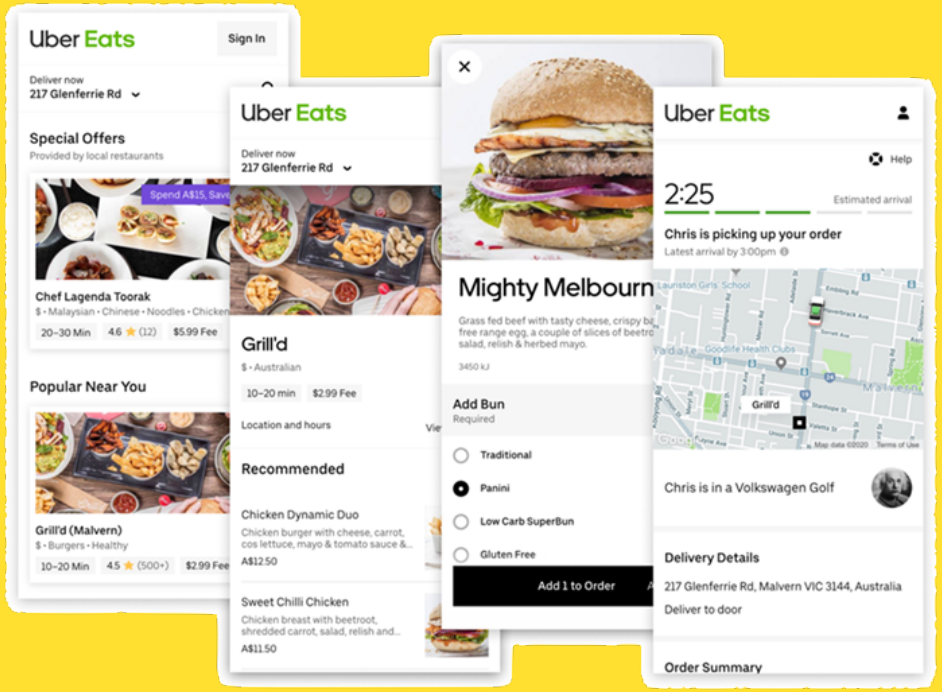
notJust

//development



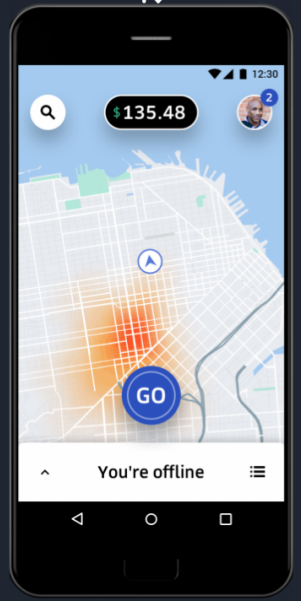
Uber Eats

5 Days Challenge

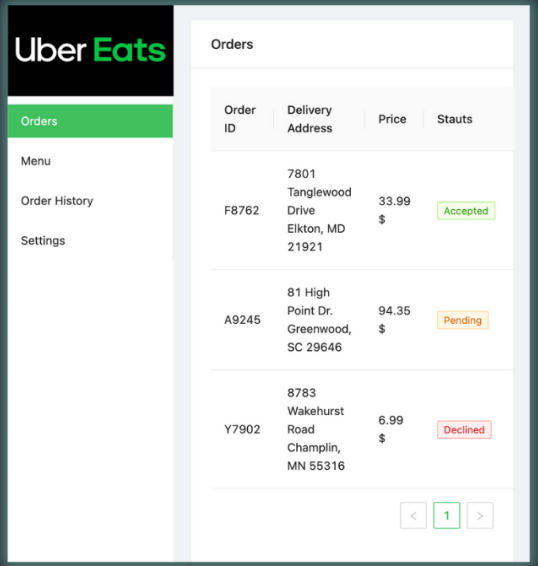
Day 2: Backend with AWS Amplify

AWS Amplify

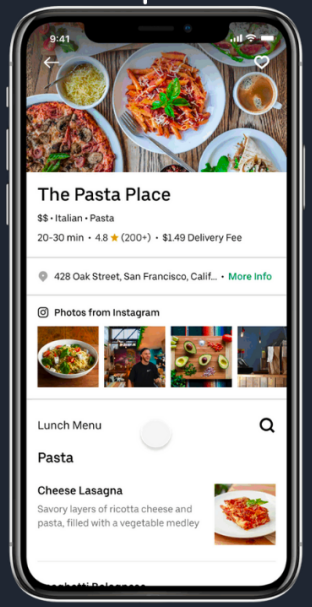
Backend



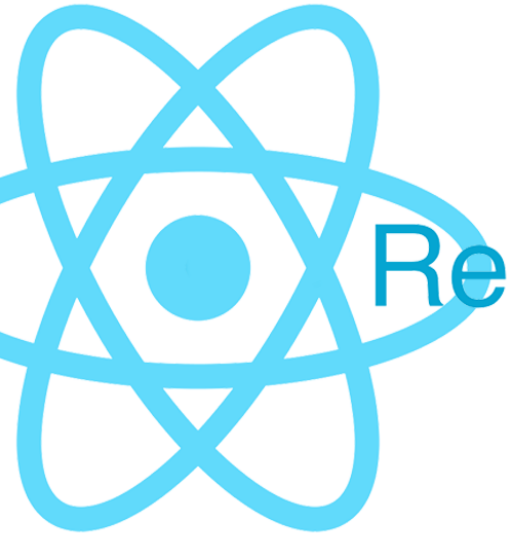
Driver app



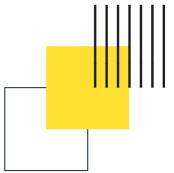
Restaurant dashboard

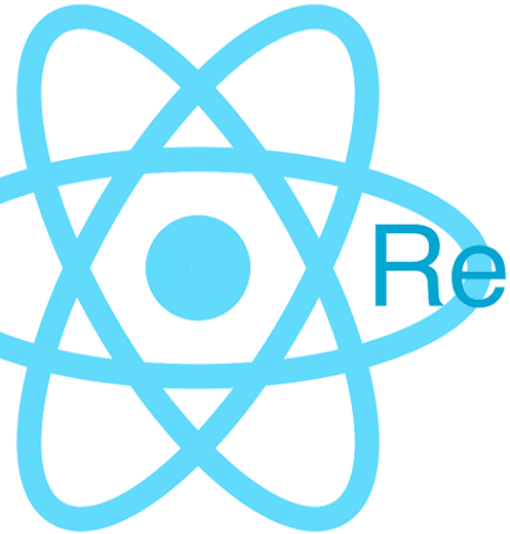


User app



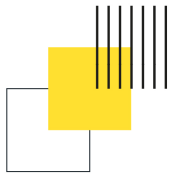
React Native





React Native

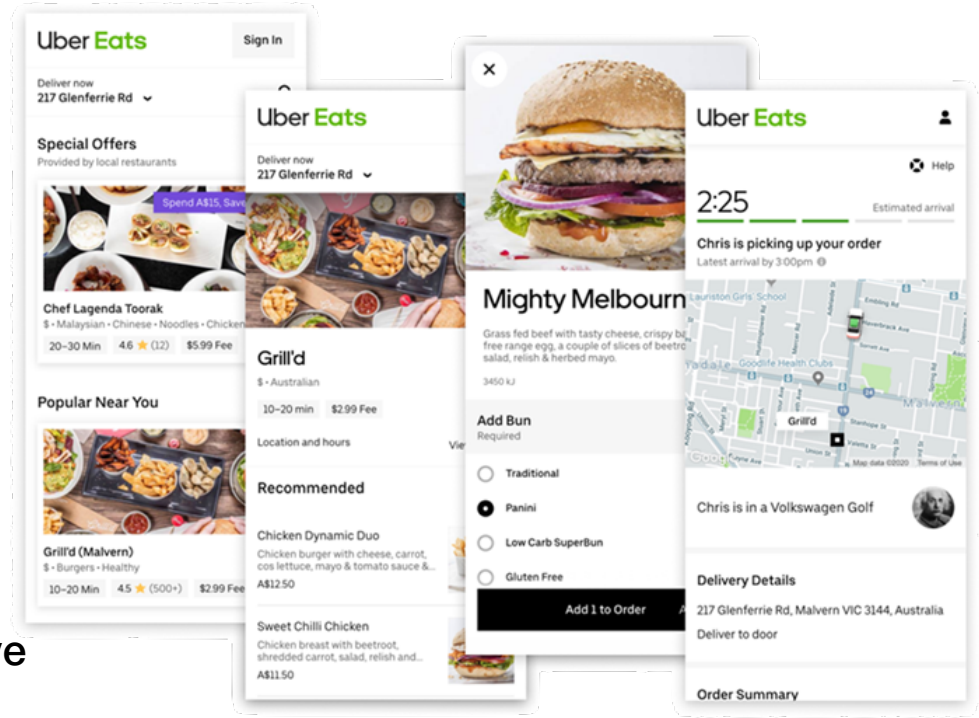
AWS Amplify



notJust

//development

The plan

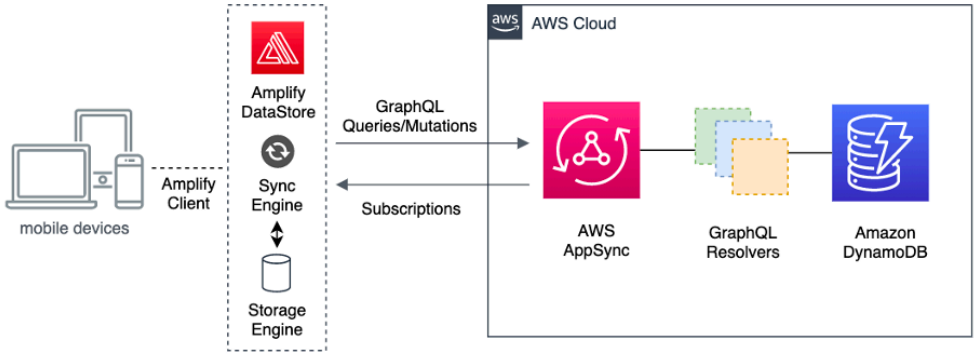


- Day 1: User app UI | React Native

