stressful-jobs.

Appendix

Barfly Code

App.js

```
import "./App.css";
import { useState, useEffect, createContext } from "react";
import { useStateValue } from "./state/StateProvider";
import { Auth, Hub } from "aws-amplify";
import {
    BrowserRouter as Router,
    Routes,
    Route,
    Navigate
} from "react-router-dom";
import {
    ThemeProvider,
    createTheme,
} from "@mui/material";
import SignIn from "./components/auth/signIn";
import SignUp from "./components/auth/signUp";
import ChangePassword from "./components/auth/passwordReset/ChangePassword";
import Menu from "./components/menu/Menu";
import OrderSummary from "./components/order/OrderSummary";
import Payment from "./components/payment/Payment";
import OrderStatus from "./components/order/OrderStatus";
import { Box } from "@mui/material";
import { SnackbarProvider } from "notistack";
import { loadStripe } from '@stripe/stripe-js';
import { Elements } from "@stripe/react-stripe-js";
import RequestPasswordReset from "./components/auth/passwordReset/RequestPasswordReset";
import ResetPasswordPage from "./components/auth/passwordReset/ResetPasswordPage";
import Common from "./components/Common";
import ViewBars from "./components/bars/ViewBars";
import Profile from "./Settings/Profile";

export const ActionsContext = createContext(
    {} as { fetchData: () => void; signOut: () => Promise<void> }
);
console.debug("================= console.debug is enabled ===============");

// Back end push: amplify push
// Front end push: git push <branch> or origin master

const theme = createTheme({
```

```
    palette: {
      divider: "#fcba03",
      mode: "dark",
      primary: {
        main: "#fcba03",
        contrastText: "black",
        light: "#fcba03",
        dark: "#fcba03",
      },
      secondary: {
        main: "#222",
        contrastText: "white",
        light: "white",
        dark: "black",
      },
      background: {
        default: "#fcba03",
      },
      text: {
        primary: "#fcba03",
        secondary: "#eee",
      },
    },
    components: {
      MuiTextField: {
        styleOverrides: {
          root: {
            backgroundClip: "black",
          },
        },
      },
    },
});

function App() {
    const stripe =
loadStripe("pk_test_51KF9vsKLUQNKwRj5JSihMuNbCzb7bZ2GvHMwPggdS7W2fJsaoHrHXGtEamHHHD7l6DD2mIQ5gOHeB9pj4
nemhcny00qmvTpHNZ");

    const [{ user }, dispatch] = useStateValue();

    const [triggerFetch, setTriggerFetch] = useState(false);

    function fetchData() {
      setTriggerFetch(true);
    }

    async function signOut() {
      try {
        await Auth.signOut();
        dispatch({ type: "RESET_USER_DATA" });
      } catch (error) {
        console.log(error);
      }
    }

    useEffect(() => {
      let isMounted = true;

      const fetchUserData = async () => {
        if (isMounted) {
```

```
            dispatch({ type: "FETCH_USER_DATA_INIT" });
        }
        try {
            if (isMounted) {
                const data = await Auth.currentAuthenticatedUser();
                if (data) {
                    dispatch({
                        type: "FETCH_USER_DATA_SUCCESS",
                        payload: { user: data },
                    });
                }
            }
        } catch (error) {
            if (isMounted) {
                dispatch({ type: "FETCH_USER_DATA_FAULURE" });
            }
        }
    };

    const HubListener = () => {
        Hub.listen("auth", (data) => {
            const { payload } = data;
            onAuthEvent(payload);
        });
    };

    const onAuthEvent = (payload) => {
        switch (payload.event) {
            case "signIn":
                if (isMounted) {
                    setTriggerFetch(true);
                }
                break;
            default:
                return;
        }
    };

    HubListener();


    fetchUserData().then(() => {
        // load order after loading user data
        dispatch({ type: "LOAD_ORDER" });
        dispatch({ type: "LOAD_BAR"})
    });

    return () => {
        Hub.remove("auth", () => {});
        isMounted = false;
    };
}, [dispatch, triggerFetch]);

return (
    <ActionsContext.Provider value={{ fetchData, signOut }}>
        <ThemeProvider theme={theme}>
            <SnackbarProvider maxSnack={1}>
                {/* ================= Router to all the pages ================= */}
                <Router>
                    <Box className="App" height="100%" width="100%">
                        <Routes>
```

```
{!user ? (
  <>
    <Route
      path="/"
      element={
        <Common>
          <SignIn />
        </Common>
      }
    />
    <Route
      path="/signup"
      element={
        <Common>
          <SignUp />
        </Common>
      }
    />
    <Route
      path="/forgotpass"
      element={
        <Common>
          <RequestPasswordReset />
        </Common>
      }
    />
    <Route
      path="/resetpass/:email"
      element={
        <Common>
          <ResetPasswordPage />
        </Common>
      }
    />
    <Route
      path="/:barid/menu"
      element={
        <Navigate to="/" />
      }
    />

  </>
) : (
  <>
    <Route
      path="/"
      element={
        <Common>

          <ViewBars/>
        </Common>
      }
    />
    <Route
      path="/forgotpass"
      element={
        <Common>
          <RequestPasswordReset />
        </Common>
      }
    />
```

```
<Route
   path="/resetpass/:email"
   element={
      <Common>
         <ResetPasswordPage />
      </Common>
   }
/>

<Route
   path="/:barid/menu"
   element={
      user ? (
         <Common>
            <Menu />
         </Common>
      ) : (
      <Navigate to="/" />
      )

   }
/>

<Route
   path="/:barid/ordersummary"
   element={
      <Common>
         <OrderSummary />
      </Common>
   }
/>

<Route
   path="/:barid/payment"
   element={
      <Common>
         <Elements stripe={stripe}>
            <Payment />
         </Elements>
      </Common>
   }
/>

<Route
   path="/orderstatus"
   element={
      <Common>
         <OrderStatus />
      </Common>
   }
/>

   <Route
   path="/changepass"
   element={
      <Common>
         <ChangePassword />
      </Common>
   }
/>
```

```
                                <Route
                                  path="/profile"
                                  element={
                                    <Common>
                                      <Profile />
                                    </Common>
                                  }
                                />
                              </>
                            )}
                          </Routes>
                        </Box>
                      </Router>
                    </SnackbarProvider>
                  </ThemeProvider>
                </ActionsContext.Provider>
    );
}

export default App;
```

Index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import Amplify from 'aws-amplify';
import config from './aws-exports';
import { StateProvider } from './state/StateProvider';
import reducer, { initialState } from './state/reducer';

Amplify.configure(config);

ReactDOM.render(
  <React.StrictMode>
    <StateProvider initialState={initialState} reducer={reducer}>
      <App />
    </StateProvider>
  </React.StrictMode>,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

App.css

```
@import url('https://fonts.googleapis.com/css2?family=Racing+Sans+One&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Lato:wght@700&display=swap');

.App {
```

```
  text-align: center;
  color: #fcba03;
}

h2{
  font-family: 'Tahoma';
  letter-spacing: 1px;
  font-weight: lighter;
  text-align:left;
}

body{
  background-color: #333;
  font-family: 'Tahoma';
}

.bio{
  font-family: 'Tahoma';
  font-size: 15px;
  text-align: center;
}

.buttons{
  letter-spacing: 1px !important;

}

.tab{
  letter-spacing: 1px !important;
  background-color: #393939 !important;
  width: 33%;
  border: 0 !important;
}


.activeTab{
  letter-spacing: 1px !important;
  background-color: #393939 !important;
  width: 33%;
  border: 0 !important;
  border-bottom: 2px solid !important;
}
.numbers{
  margin: 0;
  margin-left: 10px;
  text-align: left;
}

input:-webkit-autofill{
  -webkit-text-fill-color: black !important;
}


.App-logo {
  height: 40vmin;
  pointer-events: none;
  margin: 30px;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
```

```
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #040404;
  font-family: 'Noto Sans JP', sans-serif;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(50px + 2vmin);
  color:#fcba03 ;
}

/*
button {
  background-color: #fcba03 ;
} */



@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

Uitls.ts

```
/**
 * @returns the amount in cents formatted in a string of form: d.cc where d in dollars and c is cents. example: 5.34
 */
export function formatMoney(amount_cents: number) {
    const dollars = Math.trunc(amount_cents / 100);
    const cents = amount_cents % 100;
    return `${dollars}.${`${cents}`.padStart(2, "0")}`;
}

export async function sleep(ms: number) {
    return new Promise((resolve) => setTimeout(resolve, ms));
}
```

StateProvider.tsx

```
// data layer to prevent props drilling
// using React context API
// pretty much creating global variables
import { createContext, useContext, useReducer } from "react";

//data layer
export const StateContext = createContext(null as any);
```

```
//provider
export const StateProvider = ({ reducer, initialState, children }) => (
    <StateContext.Provider value = {useReducer(reducer, initialState)}>
        {children}
    </StateContext.Provider>
);

// How we use context api inside of a component
export const useStateValue = () => useContext(StateContext);
```

Reducer.ts

```
export const initialState = {
    order: [],
    user: null,
    currentBar: null
};

export const getOrderTotal = (order: any) =>
    order?.reduce((amount: any, item: any) => item.price + amount, 0);

function getOrderStoragekey(state) {
    return `${state?.user?.username ?? ""}-order-state`;
}

/**
 * Takes the order data from state and saves it to local storage with a key based on the username.
 */
function saveOrder(state) {
    const orderToSave = state?.order;
    const key = getOrderStoragekey(state);
    if (orderToSave.length === 0 || orderToSave == null) {
        localStorage.removeItem(key);
    } else {
        localStorage.setItem(key, JSON.stringify(orderToSave));
    }
    return state;
}

/**
 * Loads the user's order based on state and returns a new state with the loaded order.
 */
function loadOrder(state) {
    const order_str = localStorage.getItem(getOrderStoragekey(state));
    if (order_str == null) {
        return { ...state };
    }

    const order = (() => {
        try {
            return JSON.parse(order_str);
        } catch (e) {
            if (e instanceof SyntaxError) {
                return null;
            } else {
                throw e;
            }
        }
```

```
    })();

    if (order == null) {
        return {
            ...state,
        };
    } else {
        return {
            ...state,
            order: order,
        };
    }
}

function loadBar(state) {
    const bar_str = localStorage.getItem(getOrderStoragekey(state));
    if (bar_str == null) {
        return { ...state };
    }

    const currentBar = (() => {
        try {
            return JSON.parse(bar_str);
        } catch (e) {
            if (e instanceof SyntaxError) {
                return null;
            } else {
                throw e;
            }
        }
    })();

    if (currentBar == null) {
        return {
            ...state,
        };
    } else {
        return {
            ...state,
            currentBar: currentBar,
        };
    }
}

// This goes through all the items in the basket and adds them up starting from 0
// reduces the array to one value

const reducer = (state: any, action: any) => {
    //console.log(action);

    switch (
        action.type //mutable updates
    ) {
        case "FETCH_USER_DATA_INIT":
            return {
                ...state,
                isLoading: true,
                isError: false,
            };

        case "FETCH_USER_DATA_SUCCESS":
```

```
    return {
        ...state,
        isLoading: false,
        isError: false,
        user: action.payload.user,
    };

case "SET_BAR":
    return {
        ...state,
        currentBar : action.bar
    };

case "FETCH_USER_DATA_FAILURE":
    return { ...state, isLoading: false, isError: true };

case "RESET_USER_DATA":
    return { ...state, user: null };

case "ADD_TO_ORDER":
    //Logic for order
    return saveOrder({
        ...state,
        order: [...state.order, action.item],
    });

case "REMOVE_FROM_ORDER":
    let newOrder = [...state.order]; //copying the basket state to new basket

    // checking to see if product exists
    const index = state.order.findIndex(
        (orderItem: any) => orderItem.id === action.id
    );

    if (index >= 0) {
        newOrder.splice(index, 1); //remove from basket
    } else {
        console.warn(
            `Cant remove product (id: ${action.id}) as its not in the order`
        );
    }

    return saveOrder({ ...state, order: newOrder });

case "EMPTY_ORDER":
    return saveOrder({
        ...state,
        order: [],
    });

case "SAVE_ORDER":
    return saveOrder({ ...state });

case "LOAD_ORDER":
    return loadOrder(state);

case "LOAD_BAR":
    return loadBar(state);

default:
    return state;
```

```
    }
};

export default reducer;
```

UploadProfileImg.js

```
import React from "react";
import S3 from "react-aws-s3";
import { Button, Typography } from "@mui/material";
import { useStateValue } from "../state/StateProvider";
import { updateUser } from "../graphql/mutations";
import { API, graphqlOperation } from "aws-amplify";
import { getUser } from "../graphql/queries";
import { useSnackbar } from "notistack";

const S3_BUCKET = "barfly-pics";
const REGION = "us-west-2";
const ACCESS_KEY = "AKIA33JESCSVFLWNP6VN";

const UploadImageToS3WithReactS3 = () => {
    const [{ user }] = useStateValue();

    const { enqueueSnackbar } = useSnackbar();

    const [photo, setPhoto] = React.useState()

    const getVersion = async () => {
        try {
            const userRes = API.graphql(
                graphqlOperation(getUser, {
                    id: user.attributes.sub,
                })
            );

            const profile = await userRes;
            return profile.data.getUser._version;
        } catch (err) {
            console.log(err);
        }
    };

    const fileInput = React.useRef();

    async function updateUserImg(location) {
        const version = await getVersion();

        const res = API.graphql(
            graphqlOperation(updateUser, {
                input: {
                    id: user.attributes.sub,
                    profileImg: JSON.stringify({ img: location }),
                    _version: version,
                },
            })
        );

        const userResponse = await res;
```

```
    return userResponse;
}
const handleClick = async (event) => {
    event.preventDefault();
    if (fileInput.current) {
        let file = fileInput.current.files[0];
        const newFileName = user.attributes.sub;

        // TODO change to ENV
        console.log(process.env.REACT_APP_SECRET_ACCESS_KEY)
        const config = {
            bucketName: S3_BUCKET,
            dirName: "user_profile",
            region: REGION,
            accessKeyId: ACCESS_KEY,
            secretAccessKey: process.env.REACT_APP_SECRET_ACCESS_KEY
        };

        const ReactS3Client = new S3(config);
        ReactS3Client.uploadFile(file, newFileName).then((data) => {
            if (data.status === 204) {
                updateUserImg(data.location);
                enqueueSnackbar('Upload Successful', {
                    autoHideDuration: 1000,
                });
            } else {
                enqueueSnackbar('Upload Unsuccessful', {
                    autoHideDuration: 1000,
                });
                console.log("fail");
            }
        });
    }
};

return (
    <div style={{ margin: "1em" }}>
        <form
            style={{
                display: "flex",
                flexDirection: "column",
                justifyContent: "center",
                alignItems: "center",
            }}
            onSubmit={handleClick}
        >
            <Typography>Upload Profile Photo:</Typography>
            <input
                id="raised-button-file"
                style={{ display: "none" }}
                type="file"
                ref={fileInput}
                accept=".jpeg,.png"
                onChange={(e) => setPhoto(URL.createObjectURL(e.target.files[0]))}
            />
            <label htmlFor="raised-button-file">
                <Button
                    sx={{margin: '1em'}}
                    variant="outlined"
                    component="span"
```

```
                >
                    Select a file
                </Button>
            </label>
            {photo && (
                <img style={{height: 'auto', width: '10em'}} alt="Profile-Img" src={photo}/>
            )}

            <br />
            <Button variant="contained" disabled={!photo} type="submit">Upload</Button>
        </form>
    </div>
  );
};

export default UploadImageToS3WithReactS3;
```

Profile.tsx

```
import React from 'react'
import UploadImageToS3WithReactS3 from './UploadProfileImg'
import DisplayProfileImg from './DisplayProfileImg'

function Profile() {

  return (
    <div>
      <div style={{margin: '1em'}}>
        <DisplayProfileImg/>

