Master A Million for iOS and Android

React Native (Dec 2017)

Integrating Custom Hardware with React Native

• Client provided test iOS and Android apps that decoded the MasterAMillion ball when connected to the headphone port.

• I had to integrate and bridge this native code into React Native for both iOS and Android.

• Clicking a button on the ball while it is plugged into the phone's headphone port triggers a series of audio tones which create a string of binary numbers that are decoded into the ball's bounce count and it's id.

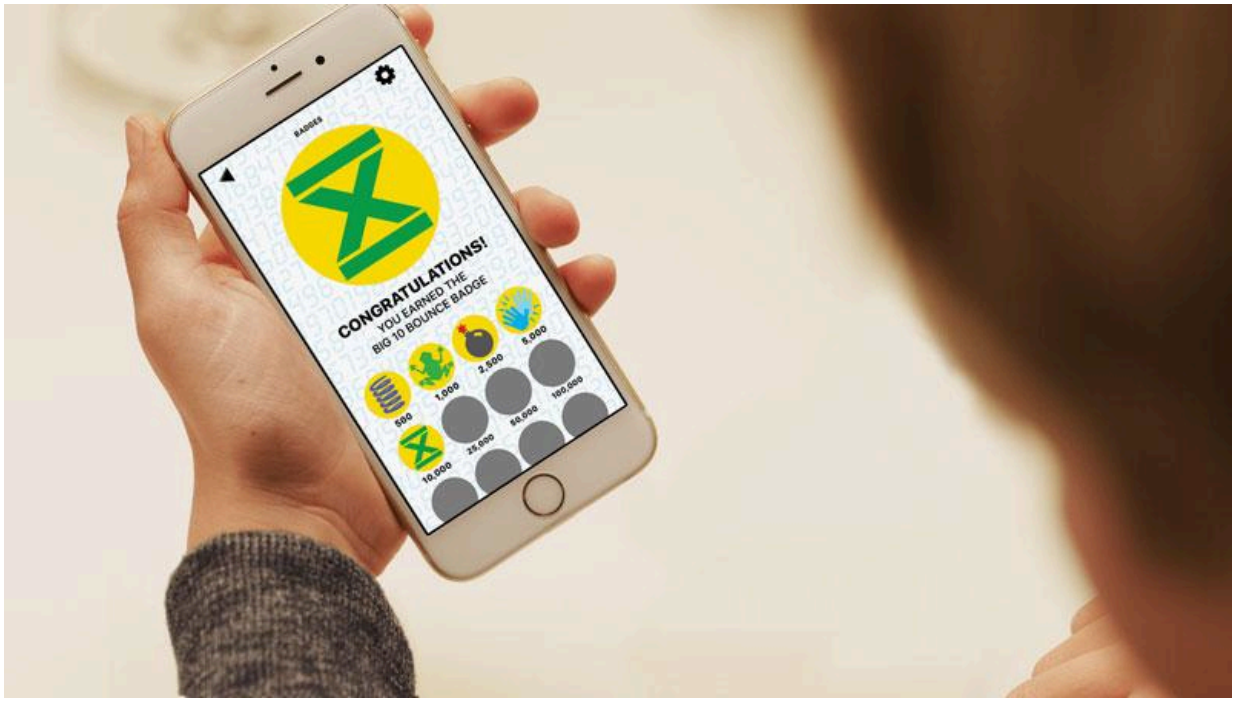
• The bridge incorporates this custom decoding library into the React Native app and sends the ball data from the native layer to React Native on ball button click.

Redux + Thunk + Persist

Using Redux and Thunk I decoupled the logic from my components and put them in Async actions instead.

This decoupling pattern made reordering components to fix rerendering issues incredibly easy, the code very reusable, and flexible if something needed to be added or rewritten.

Using Persist I was able to save state between sessions so users wouldn't always need to manually log back in to the app.