notJust
//development

AWS Amplify
Backend

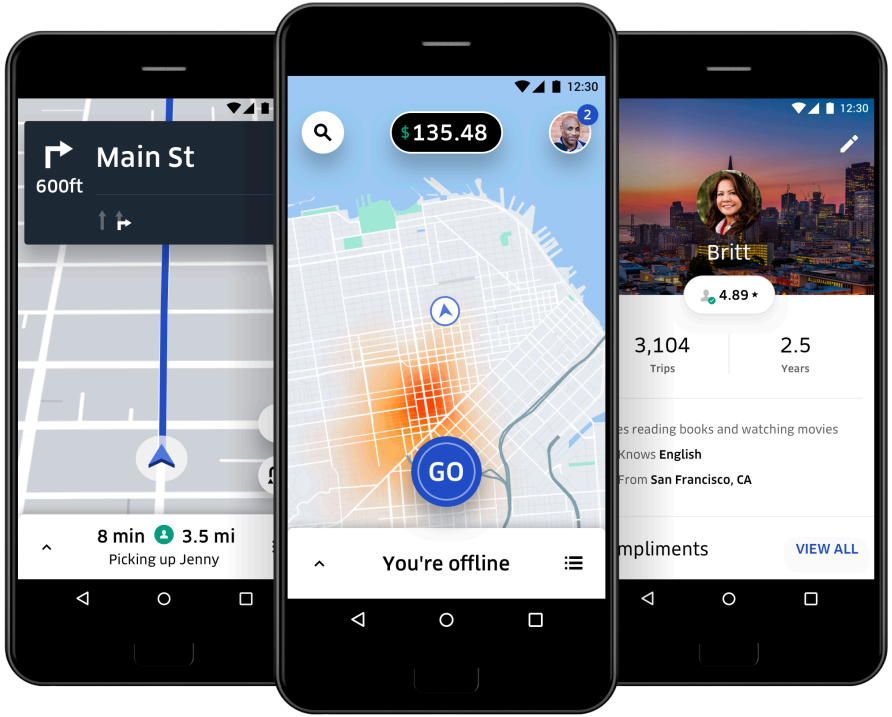
The plan



- Day 1: User app **UI** | React Native
- **Day 2: User app backend | AWS Amplify**

notJust
//development

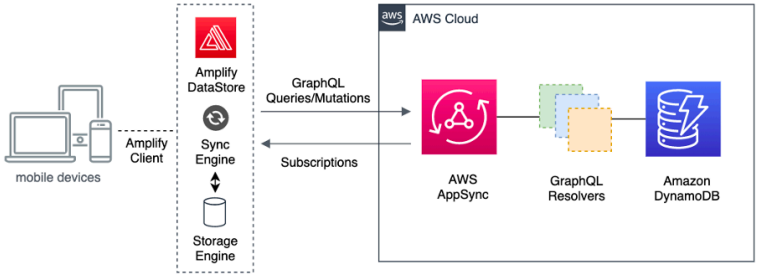
The plan



- Day 1: User app **UI** | React Native
- Day 2: User app **backend** | AWS Amplify
- **Day 3: Driver app UI | React Native**

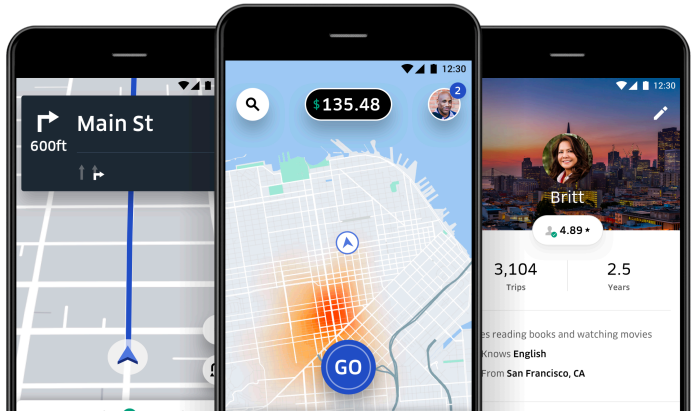
notJust
//development

AWS Amplify
Backend



The plan

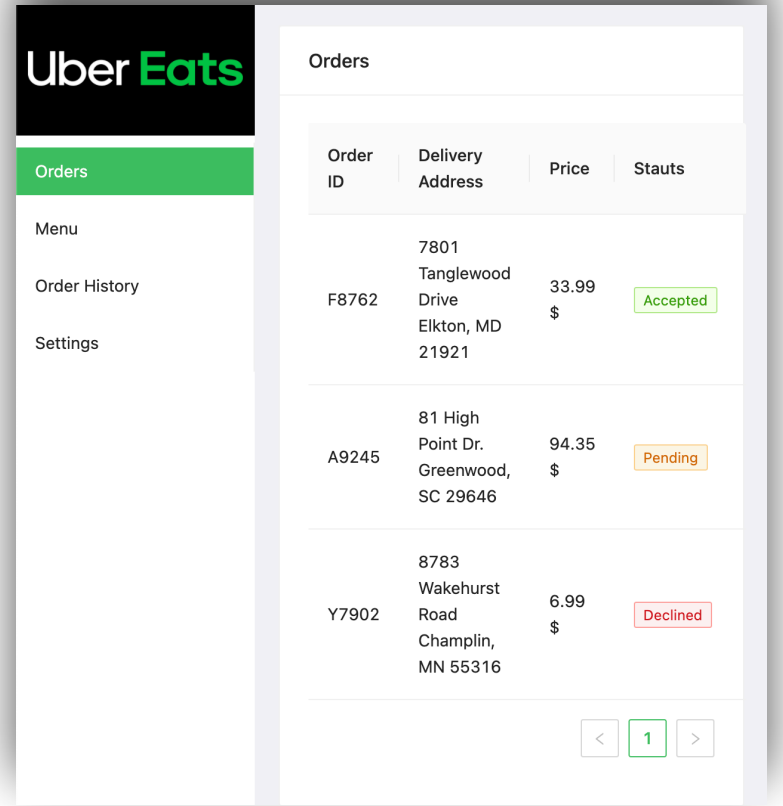
- Day 1: User app **UI** | React Native
- Day 2: User app **backend** | AWS Amplify
- Day 3: Driver app **UI** | React Native
- **Day 4: Driver app backend | AWS Amplify**



notJust
//development

The plan

- Day 1: User app **UI** | React Native
- Day 2: User app **backend** | AWS Amplify
- Day 3: Driver app **UI** | React Native
- Day 4: Driver app **backend** | AWS Amplify
- **Day 5: Restaurant dashboard | ReactJS and AWS Amplify**



notJust
//development

Prerequisites

1. Asset Bundle (dummy data, PDF presentation):
   https://assets.notjust.dev/ubereats

2. Expo environment setup
   http://bit.ly/expo-vadim

notJust

//development

Let’s get started

Day 2

notJust
//development

Who am I

- Full stack developer for >7yrs
- Founded multiple startups
- ex Amazon SDE
- Passionate about coding and building impactful startups.