      </div>
        <UploadImageToS3WithReactS3 />
    </div>

    )

}

export default Profile
```

DisplayProfileImg.tsx

```
import React from 'react'
import UploadImageToS3WithReactS3 from './UploadProfileImg'
import DisplayProfileImg from './DisplayProfileImg'

function Profile() {

  return (
    <div>
      <div style={{margin: '1em'}}>
        <DisplayProfileImg/>

      </div>
        <UploadImageToS3WithReactS3 />
    </div>

    )
```

```
}

export default Profile
```

Timting.ts

```
import { useState, useEffect } from "react";

/**
 * @returns a knock-off of the setTimeout function that's safe to use in react components.
 * Any timeouts created will be automatically cleared when the calling component de-renders.
 */
export function useTimeout() {
    // id list
    const [IDs] = useState([] as NodeJS.Timeout[]);

    // destructor
    //     clears all timeouts
    useEffect(
        () => () => {
            for (const id of IDs) {
                clearTimeout(id);
            }
        },
        []
    );

    return function reactTimeout(
        callback: (...args: any[]) => void,
        ms: number,
        ...args: any[]
    ) {
        const id = setTimeout(callback, ms, ...args);
        IDs.push(id);
        return id;
    };
}

/**
 * @returns a knock-off of the setInterval function that's safe to use in react components.
 * Any intervals created will be automatically cleared when the calling component de-renders.
 */
export function useInterval() {
    // id list
    const [IDs] = useState([] as NodeJS.Timeout[]);

    // destructor
    //     clears all intervals
    useEffect(
        () => () => {
            for (const id of IDs) {
                clearInterval(id);
            }
        },
        []
    );

    return function reactInterval(
        callback: (...args: any[]) => void,
```

```
        ms: number,
        ...args: any[]
    ) {
        const id = setInterval(callback, ms, ...args);
        IDs.push(id);
        return id;
    };
}

export function useSleep() {
    const reactTimeout = useTimeout();

    return function reactSleep(ms: number) {
        return new Promise((resolve) => reactTimeout(resolve, ms));
    };
}
```

Welcome.tsx

```
import logoWhite from "../BarflyLogoWhite.png";
import Bar from "./bars/ViewBars"

const Welcome = () => {
    return (
        <div>
            <img src={logoWhite} className="App-logo" alt="logo" />
            <div>
                <Bar/>
            </div>
        </div>
    );
};

export default Welcome;
```

RouteNavigator.tsx

```
import { useNavigate } from "react-router"

export default function RouteNavigator({navigate_ref, children} : {navigate_ref: any[], children: any}){
    const navigate = useNavigate();
    navigate_ref[0] = navigate;


    return children;
}
```

LoadingIndicator.tsx

```
/* This component displays a "loading" component to be used
when processing/waiting for an action to be completed. */

import { CircularProgress } from "@mui/material";
import Centerer from "./Centerer";
```

```
export default function LoadingIndicator({
    show,
    size = "5ch",
    ...rest
}: {
    show?: boolean;
    size?: string;
    rest?: any[];
}) {
    return (
        <Centerer>
            <CircularProgress style={{ width: size, height: size }} />
        </Centerer>
    );
}
```

Common.tsx

```
/* This component provides views and toggles that are common
throughout the website. */

import "../App.css";
import React,{ useState, createContext, useContext } from "react";
import { useStateValue } from "../state/StateProvider";
import {
    AppBar,
    IconButton,
    List,
    Toolbar,
    Link as Typeography,
    SwipeableDrawer,
    Typography,
    ListItemButton,
    Tooltip,
} from "@mui/material";
import { Box } from "@mui/material";
import MenuIcon from "@mui/icons-material/Menu";
import ChevronLeftIcon from "@mui/icons-material/ChevronLeft";
import SportsBarIcon from "@mui/icons-material/SportsBar";
import { useNavigate } from "react-router";
import { ActionsContext } from "../App";
import DisplayProfileImg from "../Settings/DisplayProfileImg";

export const NavigateContext = createContext((path: string) => undefined);
console.debug("================= console.debug is enabled ===============");


const APPBAR_HEIGHT = "7ch";

export default function Common({ children }: { children: any }) {
    const [drawerOpen, setDrawerOpen] = useState(false);
    function toggleDrawerOpen() {
        setDrawerOpen(!drawerOpen);
    }
    function closeDrawer() {
        setDrawerOpen(false);
    }
    function openDrawer() {
```

```
    setDrawerOpen(true);
}
const [{ user, order, currentBar }] = useStateValue();

const navigate = useNavigate();
const { signOut } = useContext(ActionsContext);

return (
    <>
        <AppBar>
            <Box
                display="flex"
                alignItems="center"
                justifyContent="center"
                position="relative"
                width="100%"
                height={APPBAR_HEIGHT}
            >
                {/* appbar-left */}
                {user && (
                    <Box position="absolute" left="2ch">
                        <IconButton
                            style={{ justifySelf: "flex-end" }}
                            onClick={toggleDrawerOpen}
                            color="primary"
                        >
                            <DisplayProfileImg  />
                        </IconButton>
                    </Box>
                )}

                {/* appbar-center */}
                <Box>
                    <Typeography
                        onClick={() =>
                            navigate(
                                currentBar ? `/${currentBar.id}/menu` : "/"
                            )
                        }
                        style={{
                            textDecoration: "none",
                            fontSize: "4ch",
                            cursor: "pointer",
                            fontFamily: "Racing Sans One",
                            letterSpacing: "2px",
                        }}
                    >
                        Barfly
                    </Typeography>
                </Box>
                {/* appbar-right */}
                <Box position="absolute" right="2ch">
                    {user && currentBar ? (
                        <Tooltip
                            title={`${order.length} Items in your order`}
                        >
                            <IconButton
                                onClick={() =>
                                    navigate(
                                        `/${currentBar.id}/ordersummary`
                                    )
```

```
                }
              >
                <Box
                  display="inline"
                  sx={{
                    color: (theme) =>
                      theme.palette.primary.main,
                  }}
                >
                  <SportsBarIcon color="primary" />
                  <Typography
                    color="primary"
                    display="inline"
                  >
                    {order.length}
                  </Typography>
                </Box>
              </IconButton>
            </Tooltip>
          ) : null}
        </Box>
      </Box>
    </AppBar>
    <Toolbar />
    <SwipeableDrawer
      open={drawerOpen}
      onClose={closeDrawer}
      onOpen={openDrawer}
      anchor="left"
      style={{
        height: "100%",
      }}
      onBackdropClick={closeDrawer}
    >
      <Box
        height={APPBAR_HEIGHT}
        style={{ backgroundColor: "#111" }}
        position="relative"
      >
        <Box position="absolute" right="0">
          <IconButton onClick={closeDrawer} color="primary">
            <ChevronLeftIcon />
          </IconButton>
        </Box>
      </Box>
      <Box width="min(50vw, 30ch)">
        {/* ===== Draw Content ==== */}
        <List>
          {user && (
            <>
              <ListItemButton
                onClick={() => {
                  closeDrawer();
                  navigate(`/`);
                }}
              >
                Choose A Different Bar
              </ListItemButton>
              <ListItemButton
                onClick={() => {
                  closeDrawer();
```

```
                                navigate(
                                    `/orderstatus`
                                );
                            }}
                        >
                            Order Status
                        </ListItemButton>
                        <ListItemButton
                            onClick={() => {
                                closeDrawer();
                                navigate(`/profile`);
                            }}
                        >
                            Profile
                        </ListItemButton>

                        <ListItemButton
                            onClick={() => {
                                closeDrawer();
                                navigate("/changepass");
                            }}
                        >
                            Change Password
                        </ListItemButton>
                        <ListItemButton
                            onClick={() => {
                                closeDrawer();
                                signOut().then(() => navigate("/"));
                            }}
                        >
                            Sign Out
                        </ListItemButton>
                    </>
                )}
            </List>
        </Box>
    </SwipeableDrawer>
    {children}
</>
);
}
```

Centerer.tsx

```
import { styled } from "@mui/material/styles";

const Centerer = styled("div")((theme) => ({
    display: "flex",
    justifyContent: "center",
    alignItems: "center",
    height: "100%",
    width: "100%",
}));

export default Centerer;
```

SearchList.js

```
/* This component handles filtering through provided
input data for search functionality. */

import React from "react";
import SearchItem from "./SearchItem";

function SearchList(props) {

    var filteredData = props.data.filter((el) => {
        //if no input the return the original
        if (props.input === "") {
            return el.name;
        }
        //return the item which contains the user input
        else {
            return el.name.toLowerCase().includes(props.input);
        }
    });

    return (
        <div
            style={{
                backgroundColor: "#292929",
                width: "60%",
                minWidth: "300px",
                margin: "auto",
                display: "block",
                borderBottom: "0.5px solid",
            }}
        >
            {filteredData.map((item) => (
                <SearchItem key={item.id} item={item} type={props.type} />
            ))}
        </div>
    );
}

export default SearchList;
```

SearchItem.js

```
/* This component displays a single search item and
the option to select it. */

import React from "react";
import { Button, Typography } from "@mui/material";
import { Box } from "@mui/system";
import { useSnackbar } from "notistack";
import { useNavigate } from "react-router";
import { useStateValue } from "../../state/StateProvider";

const SearchItem = ({ item, type }) => {
    const [{ state }, dispatch] = useStateValue();
    const { enqueueSnackbar } = useSnackbar();

    const navigate = useNavigate();
```

```
const goToMenu = () => {
    dispatch({
        type: "SET_BAR",
        bar: item,
    });
    navigate("./menu");
};

const addToOrder = () => {
    dispatch({
        type: "ADD_TO_ORDER",
        item: {
            id: item.id,
            name: item.name,
            price: item.price,
        },
    });

    enqueueSnackbar(`${item.name} Added To Order`, {
        autoHideDuration: 1000,
    });
};

const renderAdd = () => {
    return (
        <Button
            size="small"
            variant="contained"
            onClick={(e) => {
                e.stopPropagation();
                addToOrder();
            }}
        >
            Add To Order
        </Button>
    );
};

const renderSelect = () => {
    return (
        <Button variant="outlined" onClick={goToMenu}>
            Select Bar
        </Button>
    );
};

return (
    <Box
        display="flex"
        flexDirection="row"
        justifyContent="space-between"
        padding="1ch"
        borderColor={(theme) => theme.palette.primary.main}
        border="0.5px solid"
        borderTop="0"
        borderBottom="0"
        backgroundColor="#222"
    >
        <Typography>{item.name}</Typography>
```

```
            {type === "menuItem" ? renderAdd() : null}
            {type === "bar" ? renderSelect() : null}
        </Box>
    );
};

export default SearchItem;
```

Payment.js

```
/* This component handles the payment transcation using
Stripe.js. */

import { useState } from "react";
import {
    Button,
    ButtonGroup,
    TextField,
    Box,
    CircularProgress,
} from "@mui/material";
import { useNavigate } from "react-router-dom";
import { getOrderTotal } from "../../state/reducer";
import { useStateValue } from "../../state/StateProvider";
import { API, graphqlOperation } from "aws-amplify";
import { createOrder } from "../../graphql/mutations";
import {
    useStripe,
    useElements,
    CardCvcElement,
    CardNumberElement,
    CardExpiryElement,
} from "@stripe/react-stripe-js";
import StripeInput from "./StripeInput";
import axios from "axios";

function Payment() {
    const [{ order, user, currentBar }, dispatch] = useStateValue();
    const [message, setMessage] = useState("");
    const [loading, setLoading] = useState(false);

    const stripe = useStripe();
    const elements = useElements();

    const navigator = useNavigate();

    function navigate(destination) {
        navigator(destination);
    }

    const handleSubmit = async () => {
        setLoading(true);

        const { error, paymentMethod } = await stripe.createPaymentMethod({
            type: "card",
            card: elements.getElement(CardCvcElement),
        });

        if (error) {
            setMessage("Payment was unsuccessful !");
```

```
    } else {
      try {
        const { id } = paymentMethod;
        const response = await axios.post(
          "https://rocky-beach-86430.herokuapp.com/stripe/charge",
          {
            amount: getOrderTotal(order).toFixed(2) * 100,
            id: id,
          }
        );

        if (response.data.success) {
          const payload = {
            items: JSON.stringify(order),
            completed: false,
            userID: user.attributes.sub,
            orderStatus: "received",
            barID: currentBar.id
          };

          try {
             await API.graphql(
               graphqlOperation(createOrder, { input: payload })
            ).then((data) => {
              dispatch({
                type: "EMPTY_ORDER",
              });
              setTimeout(() => {
                navigate(`/orderstatus`);
              }, [100]);

              return data;
            });
          } catch (err) {
            console.log(err);
          }
        } else {
          setMessage("Payment was unsuccessful !");
        }
      } catch (error) {
        console.log(error);
        setMessage("Payment was unsuccessful !");
      }
    }
    setLoading(false);
};

return (
    <div>
      <h2>Payment</h2>

      <div
        style={{
          width: "35%",
          minWidth: "200px",
          margin: "auto",
          marginTop: "50px",
        }}
      >
        <TextField
          label="Credit Card Number"
```

```
        name="ccnumber"
        variant="outlined"
        fullWidth
        style={{
          backgroundColor: "#292929",
        }}
        InputProps={{
          inputComponent: StripeInput,
          inputProps: {
            component: CardNumberElement,
          },
        }}
        InputLabelProps={{ shrink: true }}
      />
    </div>
    <div
      style={{
        width: "35%",
        minWidth: "200px",
        margin: "auto",
        marginTop: "20px",
        marginBottom: "20px",
        display: "flex",
      }}
    >
      <TextField
        label="Expiration Date"
        name="ccexp"
        variant="outlined"
        required
        fullWidth
        style={{
          backgroundColor: "#292929",
        }}
        InputProps={{
          inputProps: {
            component: CardExpiryElement,
          },
          inputComponent: StripeInput,
        }}
        InputLabelProps={{ shrink: true }}
      ></TextField>
      <TextField
        style={{ width: "100px", backgroundColor: "#292929" }}
        label="CVC"
        name="cvc"
        variant="outlined"
        required
        fullWidth
        InputProps={{
          inputProps: {
            component: CardCvcElement,
          },
          inputComponent: StripeInput,
        }}
        InputLabelProps={{ shrink: true }}
      ></TextField>
    </div>
    <h4>Order Total: ${getOrderTotal(order).toFixed(2)}</h4>
    <ButtonGroup>
      <Button
```

```
                    className="buttons"
                    variant="outlined"
                    onClick={() => navigate(`/${currentBar.id}/ordersummary`)}
                    style={{
                        backgroundColor: "#292929",
                    }}
                >
                    Back to Summary
                </Button>
                <Button
                    className="buttons"
                    variant="contained"
                    disabled={loading}
                    onClick={handleSubmit}
                >
                    {loading ? <CircularProgress size={24} /> : "Place Order"}
                </Button>
            </ButtonGroup>
            <Box height="2em" style={{ marginTop: "20px" }}>
                {message}
            </Box>
        </div>
    );
}

export default Payment;
```

StripeInput.js

```
import React, { useRef, useImperativeHandle } from 'react'

const StripeInput = ({ component: Component, inputRef, ...other }) => {
    const inputStyle= {
        color: '#fcba03'
    }

    const elementRef = useRef();
    useImperativeHandle(inputRef, () => ({
        focus: () => elementRef.current.focus
    }));

    return (
        <Component onReady={element => (elementRef.current = element)} {...other}
            options={{
                style: {
                    base: inputStyle,
                },
            }}
        />
    );
}

export default StripeInput
```

OrderSummary.js

```
/* This component displays the list of items in a user's
cart as well as the order total. */

import { Button, ButtonGroup } from "@mui/material";
import { useStateValue } from "../../state/StateProvider";
import OrderItem from "./OrderItem";
import { getOrderTotal } from "../../state/reducer";
import { useNavigate } from "react-router-dom";

const OrderSummary = () => {
   const [{ order, currentBar }] = useStateValue();

   const navigator = useNavigate();

   function navigate(destination) {
      navigator(destination);
   }

   const uniqueOrder = [
      ...new Map(order.map((item) => [item["name"], item])).values(),
   ];

   return (
      <div>
         <h2>Summary</h2>
         <div>
            {uniqueOrder.map((item) => {
               var count = order.filter(
                  (v) => v.name === item.name
               ).length;
               return (
                  <OrderItem key={item.id} item={item} count={count} />
               );
            })}
         </div>

