# PROJECT CASE STUDIES

# ORDER AHEAD

**Background**

Apps like GrubHub, Seamless, Caviar, and Postmates have shown the capability of an app to manage choosing and ordering food online.

**Problem**

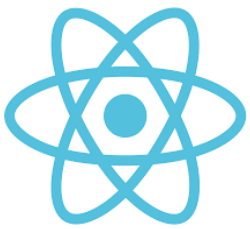
Travelers at the airport often find themselves hungry and late to catch a flight—without time to sit down and eat a good meal at a restaurant. So they just board their flight and are stuck with only the limited options offered by most domestic airlines. These passengers are excellent candidates to be converted into restaurant patrons.

**Solution**

Order Ahead is a React.js Web app for ordering food from OTG airport restaurants with a scheduled pickup time. It allows you to grab your OTG to-go food at your gate right before you board your flight.

The app dynamically integrates with all OTG restaurants around the country and their menus, handling credit card payments and sends users' food orders to the OTG Kitchen app, communicating with internal food making and legacy reporting systems

- Client (React.js/Firebase Hosting)
- Admin Panel Client (React.js/Firebase Hosting)
- Server (Node.js/Express/Heroku)
- Database (Firebase Realtime DB)
- Proxy Server (Node.js/Heroku)
- Integrated with First Data for payment
- State management with Redux, Persist, Thunk, and Reselect







# CART-SCREEN REPLACEMENT MICRO FRONT-END

**Background**

OTG's flagship iOS app is called **Concierge.** It handles the menus, payment, and ordering for every tablet in every OTG restaurant across the country. Running on more than 20,000 tablets, OTG is second only to Apple themselves in the number of client-facing tablets in use.

**Problem**

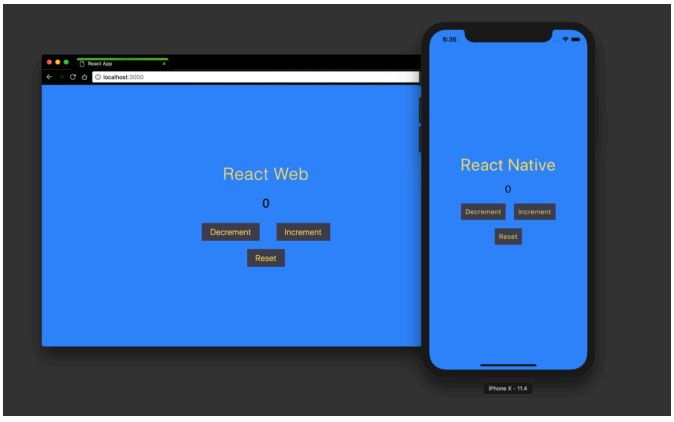
App deploys are currently done using an automated system, but internet quality is unpredictable in airports and the quality of signal is most often completely out of our control. Many tablets need to have their software manually reinstalled on our monthly release schedule; this is a heavy LOE for the team and takes all night. It also must be coordinated across multiple airports and terminals, which further complicates these releases and makes it very difficult to release a patch.

OTG made deals with several airlines to have their custom design, branding, and dynamic content applied to the app's final cart screen and they wanted the ability to make changes on a much faster release cycle.

**Solution**

A new cart screen design was approved and I implemented it using **React Native Web**[1] (RNW), which we loaded into the iOS app as a webview. This opened up an interesting alternative to brownfielding a React Native app into an existing iOS app, allowing incremental progress through micro front-ends. This helped build a bridge to converting the rest of the iOS app to React Native in stages instead of converting the entire app at once, which would be very time intensive.

- Client (React Native Web)
- Redux, Thunk, Reselect
- Storybook




[1] Enables a mobile app built with React Native share code and be translated into Rect.js for the web.

# IN-APP AD DISPLAY GENERATOR MICRO FRONT-END

**Background**

We display dynamic promotions throughout our main ordering app Conceirge, such as incentivicing users to sign up for an airline rewards credit card to unlock a discount on the cart screen or as simple as advertising a happy hour drink special.

**Problem**

These promotions were hard-coded in the app, which made them very inflexible from a design and implementation standpoint. Translation: not easy to customize or update on the fly.

**Solution**

The team built an admin panel for creating and designing flexible promotions. The admin panel would then produce a long query string that concierge would use to load an iOS web-view to display the promotion. A React.js website then uses the properties and attributes in the query string to build a variety of different promotion types out of templates. This makes the promotions easy to add on the fly and update without a formal app release.

- Client (React.js)

**Co-Developer, In-App Ad Display Generator micro front-end for iOS app Concierge**
An templated ad building system for non-technical staff

• Designed templates that dynamically generate based on selected admin panel properties
• Refactored to align with DRY, clean code, code formatting tools, and other Javascript best practices