I AM notJust
//developer

notJust
//development

0. Prerequisites

Before we begin, make sure you have the following installed:

- NodeJS and npm
- git
- AWS Account
- Amplify CLI

Docs: https://docs.amplify.aws/start/getting-started/installation/q/integration/react-native/

notJust
//development

1. Create the app backend

1. Go to aws console: https://aws.amazon.com/
2. Create an account if you don’t have one and sign in
3. Go to Amplify dashboard, by searching Amplify
4. Press “New app” and choose “Build an app”
5. Wait a bit until AWS sets up the project
6. Press “Launch Studio” to open in in Amplify Studio

notJust
//development

2. Amplify Studio overview

1. Set up is where we will setup our backend services like Auth, Database, etc.
2. Manage is where we will mange the content and the users of our application

notJust
//development

3. Setup authentication

notJust

//development

4. Install the dependencies in our project

```
npm install aws-amplify aws-amplify-react-native @react-native-community/netinfo @react-native-async-storage/async-storage @react-native-picker/picker
```

notJust
//development

5. Connect our React Native app with the Amplify backend

1. Grab the pull command from Amplify Studio (from the top right corner)
2. Execute the command in the root directory of your project
3. Configure Amplify

```
import { Amplify } from 'aws-amplify'
import awsconfig from './src/aws-exports'
Amplify.configure(awsconfig)
```

notJust

//development

6. Implement Authentication

```
import { withAuthenticator } from "aws-amplify-react-native";

export default withAuthenticator(App);
```

notJust
//development

Data modeling

- Restaurant
- Dish

The relationship between the Restaurant and dishes is 1 to many, because one Restaurant can have many dishes, and one dish belongs to one Restaurant only

notJust

//development

8. Setup data

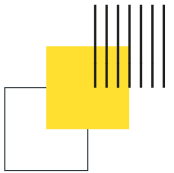
1. Deploy changes
2. Pull backend in the app
3. Populate database by seeding
    1. lat 46-47
    2. lng 28-29
4. Update images for restaurants
5. Populate manually the dishes

9. Fetch restaurants

```
import { DataStore } from "aws-amplify";
import { Restaurant } from "../../src/models";

const Restaurants = () => {
  const [restaurants, setRestaurant] = useState<any>([]);

  const fetchRestaurants = async () => {
    const response = await DataStore.query(Restaurant);
    setRestaurant(response);
  };

  useEffect(() => {
    fetchRestaurants();
  }, []);

  ...
}
```

10 Fetch restaurant details

```
const [restaurant, setRestaurant] = useState();
const [dishes, setDishes] = useState([]);

const navigation = useNavigation();
const route = useRoute();
const id = route.params?.id;

const fetchRestaurant = async () => {
  if (id) {
    const response = await DataStore.query(Restaurant, id);
    setRestaurant(response);

    const dishResponse = await DataStore.query(Dish, (dish) =>
      dish.restaurantID("eq", id)
    );
    setDishes(dishResponse);
  }
};

useEffect(() => {
  fetchRestaurant();
}, [id]);
```