         <h4>Order Total: ${getOrderTotal(order).toFixed(2)}</h4>

         <ButtonGroup
            style={{ padding: "15px" }}
            disableElevation
            variant="outlined"
         >
            <Button
               className="buttons"
               style={{ backgroundColor: "#292929" }}
               onClick={() => navigate(-1)}
            >
               Back to Menu
            </Button>
            <Button
               className="buttons"
               variant="contained"
               onClick={() => navigate(`/${currentBar.id}/payment`)}
               disabled={order.length === 0}
            >
               Checkout (${getOrderTotal(order).toFixed(2)})
            </Button>
         </ButtonGroup>
      </div>
   );
```

```
};

export default OrderSummary;
```

OrderStatus.jsx

```
/* This component displays a user's order history
by different categories of 'status'. */

import { useEffect, useState } from "react";
import { API, graphqlOperation } from "aws-amplify";
import {
    Button,
    Collapse,
    IconButton,
    Paper,
    Tooltip,
    Typography,
    ButtonGroup,
} from "@mui/material";
import { Box } from "@mui/system";
import { useTimeout } from "../../hooks/timing";
import DoNotDisturbIcon from "@mui/icons-material/DoNotDisturb";
import CancelIcon from "@mui/icons-material/Cancel";
import { useNavigate } from "react-router";
import { onOrderByUserId } from "../../graphql/subscriptions";
import { useStateValue } from "../../state/StateProvider";
import { listOrders } from "../../graphql/queries";
import _ from "lodash";
import DisplayProfileImg from "../../Settings/DisplayProfileImg";
import ArrowDropDownIcon from "@mui/icons-material/ArrowDropDown";
import ArrowDropUpIcon from '@mui/icons-material/ArrowDropUp';

//In case subscriptions delete themselves again some how
/*              onOrderByUserId(userID: String): Order
                        @aws_subscribe(mutations: ["updateOrder"])
@aws_api_key
@aws_iam
              onOrderByOrderId(id: String!): Order
                        @aws_subscribe(mutations: ["updateOrder"])
@aws_api_key
@aws_iam
*/

function OrderItem({ orderItem, style }) {
    const [showItems, setShowItems] = useState(false);
    const [showCancel, setShowCancel] = useState(false);
    const timeout = useTimeout();

    const [order, setOrder] = useState(orderItem);
    const [items] = useState(JSON.parse(orderItem.items));

    function openCancel() {
        try {
            setShowCancel(true);
        } finally {
```

```
        timeout(() => setShowCancel(false), 3000);
    }
}

const getTime = () => {
    const orderDate = new Date(order.createdAt);
    const time = orderDate.getMonth() + 1 + '/' +  orderDate.getDate() + '/' + orderDate.getFullYear();
    return time
}

return (
    <Paper onClick={() => setShowItems((show) => !show)} style={style}>
        <Box display="flex" flexDirection="row">
            <Box display="flex" flexDirection="column">
            <Box display='flex'>
                <Typography sx={{marginBottom: '.5em'}}>
                    $
                    {items
                        .reduce((total, item) => total + item.price, 0.0)
                        .toFixed(2)}
                    {" - "}
                    {items.length} item
                    {items.length !== 1 && "s"}
                    {" on "}
                    {getTime()}
                </Typography>
                <IconButton sx={{ paddingTop: '0px'}}>
                    {showItems ? <ArrowDropUpIcon /> : <ArrowDropDownIcon /> }
                </IconButton>

            </Box>
            <Typography sx={{marginBottom: '.5em'}}>
                Order code : {order.id.substring(0,5)}
            </Typography>
            </Box>

            <Box position="relative" bottom="1ch" flexGrow="1">
                <Collapse
                    in={!showCancel}
                    style={{ position: "absolute", right: "1ch" }}
                >
                    <Tooltip title="Cancel" placement="left">
                        <IconButton
                            color="primary"
                            onClick={(e) => {
                                openCancel();
                                e.stopPropagation();
                            }}
                        >
                            <DoNotDisturbIcon />
                        </IconButton>
                    </Tooltip>
                </Collapse>
                <Collapse
                    in={showCancel}
                    style={{ position: "absolute", right: "1ch" }}
                >
                    <Tooltip
                        title="Tap Again to Cancel"
                        open={showCancel}
                        placement="left"
```

```
                    >
                        <IconButton
                            color="primary"
                            onClick={(e) => {
                                e.stopPropagation();
                            }}
                        >
                            <CancelIcon />
                        </IconButton>
                    </Tooltip>
                </Collapse>
            </Box>
        </Box>
        <Collapse in={showItems}>
            {items.map((item) => (
                <Typography
                    key={Math.random(1000)}
                    style={{ marginLeft: "4ch" }}
                >
                    <span
                        key={Math.random(1000) + ""}
                        onClick={(e) => e.stopPropagation()}
                    >
                        ${item.price.toFixed(2)} - {item.name}
                    </span>
                </Typography>
            ))}
        </Collapse>
    </Paper>
  );
}

export default function OrderStatus() {
    const navigate = useNavigate();
    const [status, setStatus] = useState("received");
    const [{ user, currentBar }] = useStateValue();
    const [activeOrders, setActiveOrders] = useState();
    const [isLoading, setIsLoading] = useState(true);
    const [empty, setEmpty] = useState(false);

    useEffect(() => {
        const listOrdersBy = async () => {
            try {
                const response_promise = API.graphql(
                    graphqlOperation(listOrders, {
                        filter: {
                            orderStatus: { eq: status },
                            userID: { eq: user.attributes.sub },
                        },
                    })
                );
                const response = await response_promise;
                setActiveOrders(_.orderBy(response.data.listOrders.items, (item) => {
                    return item.createdAt
                },'desc'));
                setIsLoading(false);

                if (response.data.listOrders.items.length === 0) {
                    setEmpty(true);
                } else {
                    setEmpty(false);
```

```
            }

            console.debug(response);
        } catch (err) {
            console.debug(err);
        }
    };
    return listOrdersBy();
}, [user.attributes.sub, status]);

return (
    <Box>
        <h2>Orders</h2>
        <div>
            <DisplayProfileImg size='5em'/>
        </div>
        <ButtonGroup
            style={{ padding: "15px", width: "100%" }}
            disableElevation
            variant="outlined"
        >
            <Button
                className={status === "received" ? "activeTab" : "tab"}
                onClick={() => setStatus("received")}
            >
                Recieved
            </Button>
            <Button
                className={status === "in-progress" ? "activeTab" : "tab"}
                onClick={() => setStatus("in-progress")}
            >
                In-Progress
            </Button>
            <Button
                className={status === "complete" ? "activeTab" : "tab"}
                onClick={() => setStatus("complete")}
            >
                Completed
            </Button>
        </ButtonGroup>

        {!isLoading && !empty ? (
            activeOrders.map((order) => (
                <OrderItem
                    key={Math.random(1000)}
                    orderItem={order}
                    style={{
                        marginBottom: "1em",
                        textAlign: "left",
                        padding: "1ch",
                    }}
                />
            ))
        ) : (
            <h2>No items are {status} yet.</h2>
        )}
        <Button
            variant="outlined"
            style={{ backgroundColor: "#292929" }}
            onClick={() => navigate(`/${currentBar.id}/menu`)}
        >
```

```
            Back to Menu
          </Button>
        </Box>
    );
}
```

MenuItem.js

```
/* This component displays a single menu item and
the option to select it. */

import { useState } from "react";
import { Button, Typography, Collapse, IconButton } from "@mui/material";
import ArrowDropDownIcon from "@mui/icons-material/ArrowDropDown";
import ArrowDropUpIcon from '@mui/icons-material/ArrowDropUp';
import { Box } from "@mui/system";
import { useSnackbar } from "notistack";
import { useStateValue } from "../../state/StateProvider";

const MenuItem = ({ item }) => {
    const [{ order }, dispatch] = useStateValue();
    const { enqueueSnackbar } = useSnackbar();
    const [descriptionOpen, setDescriptionOpen] = useState();
    const toggleDescription = () => setDescriptionOpen(!descriptionOpen);

    const addToOrder = () => {
        dispatch({
            type: "ADD_TO_ORDER",
            item: {
                id: item.id,
                name: item.name,
                price: item.price,
            },
        });

        enqueueSnackbar(`${item.name} Added To Order`, {
            autoHideDuration: 1000,
        });
    };

    console.log(item)

    return (
        <Box
            display="flex"
            flexDirection="column"
            alignItems="flex-start"
            padding="1ch"
            borderColor={(theme) => theme.palette.primary.main}
            borderBottom="1px solid"
            onClick={toggleDescription}
        >
            <Box display="flex">
                <Typography
                    sx={{ marginBottom: ".5em" }}
                    fontFamily={"Tahoma"}
                    fontWeight="bold"
                >
                    {item.name}
```

```
            </Typography>
            <IconButton sx={{ paddingTop: '0px'}}>
              {descriptionOpen ? <ArrowDropUpIcon /> : <ArrowDropDownIcon /> }
            </IconButton>
          </Box>

          <Box
            display="flex"
            flexDirection="column"
            alignitems="flex-start"
            paddingLeft="1ch"
          >
            <Collapse
              in={descriptionOpen}
              // style={{ borderBottom: "1px dotted" }}
              sx={{
                borderBottom: "1px dotted",
                borderColor: (theme) => theme.palette.text.secondary,
              }}
            >
              <Typography
                textAlign="left"
                color={(theme) => theme.palette.text.secondary}
                onClick={(e) => e.stopPropagation()}
              >
                {item.description}
              </Typography>
            </Collapse>
            <Typography
              color={(theme) => theme.palette.text.secondary}
              textAlign="left"
              sx={{ marginBottom: ".5em" }}
            >
              $ {item.price.toFixed(2)}
            </Typography>
          </Box>
          <Button
            size="small"
            variant="contained"
            onClick={(e) => {
              e.stopPropagation();
              addToOrder();
            }}
          >
            Add To Order
          </Button>
        </Box>
    );
};

export default MenuItem;
```

MenuCategory.js

```
/* This component takes a menu category as
input and displays the items in it. */

import React from 'react';
import {
```

```
    Accordion,
    AccordionDetails,
    AccordionSummary,
    Typography,
} from "@mui/material";
import ExpandMoreIcon from "@mui/icons-material/ExpandMore";
import MenuItem from "./MenuItem";

const MenuCategory = ({ category, items }) => {
    const filteredItems = items.items.filter((item) => item._deleted === null);

    return (
        <Accordion>
            <AccordionSummary expandIcon={<ExpandMoreIcon />}>
                <Typography fontFamily={"Lato"} fontStyle="bold" >{category}</Typography>
            </AccordionSummary>
            <AccordionDetails>
                {filteredItems.map((item) => (
                    <MenuItem key={item.id} item={item} />
                ))}
            </AccordionDetails>
        </Accordion>
    );
};

export default MenuCategory;
```

Menu.js

```
/* This component gets the menu from the backend
and displays the items in categories. */

import { useState, useEffect } from "react";
import { API, graphqlOperation } from "aws-amplify";
import { getWholeMenu, listMenus } from "../../graphql/queries";
import LoadingIndicator from "../LoadingIndicator";
import MenuCategory from "./MenuCategory";
import SearchList from "../search/SearchList";
import { Box, Button, TextField } from "@mui/material";
import { useNavigate } from "react-router";
import { useStateValue } from "../../state/StateProvider";
import { useLocation } from "react-router-dom";
import { getBar } from "../../graphql/queries";

const Menu = () => {
    const [menu, setMenu] = useState({});
    const [menuID, setMenuID] = useState("");
    const [isLoading, setIsLoading] = useState(true);
    const navigate = useNavigate();
    const [{ currentBar }, dispatch] = useStateValue();
    const [searchText, setSearchText] = useState("");
    const [items, setItems] = useState([]);
    const [bar, setBar] = useState();

    let searchHandler = (e) => {
        var lowerCase = e.target.value.toLowerCase();
        setSearchText(lowerCase);
    };
```

```
const location = useLocation();

const barid = location.pathname.split("/")[1];

useEffect(() => {
    const GetBar = async () => {
        try {
            const bar_response = API.graphql(
                graphqlOperation(getBar, {
                    id: barid,
                })
            );

            const bar = await bar_response;

            setBar(bar.data.getBar);

            dispatch({
                type: "SET_BAR",
                bar: bar.data.getBar,
            });
        } catch (err) {
            console.log(err);
        }
    };

    GetBar();
}, [barid, dispatch]);

useEffect(() => {
    const fetchMenu = async () => {
        try {
            const response_promise = API.graphql(
                graphqlOperation(listMenus, {
                    filter: { barID: { eq: barid } },
                })
            );

            const response = await response_promise;

            var menuData = response.data.listMenus.items[0];
            setMenuID(String(menuData.id));
        } catch (err) {
            console.log(err);
        }

        if (menuID) {
            try {
                setIsLoading(true);
                const tresponse_promise = API.graphql(
                    graphqlOperation(getWholeMenu, {
                        id: menuID,
                    })
                );

                const tresponse = await tresponse_promise;

                setMenu(tresponse.data.getMenu);
                //iterate though menu to add to search list
                Object.keys(tresponse.data.getMenu)
                    .filter((category) => category !== "id")
```

```
                .map((category) =>
                    tresponse.data.getMenu[category].items
                        .filter((item) => item._deleted === null)
                        .map((item) =>
                            setItems((items) => [...items, item])
                        )
                );
            } catch (err) {
                console.log(err);
            } finally {
                setIsLoading(false);
            }
        }
    };

    fetchMenu();
}, [barid, menuID]);

const renderMenu = () => {
    if (menu) {
        return Object.keys(menu)
            .filter((category) => category !== "id")
            .map((category) => {
                return (
                    <MenuCategory
                        key={category}
                        category={category}
                        items={menu[category]}
                    />
                );
            });
    } else {
        return <></>;
    }
};

return (
    <Box paddingBottom="5em">
        {bar && (
            <>
                <h1>{bar.name}</h1>
                <div className="search">
                    <TextField
                        id="outlined-basic"
                        variant="outlined"
                        fullWidth
                        label="Search Menu"
                        onChange={searchHandler}
                        style={{
                            backgroundColor: "#292929",
                            width: "60%",
                            minWidth: "300px",
                        }}
                    />
                    {searchText !== "" && !isLoading ? (
                        <SearchList
                            input={searchText}
                            data={items}
                            type="menuItem"
                        ></SearchList>
                    ) : null}
```

```
                </div>
                <h2>Menu</h2>
                {isLoading ? (
                    <LoadingIndicator size="30px" />
                ) : (
                    renderMenu()
                )}
                <Box height="2em" />
                <Button
                    className="buttons"
                    variant="contained"
                    onClick={() => navigate(`/${bar.id}/ordersummary`)}
                >
                    Place Order
                </Button>
            </>
        )}
    </Box>
  );
};

export default Menu;
```

ViewBars.js

```
/* This component gets the list of bars from the backend
and displays them for the user to choose. */

import { useState, useEffect } from "react";
import { API, graphqlOperation } from "aws-amplify";
import { listBars } from "../../graphql/queries";
import LoadingIndicator from "../LoadingIndicator";
import Bar from "./Bar";
import { useStateValue } from "../../state/StateProvider";
import SearchList from "../search/SearchList";
import { Box, TextField } from "@mui/material";
import { useNavigate } from "react-router";

const ViewBars = () => {
  const [barList, setBarList] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
  const [searchText, setSearchText] = useState("");

  const [{ state }, dispatch] = useStateValue();

  const navigate = useNavigate();

  const goToMenu = (bar) => {
    dispatch({
      type: "SET_BAR",
      bar: bar,
    })
    navigate(`/${bar.id}/menu`)
  }

  let searchHandler = (e) => {
    var lowerCase = e.target.value.toLowerCase();
    setSearchText(lowerCase);
  };
```

```
  useEffect(() => {
    dispatch({ type: "EMPTY_ORDER" });

    const fetchBars = async () => {
      try {
        const response_promise = API.graphql(
          graphqlOperation(listBars)
        );

        const response = await response_promise;

        setBarList(response.data.listBars.items.filter((item) => item._deleted === null));
      } catch (err) {
        console.log(err);
      } finally {
        setIsLoading(false);
      }
    };

    fetchBars();
  }, [dispatch]);

  return (
    <Box paddingBottom="5em">
      <TextField
        id="outlined-basic"
        variant="outlined"
        fullWidth
        label="Search Barfly"
        onChange={searchHandler}
        style={{
          backgroundColor: "#292929",
          width: "60%",
          minWidth: "300px",
          marginTop:'1em'
        }}
      />
      {searchText !== "" && !isLoading ? (
        <SearchList
          input={searchText}
          data={barList}
          type="bar"
        ></SearchList>
      ) : null}
      <h2>Popular Bars in Your Area</h2>
      {!isLoading ? (
        barList.map((bar) => <div onClick={() => {goToMenu(bar)}} style={{cursor: "pointer"}}> <Bar key={bar.id} bar={bar}/>
</div>)
      ) : (
        <LoadingIndicator size="30px" />
      )}
      <Box height="2em" />
    </Box>
  );
};

export default ViewBars;
```

Bar.js

```
/* This component displays a single bar and
the option to select it. */

import { useStateValue } from '../../state/StateProvider';

function Bar({ bar }) {
   const [ {currentBar}, dispatch] = useStateValue();

   const barImgUrl = JSON.parse(bar.profileImg)

   return (
      <div style={{
         display: 'flex',
         justifyContent: 'space-between',
         flexDirection: 'column',
         alignItems: 'center',
         padding: '5px',
         backgroundColor:'#121212',
         backgroundImage: 'linear-gradient(rgba(255, 255, 255, 0.05), rgba(255, 255, 255, 0.05))',
         marginBottom:'10px',
         borderRadius:'9px'}}
         >

         <div>
            <h2 style={{margin: '10px', fontFamily: 'Tahoma', textAlign: 'center'}}>{bar.name}</h2>
            {barImgUrl && <img  style={{ height: 'auto', width: '12em', borderRadius:'9px'}} alt="" src={barImgUrl.img}/>}

            <h2 class="bio" >{bar.bio}</h2>
         </div>
      </div>
   )
}

export default Bar
```

SignUp.tsx

```
/* This component handles Sign Up for
users with AWS authentication and adds users
to the backend. */

import { useEffect, useState } from "react";
import logoWhite from "../../BarflyLogoWhite.png";
import { Auth, API, graphqlOperation } from "aws-amplify";
import {  useNavigate } from "react-router-dom";
import TextField from "@mui/material/TextField";
import Button from "@mui/material/Button";
import prevDef from "../../decorators/prevDef";
import { Box } from "@mui/system";
import Centerer from "../Centerer";
import { ButtonGroup } from "@mui/material";
import { createUser } from "../../graphql/mutations";
```

```
import LoadingIndicator from "../LoadingIndicator";

const SignUp = () => {
    const [email, setEmail] = useState(null);
    const [password, setPassword] = useState(null);
    const [password2, setPassword2] = useState(null);
    const [formIssue, setFormIssue_raw] = useState("");
    function setFormIssue(value : string) {
        setFormIssue_raw(value);
        setMessage(value);
    }
    const [message, setMessage] = useState("");

    const [phone, setPhone] = useState("");
    const [age, setAge] = useState("");
    const [name, setName] = useState("");
    const [code, setCode] = useState("");

    const [signingUp, setSigningUp] = useState(false);
    const [confirming, setConfirming] = useState(false);


    useEffect(() => {
        if (email === null || password === null || password2 === null) {
            setFormIssue("");
        } else if (password !== password2) {
            setFormIssue("Passwords Do Not Match");
        } else if (password === "") {
            setFormIssue("Password is Blank");
        } else if (email === "") {
            setFormIssue("Email is Blank");
        } else if (name === "") {
            setFormIssue("Name is Blank");
        } else if (phone === "") {
            setFormIssue("Phone is Blank");
        } else if (phone.match(/^(\()?\d{3}(\))?(-|\s)?\d{3}(-|\s)\d{4}$/)) {
            setFormIssue("Phone Number Issn't Valid")
        } else if (!age) {
            setFormIssue("Age is Blank")
        } else if (parseInt(age) < 21) {
            setFormIssue("You Must be 21 Years Old")
        } else {
            setFormIssue(null);
        }
    }, [email, password, password2, phone, age, name]);

    const [signedUp, setSignedUp] = useState(false);

    const navigate = useNavigate();

    const createAccount = async (event: any) => {
        event.preventDefault(); //prevents referesh
        try {
            setSigningUp(true);
            const data = Auth.signUp(name.replace(" ", ""), password, email);
            const dataResponse = await data;
            //Move this too confirm sign up later once error checking is implemented
            const user = API.graphql(
                graphqlOperation(createUser, {
                    input: {
                        id: dataResponse.userSub,
```

```
                name: name,
                email: email,
                phone: phone,
                age: age.toString()
              },
            })
        );
        const userResponse = await user;
        setSignedUp(true);
        return userResponse
        //onCreateAccount(name)
        //navigate('/confirmSignUp');
    } catch (err) {
        setMessage(err.message);
    } finally {
        setSigningUp(false);
    }
};

async function confirmSignUp() {
    try {
        setConfirming(true);
        await Auth.confirmSignUp(name.replace(" ", ""), code);
        navigate("/");
    } catch (error) {
        setMessage(error.message);
        console.log("error confirming sign up", error);
    } finally {
        setConfirming(false);
    }
}

const renderConfirmSignUp = () => {
    return (
        <div>
            <h2>Please Enter Confirmation Code From Email</h2>
            <img src={logoWhite} className="App-logo" alt="logo" />
            <TextField
                fullWidth
                margin="dense"
                value={name ?? ""}
                label="name"
                variant="outlined"
                disabled
            />
            <TextField
                margin="dense"
                value={code ?? ""}
                onChange={(e) => setCode(e.target.value)}
                label="code"
                variant="outlined"
                type="text"
                required
            />

            <Box minHeight="2em">{message}</Box>
            <ButtonGroup style={{ width: "min(50ch, 100%)" }}>
                <Button
                    style={{ width: "50%" }}
                    size="large"
                    variant="outlined"
```

```
                onClick={() => navigate("/")}
              >
                Back To Sign In
              </Button>
              {confirming ? (
                <Box width="50%">
                  <LoadingIndicator size="3ch" />
                </Box>
              ) : (
                <Button
                  style={{ width: "50%" }}
                  size="large"
                  variant="contained"
                  onClick={confirmSignUp}
                >
                  CONFIRM
                </Button>
              )}
            </ButtonGroup>
          </div>
        );
      };

      const renderSignUp = () => {
        return (
          <div>
            {/* <h1>Sign Up For Barfly!</h1> */}
            <img src={logoWhite} className="App-logo" alt="logo" />

            <h2>Create An Account</h2>
            <form
              onSubmit={prevDef((e) => {
                if (formIssue === null) {
                  createAccount(e);
                }
              })}
              style={{
                display: "flex",
                flexDirection: "column",
                alignItems: "stretch",
              }}
            >
              <TextField
                margin="dense"
                value={name ?? ""}
                onChange={(e) => setName(e.target.value)}
                label="name"
                variant="outlined"
                type="text"
                required
              />
              <TextField
                margin="dense"
                value={email ?? ""}
                onChange={(e) => setEmail(e.target.value)}
                label="email"
                variant="outlined"
                required
              />
              <TextField
                margin="dense"
```

```
    value={password ?? ""}
    onChange={(e) => setPassword(e.target.value)}
    label="Password"
    variant="outlined"
    type="password"
    required
  />
  <TextField
    margin="dense"
    value={password2 ?? ""}
    onChange={(e) => setPassword2(e.target.value)}
    label="Confirm Password"
    variant="outlined"
    type="password"
    required
  />

  <Box display="flex">
    <TextField
      margin="dense"
      value={phone ?? ""}
      onChange={(e) => setPhone(e.target.value)}
      label="phone"
      variant="outlined"
      type="phone"
      required
      style={{ flexGrow: 1, minWidth: "5ch" }}
    />
    <TextField
      margin="dense"
      value={age}
      onChange={(e) => setAge(e.target.value)}
      label="age"
      type="number"
      required
      style={{ width: "20ch", minWidth: "10ch" }}
    />

  </Box>
  <Box margin="1ch" marginBottom="1.3ch" lineHeight=".5em">
    {message}
  </Box>
  <Centerer>
    <ButtonGroup
      style={{ width: "40ch", minWidth: "25ch" }}
    >
      <Button
        style={{ width: "50%", height: "5ch" }}
        variant="outlined"
        onClick={() => navigate("/")}
      >
        Back To Sign In
      </Button>
      {signingUp ? (
        <div style={{ width: "50%", height: "5ch" }}>
          <LoadingIndicator size="3ch" />
        </div>
      ) : (
        <Button
          style={{ width: "50%", height: "5ch" }}
          variant="contained"
```

```
                        type="submit"
                        disabled={formIssue !== null}
                      >
                        Sign Up!
                      </Button>
                    )}
                  </ButtonGroup>
                </Centerer>
              </form>
              {/* <br />
              <span style={{ margin: "20px" }}>
                Have an account?{" "}
                <Link style={{ color: "white" }} to="/">
                  Sign In
                </Link>
              </span> */}
            </div>
        );
    };

    return <>{!signedUp ? renderSignUp() : renderConfirmSignUp()}</>;
};

export default SignUp;
```

SignIn.tsx

```
/* This component handles Sign-In for
users with AWS authentication. */

import { useState } from "react";
import logoWhite from "../../BarflyLogoWhite.png";
import { Auth } from "aws-amplify";
import { Link, useNavigate } from "react-router-dom";
import { TextField, Box } from "@mui/material";
import Button from "@mui/material/Button";
import "../../styles/auth.css";
import prevDef from "../../decorators/prevDef";
import LoadingIndicator from "../LoadingIndicator";
import Centerer from "../Centerer";

const SignIn = () => {
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [loggingIn, setLoggingIn] = useState(false);
    const [capsLock, setCapsLock] = useState(false);
    const [message, setMessage] = useState("");
    function detectCapsLock(e: React.KeyboardEvent) {
        setCapsLock(e.getModifierState("CapsLock"));
    }

    const navigate = useNavigate();

    const signIn = async (event: any) => {
        try {
            setLoggingIn(true);

            await Auth.signIn(email, password);
```

```
      setMessage("Loggin Successful");
      navigate("/");
    } catch (err) {
      setMessage(err.message);
      console.error(err);
    } finally {
      setLoggingIn(false);
    }
};


return (
  <div onKeyUp={prevDef(detectCapsLock)}>

    <img src={logoWhite} className="App-logo" alt="logo" />

    <h2>Sign In</h2>
    <form
      onSubmit={prevDef(signIn)}
      style={{
        display: "flex",
        flexDirection: "column",
        alignItems: "stretch",
      }}
    >
      <TextField
        value={email ?? ""}
        onChange={(e) => setEmail(e.target.value)}
        label="email"
        required
      />
      <Box height="1.3em" lineHeight="2.9em" alignSelf="flex-start">
        {capsLock && "⚠ CAPSLOCK IS ON ⚠"}
      </Box>
      <TextField
        value={password ?? ""}
        onChange={(e) => setPassword(e.target.value)}
        label="password"
        variant="outlined"
        type="password"
        margin="normal"
        required
      />
      <Box height="1.3em" lineHeight=".8em" alignSelf="flex-start">
        {message}
      </Box>
      {/* caps lock warning: only a littler pointless on a mobile-focused website */}
      <Centerer>
        <Box display="inline-block" width="12ch" marginBottom="1em">
          {loggingIn ? (
            <LoadingIndicator size="30px" />
          ) : (
            <Centerer>
              <Button
                size="large"
                type="submit"
                variant="contained"
              >
                Sign In
              </Button>
            </Centerer>
```

```
                    )}
                </Box>
            </Centerer>
            <br />
            <span style={{ fontSize: "1em" }}>
                <Link style={{ color: "white" }} to="/forgotpass">
                    Forgot Password
                </Link>{" "}
                <Link style={{ color: "white" }} to="/signup">
                    Create Account
                </Link>
            </span>
        </form>
    </div>
  );
};

export default SignIn;
```

ChangePassword.tsx

```
import  {  useEffect, useState } from "react";
import logoWhite from "../../../BarflyLogoWhite.png";
import { Auth } from "aws-amplify";
import { TextField, Box, ButtonGroup } from "@mui/material";
import Button from "@mui/material/Button";
import "../../../styles/auth.css";
import LoadingIndicator from "../../LoadingIndicator";
import { useNavigate } from "react-router";
import prevDef from "../../../decorators/prevDef";
import { useSnackbar } from "notistack";
//import { useParams } from "react-router";

export default function ChangePassword() {
  const [oldPassword, setOldPassword] = useState(null);
  const [password, setPassword] = useState(null);
  const [password2, setPassword2] = useState(null);
  const [message, setMessage] = useState("");
  const [passwordIssue, setPasswordIssue_raw] = useState("");
  const { enqueueSnackbar, closeSnackbar } = useSnackbar();
  function setPasswordIssue(value) {
    setPasswordIssue_raw(value);
    setMessage(value);
  }

  useEffect(() => {
    if (password === null || password2 === null) {
      setPasswordIssue("");
    } else if (password !== password2) {
      setPasswordIssue("Passwords Do Not Match");
    } else if (password === "") {
      setPasswordIssue("Password is Blank");
    } else {
      setPasswordIssue(null);
    }
  }, [password, password2]);

  //const [email, setEmail] = useState("");
  const TEXTFIELD_SPACING = "2ch";
```

```
// const [requestingReset, setRequestingReset] = useState(false);
const [resetingPassword, setResetingPassword] = useState(false);

const navigate = useNavigate();

async function resetPassword() {
   try {
      if (passwordIssue !== null) {
         return;
      }
      setResetingPassword(true);
      const currentUser = await Auth.currentAuthenticatedUser();
      await Auth.changePassword(
         currentUser,
         oldPassword,
         password
      );
      enqueueSnackbar("Password Reset Successfuly", {
         autoHideDuration: 5000,
      });
      navigate("/");
   } catch (err) {
      setMessage(err.message);
   } finally {
      setResetingPassword(false);
   }
}

return (
   <div>
      <img src={logoWhite} className="App-logo" alt="logo" />

      <h2>Change Password</h2>
      <Box display="flex" flexDirection="column" alignItems="center">
         <Box width="min(100%, 60ch)">