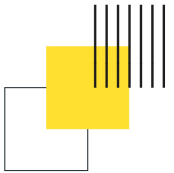


11 Fetch Dish details

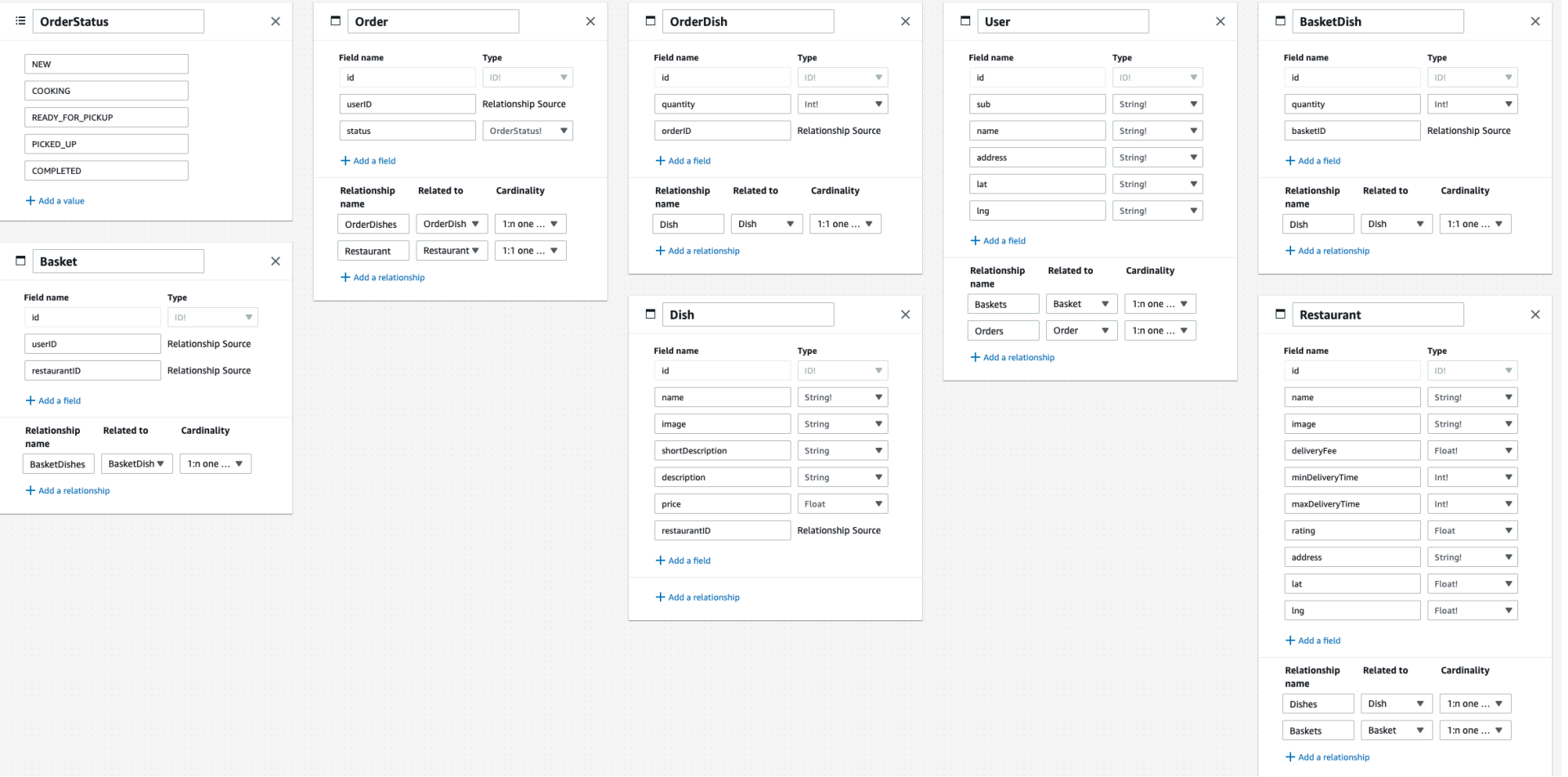
JavaScript

Copy Caption

```
useEffect(() => {
    if (id) {
      DataStore.query(Dish, id).then(setDish);
    }
  }, [id]);
```



Complete Data Model



Join the waitlist Log In

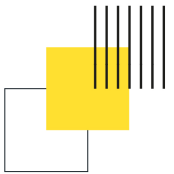
Are you ready to become a Full-Stack Mobile Developer that can build end-to-end apps in weeks?

Join the **Full Stack Mobile Developer** course and learn the full lifecycle of developing mobile applications. Master both **frontend** and **backend skills** by building complex real-world projects

Don't miss it! Join the waitlist

Course content

2	105 +	30+	77 k +	200 +
Real-world projects	video lessons	hours of video	lines of code	developers in the community



2. Project Setup

Set up a new project using **React Native CLI** and install the most important tools and libraries that we will need.

You will also learn how to version your project with **Git** and manage it on **GitHub**

3. Instagram UI

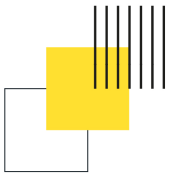
Build all the essentials screens of Instagram: Feed, Post, Comments, Profile, Edit Profile, Camera, etc.

- Building and Styling the UI
- State and Props
- Custom components
- **Image**, **Video,** and **Carousel** posts
- Camera and Media Library API
- **Form management** with React Hook Form
- Typescript

4. Navigation

Design and implement the Navigation system using **React Navigation v6**

- Stack Navigator
- Bottom Tab Navigator
- Top Tab Navigator
- Moving between screens and passing data
- Deep Linking



5. Getting Started with AWS Amplify

Get a high overview of how AWS Amplify works and how to leverage the power of AWS to build mobile apps *quickly*.

- Create and set up your AWS account
- Implement important **security** measurements
- Create the backend using **Amplify Studio** and connect it with our React Native app.

6. Authentication

Setup the Authentication layer and learn how everything works behind the scenes in AWS Cognito.

Also, you will learn how to fully customize the authentication flow with themes and custom screens.

And of course, you will also integrate **Sign in with Facebook** and **Google.**

7. API

You will learn how to **design** and **implement** a **GraphQL API** using Amplify and AWS AppSync.

On the client-side, you will learn how to use **Apollo Client** to query and cache our GraphQL API.

We will implement all the **CRUD** functionalities for users, posts, likes, comments, etc.

In the end, we will cover more advanced topics like:

- Pagination
- Sorting
- Real-time data
- Security

8. Storage

Learn about cloud storage with AWS S3 by implementing the storage layer in our application.

Upload, Download and Delete:

- Images and Carousels
- Videos
- Profile picture

9. Production

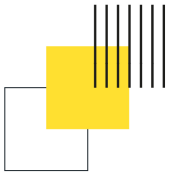
Prepare and deploy your application to production and distribute it on Play Market and AppStore.

We will start by preparing all the marketing assets needed for the stores. You will learn about the requirements and also about the **best practices** to follow to create a compelling and efficient page for your app.

We will proceed with creating your Dev accounts and setting up the app page on **Play Market** and **AppStore**.

Then we will implement a **multi-environment** system for both the source code and AWS infrastructure.

We will build and upload our **Android** and **iOS** to both stores using **Expo Application Services**.



10. Following System

Implement the **Following & Feed System.**

Follow people and track the followers and followings. The feed will display only posts from people you follow

Coming soon!

11. Notification System

Implement event-based **in-app** and **push notifications.**

- New followers notification
- Likes and comments notifications
- Push Notifications

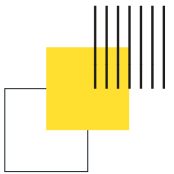
Coming soon!