            <form
               onSubmit={prevDef(resetPassword)}
               style={{
                  display: "flex",
                  flexDirection: "column",
                  alignItems: "center",
                  width: "100%",
               }}
            >
               <TextField
               label="Old Password"
               value={oldPassword ?? ""}
               type="password"
               onChange={(e) => setOldPassword(e.target.value)}
               style={{
                  marginBottom: TEXTFIELD_SPACING,
                  width: "100%",
               }}
               required
               autoFocus
               autoComplete="off"
            />
               <TextField
                  label="New Password"
```

```
                        value={password ?? ""}
                        onChange={(e) => setPassword(e.target.value)}
                        type="password"
                        style={{
                            marginBottom: TEXTFIELD_SPACING,
                            width: "100%",
                        }}
                        required
                        autoComplete="off"
                    />
                    <TextField
                        label="Confirm New Password"
                        value={password2 ?? ""}
                        onChange={(e) => setPassword2(e.target.value)}
                        type="password"
                        style={{
                            width: "100%",
                        }}
                        required
                    />
                    <Box height="2em">{message}</Box>
                    <ButtonGroup style={{ width: "100%", height: "5ch" }}>
                        <Button
                            variant="outlined"
                            style={{ width: "50%" }}
                            onClick={() => navigate("/")}
                        >
                            Cancel
                        </Button>
                        {resetingPassword ? (
                            <Box width="50%" height="100%">
                                <LoadingIndicator size="30px" />
                            </Box>
                        ) : (
                            <Button
                                size="large"
                                type="submit"
                                variant="contained"
                                style={{ width: "50%", height: "100%" }}
                                disabled={passwordIssue != null}
                            >
                                Reset
                            </Button>
                        )}
                    </ButtonGroup>
                </form>
            </Box>
        </Box>
    </div>
  );
}
```

RequesetPasswordReset.tsx

```
import  {  useEffect, useState } from "react";
import logoWhite from "../../../BarflyLogoWhite.png";
import { Auth } from "aws-amplify";
import { TextField, Box, ButtonGroup } from "@mui/material";
import Button from "@mui/material/Button";
```

```
import "../../../styles/auth.css";
import LoadingIndicator from "../../LoadingIndicator";
import { useNavigate } from "react-router";
import prevDef from "../../../decorators/prevDef";
import { useSnackbar } from "notistack";
//import { useParams } from "react-router";

export default function ChangePassword() {
    const [oldPassword, setOldPassword] = useState(null);
    const [password, setPassword] = useState(null);
    const [password2, setPassword2] = useState(null);
    const [message, setMessage] = useState("");
    const [passwordIssue, setPasswordIssue_raw] = useState("");
    const { enqueueSnackbar, closeSnackbar } = useSnackbar();
    function setPasswordIssue(value) {
        setPasswordIssue_raw(value);
        setMessage(value);
    }

    useEffect(() => {
        if (password === null || password2 === null) {
            setPasswordIssue("");
        } else if (password !== password2) {
            setPasswordIssue("Passwords Do Not Match");
        } else if (password === "") {
            setPasswordIssue("Password is Blank");
        } else {
            setPasswordIssue(null);
        }
    }, [password, password2]);

    //const [email, setEmail] = useState("");
    const TEXTFIELD_SPACING = "2ch";

    // const [requestingReset, setRequestingReset] = useState(false);
    const [resetingPassword, setResetingPassword] = useState(false);

    const navigate = useNavigate();

    async function resetPassword() {
        try {
            if (passwordIssue !== null) {
                return;
            }
            setResetingPassword(true);
            const currentUser = await Auth.currentAuthenticatedUser();
            await Auth.changePassword(
                currentUser,
                oldPassword,
                password
            );
            enqueueSnackbar("Password Reset Successfuly", {
                autoHideDuration: 5000,
            });
            navigate("/");
        } catch (err) {
            setMessage(err.message);
        } finally {
            setResetingPassword(false);
        }
    }
```

```
return (
  <div>
    <img src={logoWhite} className="App-logo" alt="logo" />

    <h2>Change Password</h2>
    <Box display="flex" flexDirection="column" alignItems="center">
      <Box width="min(100%, 60ch)">

        <form
          onSubmit={prevDef(resetPassword)}
          style={{
            display: "flex",
            flexDirection: "column",
            alignItems: "center",
            width: "100%",
          }}
        >
          <TextField
          label="Old Password"
          value={oldPassword ?? ""}
          type="password"
          onChange={(e) => setOldPassword(e.target.value)}
          style={{
            marginBottom: TEXTFIELD_SPACING,
            width: "100%",
          }}
          required
          autoFocus
          autoComplete="off"
        />
          <TextField
            label="New Password"
            value={password ?? ""}
            onChange={(e) => setPassword(e.target.value)}
            type="password"
            style={{
              marginBottom: TEXTFIELD_SPACING,
              width: "100%",
            }}
            required
            autoComplete="off"
          />
          <TextField
            label="Confirm New Password"
            value={password2 ?? ""}
            onChange={(e) => setPassword2(e.target.value)}
            type="password"
            style={{
              width: "100%",
            }}
            required
          />
          <Box height="2em">{message}</Box>
          <ButtonGroup style={{ width: "100%", height: "5ch" }}>
            <Button
              variant="outlined"
              style={{ width: "50%" }}
              onClick={() => navigate("/")}
            >
              Cancel
```

```
                    </Button>
                    {resetingPassword ? (
                        <Box width="50%" height="100%">
                            <LoadingIndicator size="30px" />
                        </Box>
                    ) : (
                        <Button
                            size="large"
                            type="submit"
                            variant="contained"
                            style={{ width: "50%", height: "100%" }}
                            disabled={passwordIssue != null}
                        >
                            Reset
                        </Button>
                    )}
                </ButtonGroup>
            </form>
        </Box>
    </Box>
  </div>
 );
}
```

ResetPassword.tsx

```
import  {  useEffect, useState } from "react";
import logoWhite from "../../../BarflyLogoWhite.png";
import { Auth } from "aws-amplify";
import { TextField, Box, ButtonGroup } from "@mui/material";
import Button from "@mui/material/Button";
import "../../../styles/auth.css";
//import prevDef from "../../../decorators/prevDef";
import LoadingIndicator from "../../LoadingIndicator";
import { useNavigate } from "react-router";
import prevDef from "../../../decorators/prevDef";
import { useSnackbar } from "notistack";
//import { useParams } from "react-router";

export default function ResetPassword({ email }: { email: string }) {
  const [code, setCode] = useState("");
  const [password, setPassword] = useState(null);
  const [password2, setPassword2] = useState(null);
  const [message, setMessage] = useState("");
  const [passwordIssue, setPasswordIssue_raw] = useState("");
  const { enqueueSnackbar, closeSnackbar } = useSnackbar();
  function setPasswordIssue(value) {
    setPasswordIssue_raw(value);
    setMessage(value);
  }

  useEffect(() => {
    if (password === null || password2 === null) {
      setPasswordIssue("");
    } else if (password !== password2) {
      setPasswordIssue("Passwords Do Not Match");
    } else if (password === "") {
      setPasswordIssue("Password is Blank");
    } else {
```

```
        setPasswordIssue(null);
    }
}, [password, password2]);

//const [email, setEmail] = useState("");
const TEXTFIELD_SPACING = "2ch";

// const [requestingReset, setRequestingReset] = useState(false);
const [resetingPassword, setResetingPassword] = useState(false);

const navigate = useNavigate();

async function resetPassword() {
    try {
        if (passwordIssue !== null) {
            return;
        }
        setResetingPassword(true);
        await Auth.forgotPasswordSubmit(email, code, password);
        enqueueSnackbar("Password Reset Successfuly", {
            autoHideDuration: 5000,
        });
        navigate("/");
    } catch (err) {
        setMessage(err.message);
    } finally {
        setResetingPassword(false);
    }
}

return (
    <div>
        <img src={logoWhite} className="App-logo" alt="logo" />

        <h2>Reset Password</h2>
        <Box display="flex" flexDirection="column" alignItems="center">
            <Box width="min(100%, 60ch)">
                <TextField
                    label="code"
                    value={code ?? ""}
                    onChange={(e) => setCode(e.target.value)}
                    style={{
                        marginBottom: TEXTFIELD_SPACING,
                        width: "100%",
                    }}
                    required
                    autoFocus
                    autoComplete="off"
                />
                <TextField
                    label="email"
                    value={email ?? ""}
                    disabled
                    style={{
                        marginBottom: TEXTFIELD_SPACING,
                        width: "100%",
                    }}
                />

                <form
                    onSubmit={prevDef(resetPassword)}
```

```
                    style={{
                        display: "flex",
                        flexDirection: "column",
                        alignItems: "center",
                        width: "100%",
                    }}
                >
                    <TextField
                        label="New Password"
                        value={password ?? ""}
                        onChange={(e) => setPassword(e.target.value)}
                        type="password"
                        style={{
                            marginBottom: TEXTFIELD_SPACING,
                            width: "100%",
                        }}
                        required
                    />
                    <TextField
                        label="Confirm New Password"
                        value={password2 ?? ""}
                        onChange={(e) => setPassword2(e.target.value)}
                        type="password"
                        style={{
                            width: "100%",
                        }}
                        required
                    />
                    <Box height="2em">{message}</Box>
                    <ButtonGroup style={{ width: "100%", height: "5ch" }}>
                        <Button
                            variant="outlined"
                            style={{ width: "50%" }}
                            onClick={() => navigate("/")}
                        >
                            Back To Signin
                        </Button>
                        {resetingPassword ? (
                            <Box width="50%" height="100%">
                                <LoadingIndicator size="30px" />
                            </Box>
                        ) : (
                            <Button
                                size="large"
                                type="submit"
                                variant="contained"
                                style={{ width: "50%", height: "100%" }}
                                disabled={passwordIssue != null}
                            >
                                Reset
                            </Button>
                        )}
                    </ButtonGroup>
                </form>
            </Box>
        </Box>
    </div>
  );
}
```

ResetPasswordPage.tsx

```
import {useParams} from "react-router";
import ResetPassword from "./ResetPassword";

export default function ResetPasswordPage(){
    const params = useParams();
    return <ResetPassword email={params.email}/>
}
```

PreDef.ts

```
export default function prevDef<E>(
    eventHandler: E
): E {
    //@ts-ignore
    return function (event, ...rest) {
        if (typeof (event as any).preventDefault === "function") {
            (event as any).preventDefault();
        }
        (event as any).preventDefault();
        //@ts-ignore
        return eventHandler(event, ...rest);
    };
}
```

Barfly-E Code

App.tsx

```
import { StatusBar } from "expo-status-bar";
import React from "react";
import { SafeAreaProvider } from "react-native-safe-area-context";
import useColorScheme from "./hooks/useColorScheme";
import Navigation from "./navigation";
import Amplify from "aws-amplify";
import config from "./src/aws-exports";
import { DefaultTheme, Provider as PaperProvider } from "react-native-paper";
import reducer, { initialState } from "./src/state/reducer";
import { StateProvider } from "./src/state/StateProvider";

Amplify.configure(config);

declare global {
  namespace ReactNativePaper {
    interface ThemeColors {
      myColor: "#fff";
    }

    interface Theme {}
```

```
  }
}

const theme = {
  ...DefaultTheme,
  colors: {
    ...DefaultTheme.colors,
    myColor: "#fff",
  },
};



export default function App() {
  const colorScheme = useColorScheme();

  return (
    <SafeAreaProvider>
      <StateProvider initialState={initialState} reducer={reducer}>
        <PaperProvider>
          <>
            <Navigation colorScheme={colorScheme} />

            <StatusBar />
          </>
        </PaperProvider>
      </StateProvider>
    </SafeAreaProvider>
  );
}
```

CreateCommon.tsx

```
import React from "react";
import { Text, View } from "../../../components/Themed";
import { Button, TextInput, Headline } from "react-native-paper";
import { API, graphqlOperation } from "aws-amplify";
import { createBar, createMenu } from "../../graphql/mutations";
import AsyncStorage from "@react-native-async-storage/async-storage";
import UploadImageToS3WithReactS3 from "./UploadImageToS3WithReactS3";
import { ScrollView } from "react-native";


function CreateCommon(props: any) {
  const [name, setName] = React.useState("");
  const [email, setEmail] = React.useState("");
  const [phone, setPhone] = React.useState("");
  const [bio, setBio] = React.useState("");

  const storeBar = async (value: any) => {
    try {
      const jsonValue = JSON.stringify(value);
      await AsyncStorage.setItem("bar", jsonValue);
    } catch (e) {
      console.log(e);
    }
  };

  async function CreateBar() {
    const payload = {
      name: name,
      email: email,
```

```
    // Have to change schema from AWSPhone to just a string
    //phone: phone,,
    bio: bio,
  };
  try {
    const res = API.graphql(
      graphqlOperation(createBar, {
        input: payload,
      })
    );

    const barPromise:any = await res;

    storeBar(barPromise.data.createBar)

    var payload2 = {
      barID: barPromise.data.createBar.id
    }

    try {
      const res = API.graphql(
        graphqlOperation(createMenu, {
          input: payload2,
        })
      );
     const menuPromise =  await res;
     console.log(menuPromise)
     props.SetMenu(menuPromise.data.createMenu.id)
    } catch (err) {
      console.log(err);
    }

  } catch (err) {
    console.log(err);
  }
  props.nextStep();
}
return (
  <ScrollView>
  <View
    style={{
      display: "flex",
      flexDirection: "column",
      alignItems: "center",
      width: "100%",
    }}
  >
    <Headline style={{ margin: 10 }}>
      Enter Your Establishment's Details
    </Headline>
    <TextInput
      onChangeText={(value) => setName(value)}
      value={name}
      label="Establishment's Name"
      style={{ width: "50%", margin: 10 }}
      autoComplete={null}
    />
    <TextInput
      onChangeText={(value) => setEmail(value)}
      value={email}
      label="Email"
```

```
        placeholder="bar@barfly.llc"
        style={{ width: "50%", margin: 10 }}
        autoComplete={null}
      />
      <TextInput
        onChangeText={(value) => setPhone(value)}
        value={phone}
        label="Phone"
        placeholder="(123) 456-7890"
        style={{ width: "50%", margin: 10 }}
        autoComplete={null}
      />
      <TextInput
        onChangeText={(value) => setBio(value)}
        value={bio}
        label="Establishment's Description"
        style={{ width: "50%", height: 150, margin: 10 }}
        autoComplete={null}
        multiline={true}
      />

      <Button
        style={{ width: "50%", margin: 20 }}
        mode="contained"
        onPress={CreateBar}
        disabled={name === "" || email === "" || phone === "" || bio === ""}
      >
        Next
      </Button>

      <UploadImageToS3WithReactS3 />
    </View>
    </ScrollView>
  );
}

export default CreateCommon;
```

CreateEmployees.tsx

```
import React from "react";
import { View,  ScrollView } from "react-native";
import { Button, TextInput, Headline, Chip } from "react-native-paper";
import { API, graphqlOperation } from "aws-amplify";
import { createEmployee, deleteEmployee } from "../../graphql/mutations";
import { listEmployees } from "../../graphql/queries";
import AsyncStorage from "@react-native-async-storage/async-storage";

// Create emplyoees based on bar id
function CreateEmployees(props: any) {
  const [name, setName] = React.useState("");
  const [email, setEmail] = React.useState("");
  const [employees, setEmployees] = React.useState([]);

  const getBar = async () => {
    try {
      const jsonValue = await AsyncStorage.getItem("bar");

      if (jsonValue !== null) {
          return JSON.parse(jsonValue)
```

```
    } else {
        console.log("bar not found")
        return null
    }
  } catch (e) {
    // error reading value
  }
};


async function addEmployee() {

  const bar = await getBar()
  try {
    const payload = {
      name: name,
      email: email,
      barID: bar.id,
      admin: false,
    };

    const res = API.graphql(
      graphqlOperation(createEmployee, {
        input: payload,
      })
    );

    const employeePromise: any = await res;

    if (employeePromise.data) {
      setEmployees([...employees, employeePromise.data.createEmployee]);
      setName("");
      setEmail("");
    }
  } catch (err) {
    console.log(err);
  }
}

async function DeleteEmployee(employee: any) {
  const payload = {
    id: employee.id,
    _version: employee._version,
  };
  const res = API.graphql({
    query: deleteEmployee,
    variables: { input: payload },
  });
  const deletePromise: any = await res;

  setEmployees(employees.filter((item) => item.id !== employee.id));
  return deletePromise;
}

async function getEmployees() {
  const bar = await getBar()
  try {
    const ordersPromise = API.graphql(
      graphqlOperation(listEmployees, {
        filter: { barID: { eq: bar.id }},
      })
```

```
    );
    const response = await ordersPromise;
    setEmployees(response.data.listEmployees.items.filter((item:any) => item._deleted === null))
  } catch (err) {
    console.log(err);
  }
}

React.useEffect(() => {
  getEmployees()
}, [])


return (
  <ScrollView>
  <View style={{
      display: "flex",
      flexDirection: "column",
      alignItems: "center",
      width: "100%",
    }}>
    <View
      style={{
        display: "flex",
        flexDirection: "row",
        alignItems: "center",
        width: "100%",
      }}
    >
      <View
        style={{
          display: "flex",
          flexDirection: "column",
          alignItems: "center",
          width: "20%",
        }}
      >
        {employees.map((employee) => {
          return (
            <Chip
              style={{ margin: 10, height:40 }}
              icon="delete"
              key={Math.random() + ""}
              onPress={() => DeleteEmployee(employee)}
            >
              {employee.name}
            </Chip>
          );
        })}
      </View>
      <View
        style={{
          display: "flex",
          flexDirection: "column",
          alignItems: "center",
          width: "80%",
        }}
      >
        <Headline style={{ margin: 10 }}>Add Employees</Headline>

        <TextInput
```

```
          autoComplete={null}
          label="Name"
          value={name}
          placeholder="Name"
          onChangeText={(text) => setName(text)}
          style={{ width: "50%", margin: 10 }}
        />
        <TextInput
          autoComplete={null}
          label="Email"
          placeholder="Email"
          value={email}
          onChangeText={(text) => setEmail(text)}
          style={{ width: "50%", margin: 10 }}
        />

        <Button
          style={{ width: "50%", margin: 10 }}
          mode="contained"
          onPress={addEmployee}
          disabled={email === "" || name === ""}
        >
          Add
        </Button>
        <Button
      style={{ width: "50%", margin: 10 }}
      mode="contained"
      onPress={props.nextStep}
    >
      Finish
    </Button>
    <Button
      style={{ width: "50%", margin: 10 }}
      mode="contained"
      onPress={props.prevStep}
    >
      Back
    </Button>
      </View>
    </View>

   </View>
   </ScrollView>
  );
}

export default CreateEmployees;
```

CreateMenu.tsx

```
import React, { useState} from 'react';
import { View } from "../../../components/Themed";
import { Button, TextInput, Chip } from "react-native-paper";
import DropDownPicker from "react-native-dropdown-picker";
import { API, graphqlOperation } from "aws-amplify";
import { createFood, createCocktail, createBeer, createShot } from "../../graphql/mutations";

function CreateMenu(props:any) {
  const [itemPrice, SetPrice] = useState("")
  const [itemName, SetName] = useState("")
```

```
const [type, SetType] = useState("")
const [menuID, setMenuID] = useState()
const [open, SetOpen] = React.useState(false)
const [items, SetItems] = React.useState([]);
const [typelist, setTypeList] = React.useState([
  {label: 'Beer', value: 'Beer'},
  {label: 'Shot', value: 'Shot'},
  {label: 'Cocktail', value: 'Cocktail'},
  {label: 'Food', value: 'Food'},
])

const createItem = async () => {

    var payload = {
      name: itemName,
      price: itemPrice,
      menuID: props.menu
   }

   console.debug(payload)
   try {
      var update
      var updateResponse

      if(type=="Food"){
         update = API.graphql(graphqlOperation(createFood, {
            input: payload
         }))
         updateResponse = await update
         if (updateResponse.data) {
          SetItems([...items, updateResponse.data.createFood]);
         }
      }
      else if(type=="Beer"){
         update = API.graphql(graphqlOperation(createBeer, {
            input: payload
         }))
         updateResponse = await update
         if (updateResponse.data) {
          SetItems([...items, updateResponse.data.createBeer]);
         }
      }
      else if(type=="Cocktail"){
         update = API.graphql(graphqlOperation(createCocktail, {
            input: payload
         }))
         updateResponse = await update
         if (updateResponse.data) {
          SetItems([...items, updateResponse.data.createCocktail]);
         }
      }
      else if(type=="Shot"){
         update = API.graphql(graphqlOperation(createShot, {
            input: payload
         }))
         updateResponse = await update
         if (updateResponse.data) {
          SetItems([...items, updateResponse.data.createShot]);
         }

      }
```

```
        if (updateResponse.data) {
          SetName("");
          SetPrice("");
          SetType("");
        }

    } catch (err) {
        console.log(err)
    }

}

 return (
   <View style={{
     display: "flex",
     flexDirection: "column",
     alignItems: "center",
     width: "100%",
   }}>
   <View
     style={{
       display: "flex",
       flexDirection: "row",
       alignItems: "center",
       width: "100%",
     }}
   >
     <View
       style={{
         display: "flex",
         flexDirection: "column",
         alignItems: "center",
         width: "20%",
       }}
     >
       {items.map((item) => {
         return (
           <Chip
             style={{ margin: 10 }}
             icon="delete"
             key={Math.random() + ""}
             //onPress={() => DeleteItem(item)}
           >
             {item.name}
           </Chip>
         );
       })}
     </View>
     <View
       style={{
         display: "flex",
         flexDirection: "column",
         alignItems: "center",
         width: "80%",
       }}
     >
       <DropDownPicker
         placeholder='Select Item Type'
         setValue={SetType}
         setItems={setTypeList}
```

```
          setOpen={SetOpen}
          value={type}
          items={typelist}
          open={open}
          style={{ width: "50%", margin: 10, alignSelf:"flex-end", backgroundColor: 'rgb(187, 134, 252)' }}
          />

        <TextInput
          value={itemName}
          onChangeText={(e) => SetName(e)}
          label="Item Name"
          autoComplete={null}
          style={{ width: "50%", margin: 10, alignSelf:"flex-end"  }}
        />
        <TextInput
          onChangeText={(e) => SetPrice(e)}
          value={itemPrice}
          keyboardType="numeric"
          label="Item Price"
          autoComplete={null}
          style={{ width: "50%", margin: 10,  alignSelf:"flex-end" }}
         />

        <Button onPress={createItem} style={{alignSelf:"flex-end"}}>Create</Button>
        </View>
      </View>
      <Button
        style={{ width: "50%", margin: 10 }}
        mode="contained"
        onPress={props.nextStep}
        disabled={itemPrice === "" || itemName === "" || type === ""}
      >
        Finish
      </Button>
      <Button
        style={{ width: "50%", margin: 10 }}
        mode="contained"
        onPress={props.prevStep}
      >
        Back
      </Button>
    </View>
    );
}

export default CreateMenu;
```

EditBar.tsx

```
import React from "react";
import { Text, View } from "../../../components/Themed";
import { Button, TextInput, Headline } from "react-native-paper";
import { API, graphqlOperation } from "aws-amplify";
import { updateBar } from "../../graphql/mutations";
import UploadImageToS3WithReactS3 from "./UploadImageToS3WithReactS3"
import AsyncStorage from "@react-native-async-storage/async-storage";

function EditBar() {
    const [name, setName] = React.useState("");
    const [email, setEmail] = React.useState("");
```

```
const [phone, setPhone] = React.useState("");
const [bio, setBio] = React.useState("");

const getBar = async () => {
   try {
     const jsonValue = await AsyncStorage.getItem("bar");

     if (jsonValue !== null) {
        return JSON.parse(jsonValue)
     } else {
        console.log("bar not found")
        return null
     }
   } catch (e) {
     // error reading value
   }
 };

async function editBar() {

   const bar  = await getBar()

  const payload = {
   id: bar.id,
   _version: bar._version,
   name: name,
   email: email,
   // Have to change schema from AWSPhone to just a string
   //phone: phone,
   bio: bio,
  };
  try {
   const res = API.graphql(
     graphqlOperation(updateBar, {
       input: payload,
     })
   );

   const barPromise:any = await res;

  } catch (err) {
   console.log(err);
  }
}
return (
  <View
   style={{
     display: "flex",
     flexDirection: "column",
     alignItems: "center",
     width: "100%",
   }}
  >
   <Headline style={{ margin: 10 }}>
      Update Bar's Details
   </Headline>
   <TextInput
     onChangeText={(value) => setName(value)}
     value={name}
     label="Establishment's Name"
     style={{ width: "50%", margin: 10 }}
```

```
          autoComplete={null}
        />
        <TextInput
          onChangeText={(value) => setEmail(value)}
          value={email}
          label="Email"
          placeholder="bar@barfly.llc"
          style={{ width: "50%", margin: 10 }}
          autoComplete={null}
        />
        <TextInput
          onChangeText={(value) => setPhone(value)}
          value={phone}
          label="Phone"
          placeholder="(123) 456-7890"
          style={{ width: "50%", margin: 10 }}
          autoComplete={null}
        />
        <TextInput
          onChangeText={(value) => setBio(value)}
          value={bio}
          label="Establishment's Description"
          style={{ width: "50%", height: 150, margin: 10 }}
          autoComplete={null}
          multiline={true}
        />

        <UploadImageToS3WithReactS3/>

        <Button
          style={{ width: "50%", margin: 20 }}
          mode="contained"
          onPress={editBar}
          disabled={name === "" || email === "" || phone === "" || bio === ""}
        >
          Next
        </Button>
      </View>
    );
}

export default EditBar
```

MultiStepForm.tsx

```
import React from 'react';
import SignIn from '../../../components/auth/SignIn';
import CreateCommon from './CreateCommon';
import CreateEmployees from './CreateEmployees';
import CreateMenu from './CreateMenu';


function MultiStepForm({navigation}) {
    const [step, setStep] = React.useState(1);
    const [menu, SetMenu] = React.useState();
    const [payload, setPayload] = React.useState({
        name: '',
        email: '',  // awsEmail
        phone: '',  // awsPhone
        payment: {
```

```
            routingNum: '',
            accountNum: ''
        },    // json
        address: {
            address: ''
        }, //json
        bio: ''
    });

    const nextStep = () => {
        setStep(step + 1);
    }

    const prevStep = () => {
        setStep(step - 1);
    }

    const handleInputData = (value, key) => {

        setPayload(prevState => ({
            ...prevState,
            [key]: value
        }));
    }

    switch (step) {
        case 1:
            return (
                <CreateCommon SetMenu={SetMenu} nextStep={nextStep} handleInputData={handleInputData} payload={payload} />
            );
        case 2:
            return (
                <CreateMenu menu={menu} nextStep={nextStep} prevStep={prevStep} setPayload={setPayload} />
            );
        case 3:
            return (
                <CreateEmployees nextStep={nextStep} prevStep={prevStep} setPayload={setPayload} />
            );
        case 4:
            return <SignIn />
        default:
            return <></>
    }

}

export default MultiStepForm;
```

QRCodeGenerator.tsx

```
import { StatusBar } from 'expo-status-bar';
import React, { useEffect} from 'react';
import { Platform, StyleSheet } from 'react-native';
import { Button } from 'react-native-paper';
import {Auth} from 'aws-amplify'
import QRCode from "react-native-qrcode-svg";
import { Text, View } from '../../../components/Themed';
import AsyncStorage from "@react-native-async-storage/async-storage";

const styles = StyleSheet.create({
```

```
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  title: {
    fontSize: 20,
    fontWeight: 'bold',
  },
  separator: {
    marginVertical: 30,
    height: 1,
    width: '80%',
  },
});

function QRCodeGenerator() {
  const [showCode, setShowCode] = React.useState(false);
  const [bar, setBar] = React.useState();

  async function signOut() {
      try {
        await Auth.signOut();
      } catch (error) {
        console.log(error);
      }
    }

    const getBar = async () => {
      try {
        const jsonValue = await AsyncStorage.getItem("bar");

        if (jsonValue !== null) {
          return JSON.parse(jsonValue)
        } else {
          console.log("bar not found")
          return null
        }
      } catch (e) {
        console.log(e)
      }
    };

  useEffect(() => {
    const fetchBar = async () => {
      const barr = await getBar()
      setBar(barr);
    }
    fetchBar()
  }, [])


  function toggleQR(){
    setShowCode(!showCode)
  }

  function showQR(){

    let logoFromFile = require('../../../BarflyLogo.png');
    let website = "www.barfly.llc/"
    let link = website + bar.id +"/menu"
```

```
    return <QRCode value={link} logo={logoFromFile} logoSize={100} size={300}/>
  }

  return (
    <View style={styles.container}>
      <Button onPress={signOut}>Log Out</Button>
      <View style={styles.separator} lightColor="#eee" darkColor="rgba(255,255,255,0.1)" />
      <Button onPress={toggleQR}>{showCode ? "Hide" : "Show"} QR code</Button>
      {showCode ? showQR() : null}
      <StatusBar style={Platform.OS === 'ios' ? 'light' : 'auto'} />
    </View>
  );
}


export default QRCodeGenerator
```

UploadImageToS3WithReactS3.js

```
import React, {useState} from 'react';
// Import required components
import {
  StyleSheet,
  Image,
  Platform,
  ScrollView
} from 'react-native';
import { Button } from 'react-native-paper';
import { Text, View } from "../../../components/Themed";
import {RNS3} from 'react-native-aws3';
import AsyncStorage from "@react-native-async-storage/async-storage";
import * as ImagePicker from 'expo-image-picker';
import { updateBar } from '../../graphql/mutations';
import { API, graphqlOperation } from "aws-amplify";
import { getBar } from '../../graphql/queries';

const UploadImageToS3WithReactS3 = () => {
  const [filePath, setFilePath] = useState({});
  const [uploadSuccessMessage, setUploadSuccessMessage] = useState('');

  const [image, setImage] = useState(null);

  const getVersion = async () => {

    const bar = await getBarObj()
    try {
        const userRes = API.graphql(
            graphqlOperation(getBar, {
                id: bar.id
            })
        );

        const profile = await userRes;
        console.log(profile.data.getBar)
        return profile.data.getBar._version;
    } catch (err) {
        console.log(err);
    }
};
```

```
async function updateBarImg(location) {
  const version = await getVersion();
  const bar = await getBarObj();
  const res = API.graphql(
     graphqlOperation(updateBar, {
        input: {
           id: bar.id,
           profileImg: JSON.stringify({ img: location }),
           _version: version,
        },
     })
  );

  const barResponse = await res;
  return barResponse;
}

  const getBarObj = async () => {
    try {
      const jsonValue = await AsyncStorage.getItem("bar");

      if (jsonValue !== null) {
         return JSON.parse(jsonValue)
      } else {
         console.log("bar not found")
         return null
      }
    } catch (e) {
      // error reading value
    }
  };

  React.useEffect(() => {
    (async () => {
      if (Platform.OS !== 'web') {
        const { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();
        if (status !== 'granted') {
          alert('Sorry, we need camera roll permissions to make this work!');
        }
      }
    })();
  }, []);

  const pickImage = async () => {
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: "Images",
      allowsEditing: true,
      aspect: [4, 3],
      quality: 1,
    });

    if (!result.cancelled) {
      setImage(result);