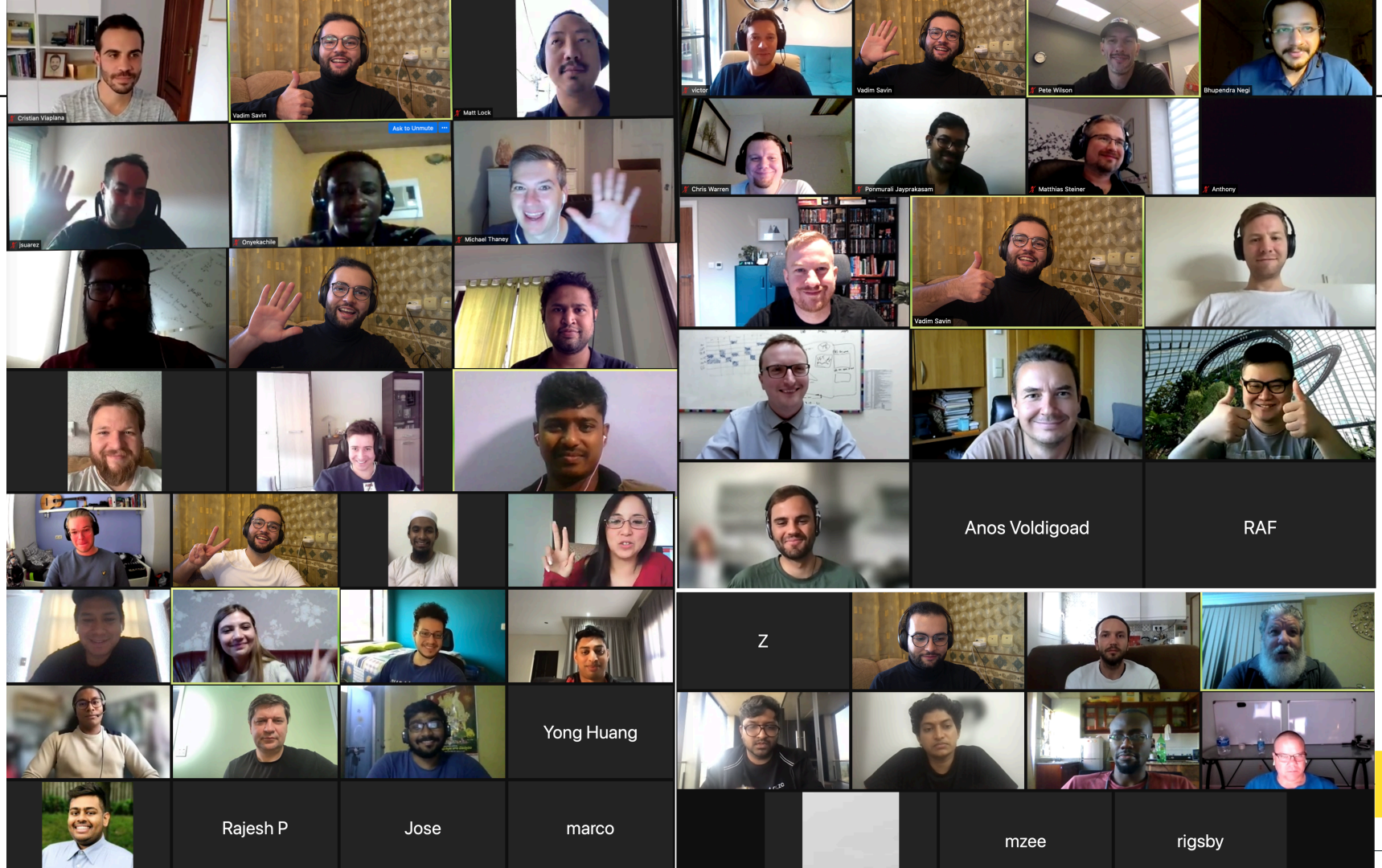
12. Testing

Practice Test Drive Development and write unit testing for your React Native application.

Have peace of mind and make sure that your future updates will not brake your application.

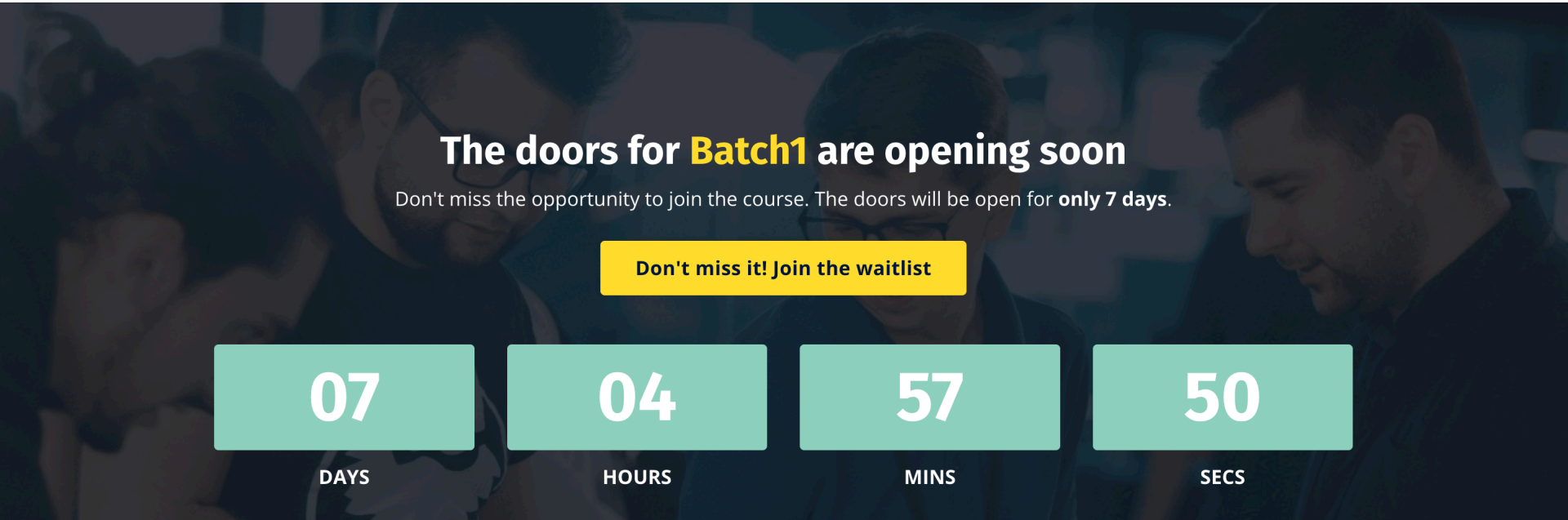
Coming soon!