    }
  };


  const uploadFile = async () => {
    const bar = await getBarObj()
    if (Object.keys(image).length == 0) {
```

```
    alert('Please select image first');
    return;
  }
  RNS3.put(
    {
      uri: image.uri,
      name: bar.id,
      type:  "image/jpeg",
    },
    {
      bucket: 'barfly-pics',
      region: 'us-west-2',
      accessKey: 'AKIA33JESCSVFLWNP6VN',
      secretKey: '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',
      successActionStatus: 201,
    },
  )
    .progress((progress) =>
      setUploadSuccessMessage(
        `Uploading: ${progress.loaded / progress.total} (${
          progress.percent
        }%)`,
      ),
    )
    .then((response) => {
      if (response.status !== 201)
        alert('Failed to upload image to S3');

      updateBarImg(response.body.postResponse.location)
      setFilePath('');
      setUploadSuccessMessage(
        `Uploaded Successfully`,
      );

    });
};

return (
  <ScrollView>
    <View>
      {image ? (
        <View>
          <Text style={styles.textStyle}>
            {filePath.uri}
          </Text>
          <Button
            onPress={uploadFile}>
          Upload
          </Button>
        </View>
      ) : null}

      <Button
        activeOpacity={0.5}
        onPress={pickImage}>
        Pick photo
      </Button>
      {image && <Image source={{ uri: image.uri }} style={{ width: 200, height: 200 }} />}
      {uploadSuccessMessage ? (
        <Text style={styles.textStyleGreen}>
          {uploadSuccessMessage}
```

```
          </Text>
        ) : null}
      </View>
    </ScrollView>
  );
};

export default UploadImageToS3WithReactS3;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 10,
    backgroundColor: '#fff',
    height:300
  },
  titleText: {
    fontSize: 22,
    fontWeight: 'bold',
    textAlign: 'center',
    paddingVertical: 20,
  },
  textStyle: {
    padding: 10,
    color: 'black',
    textAlign: 'center',
  },
  textStyleGreen: {
    padding: 10,
    color: 'green',
  },
  textStyleWhite: {
    padding: 10,
    color: 'white',
  },
  buttonStyle: {
    alignItems: 'center',
    backgroundColor: 'orange',
    marginVertical: 10,
    width: '100%',
  },
  buttonStyleGreen: {
    alignItems: 'center',
    backgroundColor: 'green',
    marginVertical: 10,
    width: '100%',
  },
  imageStyle: {
    flex: 1,
    width: '100%',
    height: '100%',
    resizeMode: 'contain',
    margin: 5,
  },
});
```

Menu.js

```
import React, { useState, useEffect } from "react";
import { API, graphqlOperation } from "aws-amplify";
import {
  listFoods,
  listBeers,
  listCocktails,
  listShots,
  listMenus,
} from "../../graphql/queries";
import {
  createFood,
  createCocktail,
  createBeer,
  createShot,
} from "../../graphql/mutations";
import { Text, View } from "../../../components/Themed";
import MenuItem from "./MenuItem";
import { StyleSheet } from "react-native";
import AsyncStorage from "@react-native-async-storage/async-storage";
import DropDownPicker from "react-native-dropdown-picker";
import { Button, TextInput, Chip } from "react-native-paper";

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: "center",
    justifyContent: "center",
    display: "flex",
  },
  box: {
    width: "100%",
  },
  orderQueueContainer: {
    flex: 1,
    alignItems: "center",
    justifyContent: "center",
    textAlign: "left",
    display: "flex",
    flexDirection: "column",
    // flexWrap: 'wrap'
  },
  title: {
    fontSize: 30,
    fontWeight: "bold",
    margin: 20,
  },
  subtitle: {
    fontSize: 23,
    fontWeight: "bold",
  },
  separator: {
    marginVertical: 30,
    height: 1,
    width: "80%",
  },
});

const Menu = () => {
  const [menuID, setMenuID] = useState("");
```

```
const [FoodItems, setFoodItems] = useState([]);
const [BeerItems, setBeerItems] = useState([]);
const [ShotItems, setShotItems] = useState([]);
const [CocktailItems, setCocktailItems] = useState([]);
const [isLoading, setIsLoading] = useState(true);
const [itemName, SetName] = useState("");
const [itemPrice, SetPrice] = useState("");
const [open, SetOpen] = React.useState(false);
const [items, SetItems] = React.useState([]);
const [type, SetType] = useState("");
const [typelist, setTypeList] = React.useState([
  { label: "Beer", value: "Beer" },
  { label: "Shot", value: "Shot" },
  { label: "Cocktail", value: "Cocktail" },
  { label: "Food", value: "Food" },
]);

const getBar = async () => {
  try {
    const jsonValue = await AsyncStorage.getItem("bar");

    if (jsonValue !== null) {
      return JSON.parse(jsonValue);
    } else {
      console.log("bar not found");
      return null;
    }
  } catch (e) {
    // error reading value
  }
};

useEffect(() => {
  const listMenu = async () => {
    try {
      var menuPromise = API.graphql(
        graphqlOperation(listFoods, { filter: { menuID: { eq: menuID } } })
      );
      var response = await menuPromise;
      setFoodItems(response.data.listFoods.items.filter((item) => item._deleted === null));

      menuPromise = API.graphql(
        graphqlOperation(listBeers, { filter: { menuID: { eq: menuID } } })
      );
      response = await menuPromise;
      setBeerItems(response.data.listBeers.items.filter((item) => item._deleted === null));

      menuPromise = API.graphql(
        graphqlOperation(listCocktails, {
          filter: { menuID: { eq: menuID } },
        })
      );
      response = await menuPromise;
      setCocktailItems(response.data.listCocktails.items.filter((item) => item._deleted === null));

      menuPromise = API.graphql(
        graphqlOperation(listShots, { filter: { menuID: { eq: menuID } } })
      );
      response = await menuPromise;
      setShotItems(response.data.listShots.items.filter((item) => item._deleted === null));
```

```
      setIsLoading(false);
    } catch (err) {
      console.log(err);
    }
  };

  return listMenu();
}, [menuID]);

useEffect(() => {
  const getMenuID = async () => {
    const bar = await getBar();

    try {
      var menuPromise = API.graphql(
        graphqlOperation(listMenus, { filter: { barID: { eq: bar.id } } })
      );
      var response = await menuPromise;
      setMenuID(response.data.listMenus.items[0].id);
    } catch (err) {
      console.log(err);
    }
  };
  getMenuID();
}, []);

const createItem = async () => {
  var payload = {
    name: itemName,
    price: itemPrice,
    menuID: menuID,
  };

  console.debug(payload);
  try {
    var update;
    var updateResponse;

    if (type == "Food") {
      update = API.graphql(
        graphqlOperation(createFood, {
          input: payload,
        })
      );
      updateResponse = await update;
      if (updateResponse.data) {
        SetItems([...items, updateResponse.data.createFood]);
      }
    } else if (type == "Beer") {
      update = API.graphql(
        graphqlOperation(createBeer, {
          input: payload,
        })
      );
      updateResponse = await update;
      if (updateResponse.data) {
        SetItems([...items, updateResponse.data.createBeer]);
      }
    } else if (type == "Cocktail") {
      update = API.graphql(
        graphqlOperation(createCocktail, {
```

```
          input: payload,
        })
      );
      updateResponse = await update;
      if (updateResponse.data) {
        SetItems([...items, updateResponse.data.createCocktail]);
      }
    } else if (type == "Shot") {
      update = API.graphql(
        graphqlOperation(createShot, {
          input: payload,
        })
      );
      updateResponse = await update;
      if (updateResponse.data) {
        SetItems([...items, updateResponse.data.createShot]);
      }
    }

    if (updateResponse.data) {
      SetName("");
      SetPrice("");
      SetType("");
    }
  } catch (err) {
    console.log(err);
  }
};

return (
  <View style={styles.container}>
    <Text style={styles.title}>Add Menu Items</Text>
    <View
      style={{
        display: "flex",
        flexDirection: "row",
        alignItems: "center",
        width: "100%",
      }}
    >

      <View
        style={{
          display: "flex",
          flexDirection: "column",
          alignItems: "center",
          width: "20%",
        }}
      >
        {items.map((item) => {
          return (
            <Chip
              style={{ margin: 10, height:40 }}
              icon="delete"
              key={Math.random() + ""}
              //onPress={() => DeleteItem(item)}
            >
              {item.name}
            </Chip>
          );
        })}
```

```
  </View>
  <View
    style={{
      display: "flex",
      flexDirection: "column",
      alignItems: "center",
      width: "80%",
    }}
  >
    <DropDownPicker
      placeholder="Select Item Type"
      setValue={SetType}
      setItems={setTypeList}
      setOpen={SetOpen}
      value={type}
      items={typelist}
      open={open}
      style={{
        width: "50%",
        margin: 10,
        alignSelf: "flex-end",
        backgroundColor: "rgb(187, 134, 252)",
      }}
    />

    <TextInput
      value={itemName}
      onChangeText={(e) => SetName(e)}
      label="Item Name"
      autoComplete={null}
      style={{ width: "50%", margin: 10, alignSelf: "flex-end" }}
    />
    <TextInput
      onChangeText={(e) => SetPrice(e)}
      value={itemPrice}
      keyboardType="numeric"
      label="Item Price"
      autoComplete={null}
      style={{ width: "50%", margin: 10, alignSelf: "flex-end" }}
    />

    <Button onPress={createItem} style={{ alignSelf: "flex-end" }}>
      Create
    </Button>
  </View>
</View>

<Text style={styles.title}>Edit Menu</Text>

<View style={styles.box}>
  <Text style={[styles.subtitle, { marginBottom: 100 }]}>Beers</Text>
  <View style={styles.orderQueueContainer}>
    {!isLoading
      ? BeerItems.map((item) => {
          return (
            <MenuItem
              key={item.id + Math.random()}
              item={item}
              type="Beer"
            />
          );
```

```
            })
          : null}
        </View>

        <Text style={[styles.subtitle, { marginTop: 100 }]}>Cocktails</Text>
        <View style={styles.orderQueueContainer}>
          {!isLoading
            ? CocktailItems.map((item) => {
                return (
                  <MenuItem
                    key={item.id + Math.random()}
                    item={item}
                    type="Cocktail"
                  />
                );
              })
            : null}
        </View>

        <Text style={styles.subtitle}>Food</Text>
        <View style={styles.orderQueueContainer}>
          {!isLoading
            ? FoodItems.map((item) => {
                return (
                  <MenuItem
                    key={item.id + Math.random()}
                    item={item}
                    type="Food"
                  />
                );
              })
            : null}
        </View>

        <Text style={styles.subtitle}>Shots</Text>
        <View style={styles.orderQueueContainer}>
          {!isLoading
            ? ShotItems.map((item) => {
                return (
                  <MenuItem
                    key={item.id + Math.random()}
                    item={item}
                    type="Shots"
                  />
                );
              })
            : null}
        </View>
      </View>
    </View>
  );
};

export default Menu;
```

MenuItem.js

```
import React, {useState, useEffect} from 'react';
import { Text, View } from '../../../components/Themed';
import {Button,TextInput} from 'react-native-paper';
import {API, graphqlOperation} from 'aws-amplify';
import {updateFood, updateBeer, updateCocktail, updateShot, deleteFood, deleteBeer, deleteCocktail, deleteShot} from
'../../graphql/mutations';
import { StyleSheet } from 'react-native';

const MenuItem = ({ item, type }) => {

    const [itemPrice, SetPrice] = useState(item.price.toFixed(2))
    const [itemName, SetName] = useState(item.name)

    message: "The variables input contains a field name 'name' that is not defined for input object type 'DeleteBeerInput' "


    async function deleteItem() {
        const payload = {
            id: item.id,
            _version: item._version

        }
        try {
            var update
            var updateResponse

            if(type=="Food"){
                update = API.graphql(graphqlOperation(deleteFood, {
                    input: payload
                }))
                updateResponse = await update
            }
            else if(type=="Beer"){
                update = API.graphql(graphqlOperation(deleteBeer, {
                    input: payload
                }))
                updateResponse = await update
            }
            else if(type=="Cocktail"){
                update = API.graphql(graphqlOperation(deleteCocktail, {
                    input: payload
                }))
                updateResponse = await update
            }
            else if(type=="Shot"){
                update = API.graphql(graphqlOperation(deleteShot, {
                    input: payload
                }))
                updateResponse = await update
            }

        } catch (err) {
            console.log(err)
        }

    }


    const updateItem = async () => {
```

```
    const payload = {
        id: item.id,
        name: itemName,
        price: itemPrice,
        _version: item._version

    }
    try {
        var update
        var updateResponse

        if(type=="Food"){
            update = API.graphql(graphqlOperation(updateFood, {
                input: payload
            }))
            updateResponse = await update
        }
        else if(type=="Beer"){
            update = API.graphql(graphqlOperation(updateBeer, {
                input: payload
            }))
            updateResponse = await update
        }
        else if(type=="Cocktail"){
            update = API.graphql(graphqlOperation(updateCocktail, {
                input: payload
            }))
            updateResponse = await update
        }
        else if(type=="Shot"){
            update = API.graphql(graphqlOperation(updateShot, {
                input: payload
            }))
            updateResponse = await update
        }

    } catch (err) {
        console.log(err)
    }

}

const OnChangeName = (newName) => {
    SetName(newName);
}

const OnChangePrice = (newPrice) => {
    SetPrice(newPrice);
}


const styles = StyleSheet.create({
    orderContainer: {
      margin: 5,
      flexDirection: 'row',
      alignItems: 'center',
      justifyContent: 'center',

    },
    orderQueueContainer: {
    flex: 1,
```

```
    alignItems: 'center',
    justifyContent: 'center',
    display: 'flex',
    flexDirection: 'column'
  },
  title: {
    fontSize: 20,
    fontWeight: 'bold',

  },
  separator: {
    marginVertical: 30,
    height: 1,
    width: '80%',
  },

  time: {
      marginLeft: 15,
  },

  divider: {
      margin: 5
  },

  input: {
      height: 40,
      margin: 12,
     // borderWidth: 1,
      padding: 10,
      width:'50%'
    },
    inputNum: {
      height: 40,
      margin: 12,
    //  borderWidth: 1,
      padding: 10,
    },

});

return (

  <View style={styles.orderContainer}>

      <TextInput
          style={styles.input}
          onChangeText={(e) => OnChangeName(e)}
          value={itemName}
      />

      <TextInput
          style={styles.inputNum}
          onChangeText={(e) => OnChangePrice(e)}
          value={itemPrice}
          keyboardType="numeric"
      />
      <Button onPress={updateItem}>Save</Button>
      <Button onPress={deleteItem}>Delete</Button>
```

```
      </View>
    )
}


export default MenuItem
```

IncomeSummary.js

```
import React, { useState, useEffect } from "react";
import { API, graphqlOperation } from "aws-amplify";
import { listOrders } from "../../graphql/queries";
import { Text, View } from "../../../components/Themed";
import { ScrollView, StyleSheet } from "react-native";
import AsyncStorage from "@react-native-async-storage/async-storage";

const styles = StyleSheet.create({
  scrollView: {
    width: '100%'
  },
  container: {
    flex: 1,
    alignItems: "center",
    justifyContent: "center",
    display: "flex",
    width: "100%",
  },
  summaryContainer: {
    flex: 1,
    alignItems: "center",
    justifyContent: "center",
    display: "flex",
    flexDirection: "column",
    width: "70%",
    height: "auto",
  },
  title: {
    fontSize: 30,
    fontWeight: "bold",
    margin: 20,
  },
  summary: {
    marginVertical: 30,
    height: 1,
    width: "100%",
    height:"auto",
    borderColor: "#6200ee",
    borderBottomWidth: 1,
  },
  content: {
    fontSize: 20,
    padding: 10,
   // color: "#000"
  },
  special: {
    fontStyle: "italic",
    fontWeight: "bold",
  },
});
```