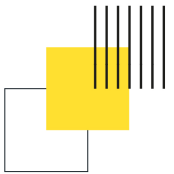

Cristian Viaplana
Vadim Savin
Matt Lock
victor
Vadim Savin
Pete Wilson
Bhupendra Negi
Ask to Unmute
Chris Warren
Ponmurali Jayprakasam
Matthias Steiner
Anthony
jsuarez
Onyekachile
Michael Thaney
Vadim Savin
Anos Voldigoad
RAF
Z
Yong Huang
Rajesh P
Jose
marco
mzee
rigsby

Launching next Monday, 25th April

🕗 8 am PT 🕚 11 am ET 🕟 4 pm WEST 🕣 8:30 pm IST



https://academy.notjust.dev/



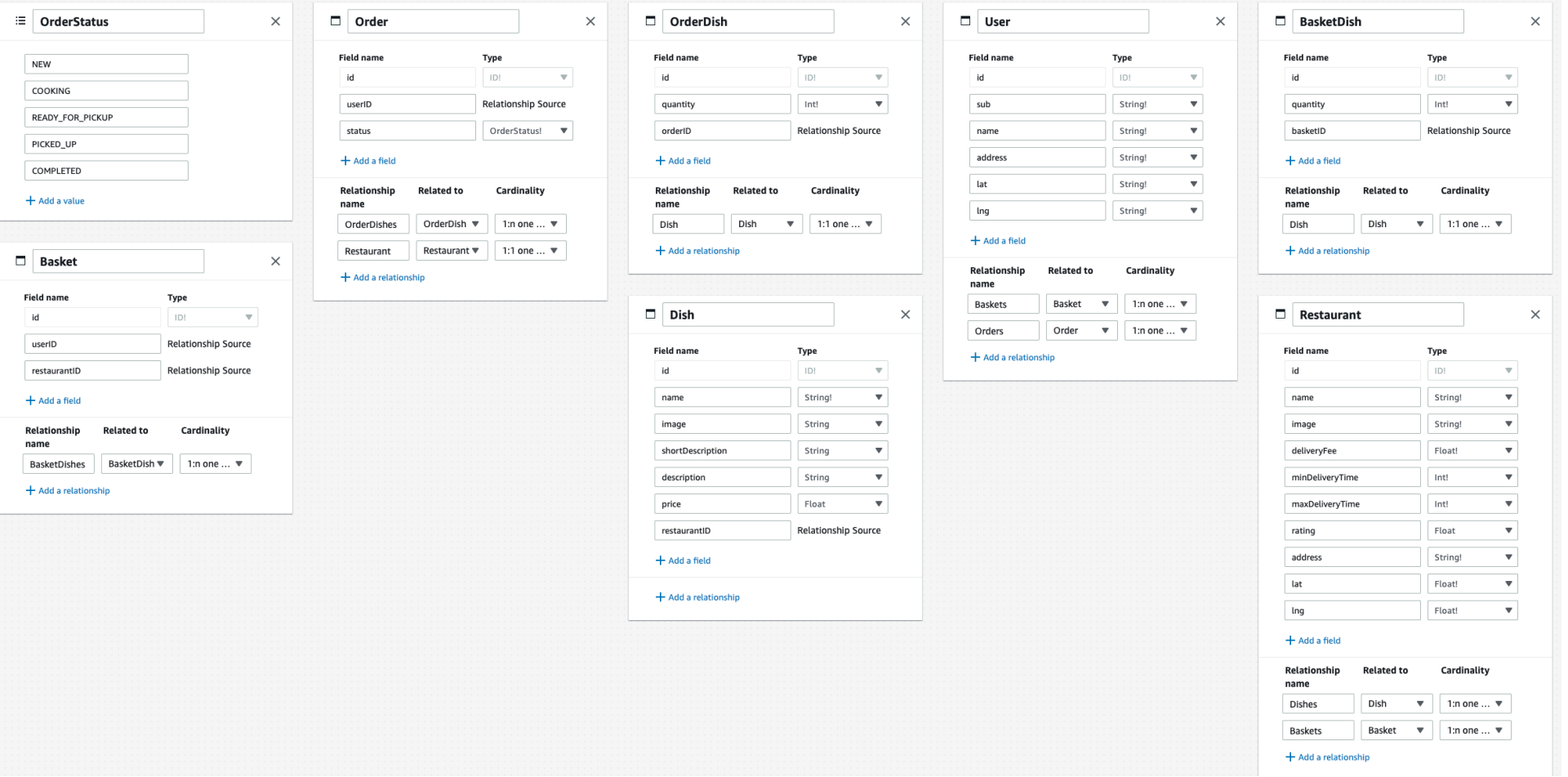
notJust

//development

Data modeling

- Restaurant
- Dish
- User
- Basket
- BasketDish
- Order
- OrderDish

Complete Data Model



notJust

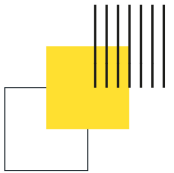
//development

User CRUD

User CRUD

- AuthContext
  - create auth context
  - save in state authUser using Auth module, and dbUser by querying Datastore
  - export the provider and a custom useAuthContext hook
  - Show the profile form if the dbUser is null
  - Save a new user

```
const createUser = async () => {
    const response = await DataStore.save(
      new User({
        sub,
        name,
        address,
        lat,
        lng,
      })
    );
    setDbUser(response);
  };
```