```
const getFilterTime = (filter) => {
  var today = new Date();
  var fullday = 84600000; // number of milliseconds
  if (filter == "yearly") {
    today = new Date(today - 365 * fullday);
  } else if (filter == "monthly") {
    today = new Date(today - 31 * fullday);
  } else if (filter == "weekly") {
    today = new Date(today - 7 * fullday);
  } else if (filter == "daily") {
    today = new Date(today - fullday);
  }

  var month = today.getMonth() + 1 + "";
  if (month.length < 2) {
    month = "0" + month;
  }

  var date = today.getDate() + "";
  if (date.length < 2) {
    date = "0" + date;
  }

  var hours = today.getHours() + "";
  if (hours.length < 2) {
    hours = "0" + hours;
  }

  const time = today.getFullYear() + "-" + month + "-" + date + "T" + hours;
  return time;
};

const getOrderTotal = (orders) => {
  var amount = 0;
  orders?.forEach((order) => {
    var list = JSON.parse(order.items);
    list?.forEach((item) => {
      amount += item.price;
    });
  });
  return amount.toFixed(2);
};

const IncomeSummary = () => {
  const [daily, setDaily] = useState([]);
  const [dailyNum, setDailyNum] = useState([]);
  const [weekly, setWeekly] = useState([]);
  const [weeklyNum, setWeeklyNum] = useState([]);
  const [monthly, setMonthly] = useState([]);
  const [monthlyNum, setMonthlyNum] = useState([]);
  const [yearly, setYearly] = useState([]);
  const [yearlyNum, setYearlyNum] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
  const [bar, setBar] = useState();

  const getBar = async () => {
    try {
      const jsonValue = await AsyncStorage.getItem("bar");

      if (jsonValue !== null) {
```

```
      return JSON.parse(jsonValue);
    } else {
      console.log("bar not found");
      return null;
    }
  } catch (e) {
    // error reading value
  }
};

useEffect(() => {
  const listSummaries = async () => {
    const bar = await getBar();
    try {
      var response_promise = API.graphql(
        graphqlOperation(listOrders, {
          filter: {
            createdAt: { ge: getFilterTime("daily") },
            barID: { eq: bar.id },
          },
        })
      );
      var response = await response_promise;
      setDaily(getOrderTotal(response.data.listOrders.items));
      setDailyNum(response.data.listOrders.items.length);

      response_promise = API.graphql(
        graphqlOperation(listOrders, {
          filter: {
            createdAt: { ge: getFilterTime("weekly") },
            barID: { eq: bar.id },
          },
        })
      );
      response = await response_promise;
      setWeekly(getOrderTotal(response.data.listOrders.items));
      setWeeklyNum(response.data.listOrders.items.length);

      response_promise = API.graphql(
        graphqlOperation(listOrders, {
          filter: {
            createdAt: { ge: getFilterTime("monthly") },
            barID: { eq: bar.id },
          },
        })
      );
      response = await response_promise;
      setMonthly(getOrderTotal(response.data.listOrders.items));
      setMonthlyNum(response.data.listOrders.items.length);

      response_promise = API.graphql(
        graphqlOperation(listOrders, {
          filter: {
            createdAt: { ge: getFilterTime("yearly") },
            barID: { eq: bar.id },
          },
        })
      );
      response = await response_promise;
      setYearly(getOrderTotal(response.data.listOrders.items));
      setYearlyNum(response.data.listOrders.items.length);
```

```
      setIsLoading(false);
    } catch (err) {
      console.log(err);
    }
  };

  listSummaries();
}, [bar]);

return (
  <ScrollView style={styles.scrollView}>
    <View style={styles.container}>
      <Text style={styles.title}>Income Summary</Text>
      {!isLoading ? (
        <View style={styles.summaryContainer}>
          <View style={styles.summary}>
            <Text style={styles.content}>
              Past 24 Hours: <Text style={styles.special}>${daily}</Text> on{" "}
              {dailyNum} orders.{"\n"}
            </Text>
          </View>
          <View style={styles.summary}>
            <Text style={styles.content}>
              Past Week: <Text style={styles.special}>${weekly}</Text> on{" "}
              {weeklyNum} orders. {"\n"}
            </Text>
          </View>
          <View style={styles.summary}>
            <Text style={styles.content}>
              Past Month: <Text style={styles.special}>${monthly}</Text> on{" "}
              {monthlyNum} orders. {"\n"}
            </Text>
          </View>
          <View style={styles.summary}>
            <Text style={styles.content}>
              Past Year: <Text style={styles.special}>${yearly}</Text> on{" "}
              {yearlyNum} orders. {"\n"}
            </Text>
          </View>
        </View>
      ) : null}
    </View>
  </ScrollView>
);
};

export default IncomeSummary;
```

Order.js

```
import React, {useState, useEffect} from 'react';
import { View } from '../../../components/Themed';
import {Card, Snackbar, Caption, Button, Subheading, Divider} from 'react-native-paper';
import { getUser } from '../../../graphql/queries';
import {API, graphqlOperation} from 'aws-amplify';
import {updateOrder, deleteOrder} from '../../../graphql/mutations';
import { StyleSheet } from 'react-native';

const styles = StyleSheet.create({
    orderContainer: {
      margin: 10,
      width: "100%",
      height: "auto",
      //maxHeight: 1000,
    },
    orderQueueContainer: {
    flex: 1,
      alignItems: 'center',
      justifyContent: 'center',
      display: 'flex',
      flexDirection: 'row',
      height:10
    },
    title: {
      fontSize: 20,
      fontWeight: 'bold',

    },
    separator: {
      marginVertical: 30,
      height: 1,
      width: '80%',
    },

    time: {
        marginLeft: 15,
    },

    divider: {
        margin: 5
    },

  });

const Order = ({ order, index, employee, setOrders, orders}) => {

    const [orderItems, setOrderItems] = useState(JSON.parse(order.items))
    const [customer, setcustomer] = useState()
    const [isLoading, setIsLoading] = useState(true)
    const [visible, setVisable] = useState(false)

    const onDismissSnackBar = () => setVisable(false);

    const startOrder = async () => {
        const payload = {
            id: order.id,
            orderStatus: "in-progress",
            employeeID: employee.attributes.sub,
            _version: order._version
```

```
    }

    try {
        const update = API.graphql(graphqlOperation(updateOrder, {
            input: payload
        }))
        const updateResponse = await update

        let newOrders = [...orders]
        newOrders[index] = updateResponse.data.updateOrder
        setOrders(newOrders)
    } catch (err) {
        console.log(err)
    }

}

const completeOrder = async () => {
    const payload = {
        id: order.id,
        orderStatus: "complete",
        completed: true,
        employeeID: employee.attributes.sub,
        _version: order._version
    }

    try {
        const update = API.graphql({query: updateOrder, variables: {
            input: payload
        }});
        const updateResponse = await update
        let newOrders = [...orders]
        newOrders[index] = updateResponse.data.updateOrder
        setOrders(newOrders)
    } catch (err) {
        console.log(err)
    }

}

const dismissOrder = async () => {
    const payload = {
        id: order.id,
        _version: order._version

    }

    try {
        const update = API.graphql(graphqlOperation(deleteOrder, {
            input: payload
        }))
        const deleteResponse = await update
        let newOrders = [...orders]
        setOrders(newOrders.filter((item) => item!== order))
    } catch (err) {
        console.log(err)
    }

}
```

```
useEffect(() => {

    const gatherCustomer = async () => {
        try {
            const data = API.graphql( {query: getUser,
                variables: {
                    id: order.userID
                }
            })
            const response =  await data

            setcustomer(response.data.getUser)
            setIsLoading(false)

        } catch (err) {
            console.log(err)
        }
    }

    gatherCustomer()

    return () => {
        setcustomer();
     };


}, [])


const getTime = () => {
    const orderDate = new Date(order.createdAt);
    const time = orderDate.getHours() + ':' +  orderDate.getMinutes() + ':' + orderDate.getSeconds();
    return time
}

return (

    <View style={styles.orderContainer}>
        {!isLoading ? (
            <>
            <Card key={order.id}>
                <Card.Title title={customer.name.charAt(0).toUpperCase() + customer.name.slice(1) + "'s Order"} />
                <Caption style={styles.time} > Placed: {getTime()}</Caption>
                <Caption style={styles.time} > Code: {order.id.substring(0,5)}</Caption>
                {order.employeeID ? (
                    <>
                    <Caption style={styles.time} > Employee: {order.employeeID.substring(0,5)}</Caption>
                    </>
                ) : null}
                <Divider style={styles.divider} />

                <Card.Content>
                    {orderItems.map(item => {
                        return (
                          <Subheading key={item.id+Math.random()}>
                              {item.name}
                          </Subheading>
                        )
                    })}

                </Card.Content>
```

```
                <Card.Actions>
                  <Button disabled={order.orderStatus === "complete" || order.orderStatus === "in-progress"}
onPress={startOrder}>Start</Button>
                  <Button  disabled={!(order.orderStatus === "in-progress")} onPress={completeOrder}>Complete</Button>
                  <Button  disabled={!(order.orderStatus === "complete")} onPress={dismissOrder}>Dismiss</Button>
                </Card.Actions>

              </Card>
               <View style={styles.snackbar}>
                 <Snackbar
                   visible={visible}
                   onDismiss={onDismissSnackBar}
                   action={{
                   label: 'Undo',
                   onPress: () => {
                     // Do something
                   },
                   }}>
                   Click to Undo
                 </Snackbar>
               </View>

            </>

        ) : null}

      </View>
  )
}


export default Order
```

OrderItem.js

```
import React, { useState, useEffect } from "react";
import { API, graphqlOperation } from "aws-amplify";
import { onCreateOrder } from "../../graphql/subscriptions";
import { listOrders } from "../../graphql/queries";
import { Text, View } from "../../../components/Themed";
import Order from "./Order";
import { ScrollView, StyleSheet } from "react-native";
import { Auth } from "aws-amplify";
import AsyncStorage from "@react-native-async-storage/async-storage";

const styles = StyleSheet.create({
  scrollView: {
    width: '100%'
  },
  container: {
    flex: 1,
    alignItems: "center",
    justifyContent: "center",
    display: "flex",
    width: "100%",
    height: "auto",
  },
  Innercontainer: {
    flex: 1,
    justifyContent: "center",
```

```
    display: "flex",
    flexDirection: "row",
    width: "100%",
  },
  orderQueueContainer: {
    flex: 1,
    alignItems: "center",
   // alignContent:"center",
    justifyContent: "flex-start",
    display: "flex",
    flexDirection: 'column',
    flexWrap: "wrap",
    width: "30%",
   // maxHeight:600
  },
  title: {
    fontSize: 30,
    fontWeight: "bold",
    margin: 20,
  },
  subtitle: {
 //   alignItems: 'center',
    fontSize: 20,
    fontWeight: "bold",
    margin: 5,
  },
  separator: {
    marginVertical: 30,
    height: 1,
    width: "80%",
  },
});

const OrderQueue = () => {
  const [orders, setOrders] = useState([]);
  const [isLoading, setIsLoading] = useState(false);
  const [pressed, setPressed] = useState(false);
  const [employee, setEmployee] = useState();


  const getEmployee = async () => {
    const authEmployee = await Auth.currentUserInfo()
      .then((auth) => setEmployee(auth))
      .catch((err) => console.log(err));
  };

  const getBar = async () => {
    try {
      const jsonValue = await AsyncStorage.getItem("bar");

      if (jsonValue !== null) {
          return JSON.parse(jsonValue)
      } else {
          console.log("bar not found")
          return null
      }
    } catch (e) {
      // error reading value
    }
  };
```

```
useEffect(() => {
  getBar()
}, [])

useEffect(() => {
  let isMounted = true;
  getEmployee();


  const subscribe = async () => {
    const barr = await getBar()

    try {
      const ordersPromise = API.graphql(
        graphqlOperation(listOrders, {
          filter: { barID: { eq: barr.id } },
        })
      );

      const response = await ordersPromise;
      console.log(barr.id)
      setOrders(response.data.listOrders.items.filter((item) => item._deleted === null))
    } catch (err) {
      console.log(err);
    }

    try {
      const orderResponse = API.graphql(
        graphqlOperation(onCreateOrder)
      ).subscribe({
        next: (orderData) => {
          const item = orderData.value.data.onCreateOrder
          if(item.barID === barr.id){
            setOrders((orders) => [
              ...orders,
              item,
            ]);
          }

        },
      });
    } catch (err) {
      console.log(err);
    }
  };

  setIsLoading(false);
  subscribe();
  return () => {
    isMounted = false;
  };
}, [isLoading, pressed]);

return (
  <ScrollView style={styles.scrollView}>
    <View style={styles.container}>
    <Text style={styles.title}>Order Queue</Text>
    <View style={styles.Innercontainer}>
      <View style={styles.orderQueueContainer}>
        <Text style={styles.subtitle}>Received</Text>
        {!isLoading
```

```
    ? orders.map((order, index) => {
        if (order.orderStatus === "received") {
          return (
            <Order
              key={order.id + Math.random()}
              index={index}
              order={order}
              employee={employee}
              setPressed={setPressed}
              pressed={pressed}
              setOrders={setOrders}
              orders={orders}
            />
          );
        }
      })
    : null}
  </View>
  <View style={styles.orderQueueContainer}>
    <Text style={styles.subtitle}>In-Progress</Text>
    {!isLoading
      ? orders.map((order, index) => {
          if (order.orderStatus === "in-progress") {
            return (
              <Order
                key={order.id + Math.random()}
                order={order}
                index={index}
                employee={employee}
                setPressed={setPressed}
                pressed={pressed}
                setOrders={setOrders}
                orders={orders}
              />
            );
          }
        })
      : null}
  </View>
  <View style={styles.orderQueueContainer}>
    <Text style={styles.subtitle}>Completed</Text>
    {!isLoading
      ? orders.map((order, index) => {
          if (order.orderStatus === "complete")
            return (
              <Order
                key={order.id + Math.random()}
                order={order}
                index={index}
                employee={employee}
                setPressed={setPressed}
                pressed={pressed}
                setOrders={setOrders}
                orders={orders}
              />
            );
        })
      : null}
    </View>
  </View>
</View>
```

```
    </ScrollView>
  );
};

export default OrderQueue;
```

reducer.ts

```
import AsyncStorage from "@react-native-async-storage/async-storage";

export const initialState = {
  order: [],
  user: null,
  bar: null,
  completed: [],
  inprogress: [],
};


const storeBar = async (value : any) => {
  try {
    const jsonValue = JSON.stringify(value)
    await AsyncStorage.setItem("bar", jsonValue);
  } catch (e) {
    console.log(e);
  }
};

const getBar = async (state : any) => {
  try {
    const jsonValue = await AsyncStorage.getItem("bar");

    if (jsonValue !== null) {
      return {
        ...state,
        bar: JSON.parse(jsonValue),
      };
    } else {
        console.log("bar not found")
        return null
    }
  } catch (e) {
    // error reading value
  }
};

const reducer = (state: any, action: any) => {
  switch (
    action.type //mutable updates
  ) {
    case "SET_BAR":
      console.log(action);
      storeBar(action.bar)
      return {
        ...state,
        bar: action.bar,
      };
```

```
    case "ADD_TO_IN_PROGRESS":
      //Logic for order
      return {
        ...state,
        order: [...state.order, action.item],
      };

    case "ADD_TO_COMPLETED":
      //Logic for order
      return {
        ...state,
        order: [...state.order, action.item],
      };
    case "LOAD_BAR":
      return getBar(state);
    default:
      return state;
  }
};

export default reducer;
```