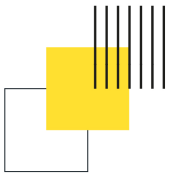


Update User

- first, populate fields with db user
- save user

JavaScript Copy Caption

```javascript
const saveUser = async () => {
    const response = await DataStore.save(
      User.copyOf(dbUser, (updated) => {
        updated.name = name;
        updated.address = address;
        updated.lat = lat;
        updated.lng = lng;
      })
    );
    setDbUser(response);
  };
```



notJust

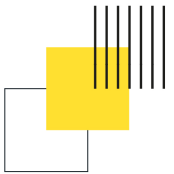
//development

Basket

Basket Context

- Create the Basket context, export the provider and the useBasket hook
- Wrap navigation in BasketContextProvider
- In BasketItemScreen:

```
const addToBasket = async () => {
    await addDishToBasket(dish, quantitiy);
    navigation.goBack();
  };
```



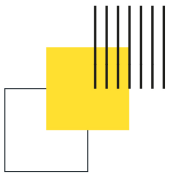
Add to basket functionality

- Fetch the basket when the restaurant or dbUser changes

```
const fetchBasket = async () => {
    const response = await DataStore.query(Basket, (b) =>
      b.userID("eq", dbUser.id).restaurantID("eq", restaurant.id)
    );
    setBasket(response[0]);
  };
```

- Add dish to basket

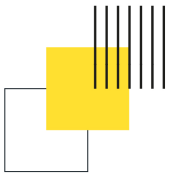
```
  const createBasket = async () => {
      console.log("Creating Basket");
      const response = await DataStore.save(
        new Basket({ userID: dbUser.id, restaurantID: restaurant.id })
      );
      setBasket(response);
      return response;
    };

  const addDishToBasket = async (dish, quantity) => {
      const newBasket = basket || (await createBasket());

      const response = await DataStore.save(
        new BasketDish({
          basketID: newBasket.id,
          Dish: dish,
          quantity,
        })
      );
      fetchBasketDishes();

      return response;
    };
```



Display View Basket Button

- only if basket is created
- fetch all BasketDishes and display the length in the button

```
const fetchBasketDishes = async () => {
    if (basket) {
      const response = await DataStore.query(BasketDish, (b) =>
        b.basketID("eq", basket.id)
      );
      setBasketDishes(response);
    }
  };
```

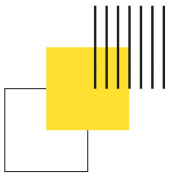


notJust
//development

Orders

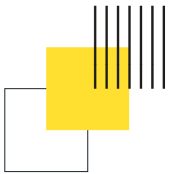
Order Context

- Create the Order context, export the provider and the useOrders hook
- Wrap navigation in OrderContextProvider



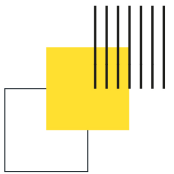
Create order

```
// Create Order
    const newOrder = await DataStore.save(
      new Order({
        userID: dbUser.id,
        status: "NEW",
        Restaurant: restaurant,
      })
    );
```



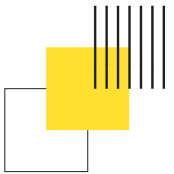
Create OrderDishes based on BasketDishes

```
// Create OrderDishes based on basketDishes
    await Promise.all(
      basketDishes.map((basketDish: BasketDish) =>
        DataStore.save(
          new OrderDish({
            Dish: basketDish.Dish,
            quantity: basketDish.quantity,
            orderID: newOrder.id,
          })
        )
      )
    );
```



Delete basket

```
await DataStore.delete(basket);
```



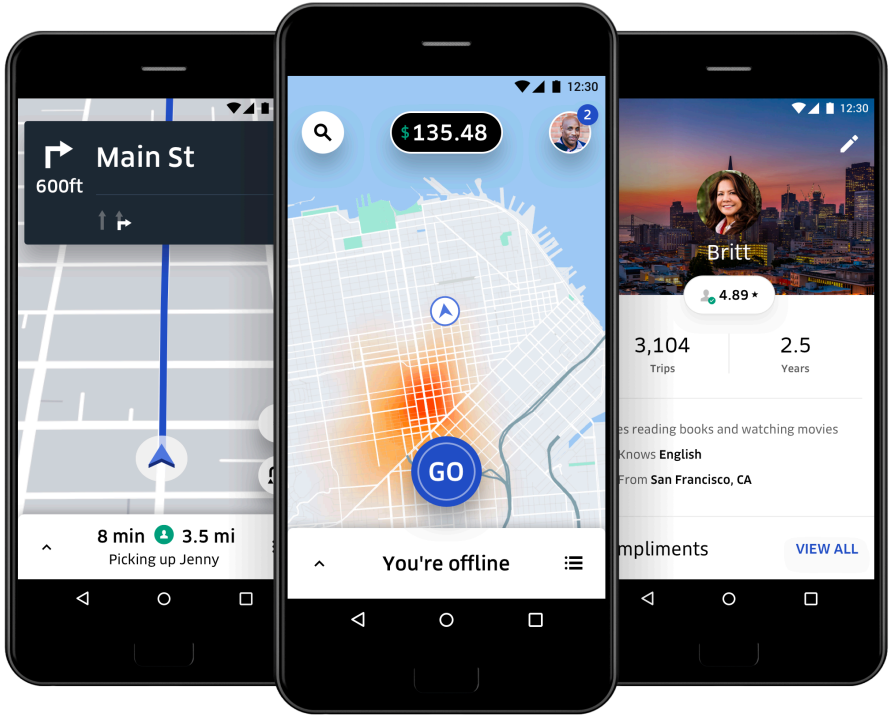
notJust

//development

Demo

notJust
//development

Join me tomorrow



- Day 1: User app **UI** | React Native
- Day 2: User app **backend** | AWS Amplify
- **Day 3: Driver app UI | React Native**