Building an app that looks great on all phones, tablets, and operating systems

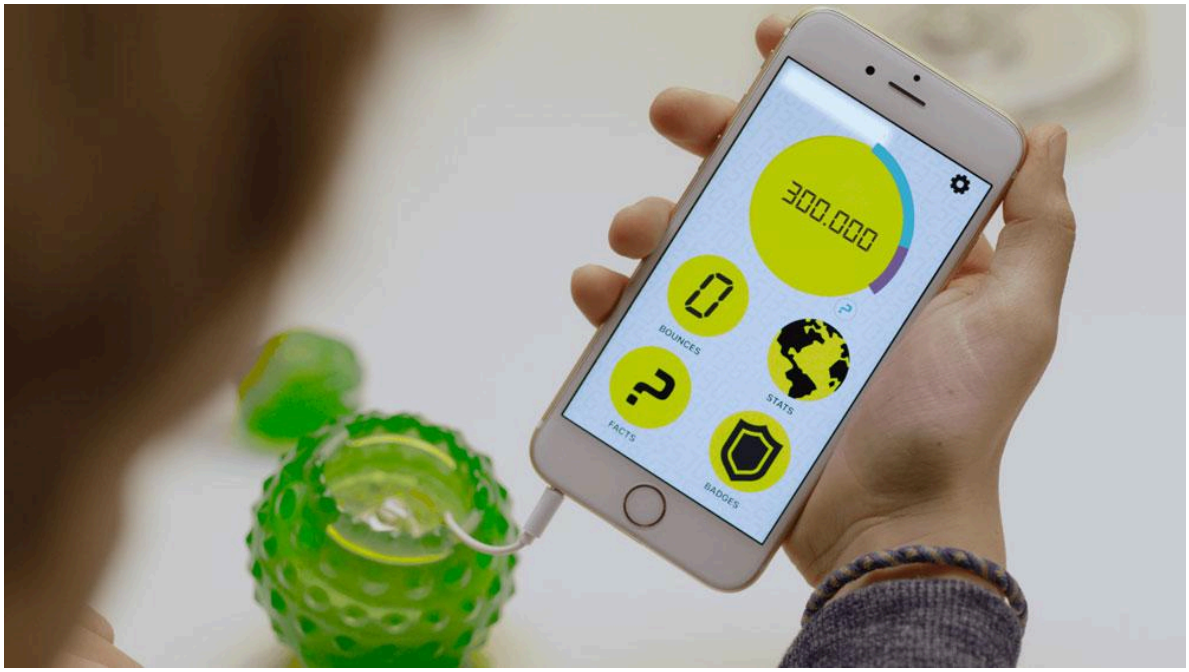
• In order to write styles that would look great on all devices I quickly realized I needed to set different values for phones and tablets.

• Later I realized I needed to be able to specify different values for phones and tablets by OS to truly cover everything with one code base.

• I developed and published an api of methods that can be used in a variety of different ways to make conditional styles based on phone, device type, or OS and published to NPM.

• Install by typing 'npm i --save react-native-cross-platform-responsive-dimensions'

Frontend UI and Animation



• **Animated** bouncing ball **splash screen**, bouncing spinning badges, and score count up effect

• **Touchable highlight** for native pressure sensitive button animations

• **Loading animations** at all wait points along with informative error messages

• Utilized **FlatList** for displaying an infinite list with many elements

• Positioned elements with **FlexBox** CSS

• Used **Higher Order components** to reuse design elements and check for Network Connectivity

• Configured **custom fonts** and implemented designer styled **react-native-router-flux** navigation

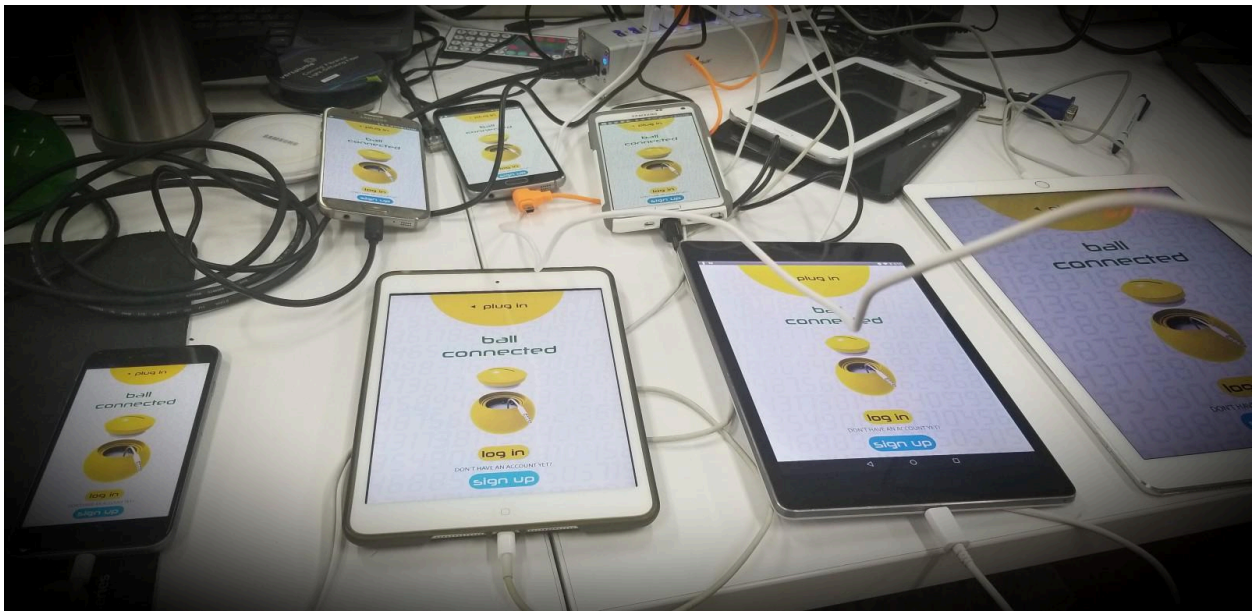
Testing/Debugging

• Used React, React Native, Redux - DevTools, VSCode debugger, Android Device Manager

• **Detox** e2e testing: [Video of End to End Test]

• Created **end to end tests** to automate testing user sign up, login, and edit user info screens

• Test on up to 10 physical devices simultaneously



Deployment

• Used TestFairy to send test builds to the client

• Configured Microsoft Code Push for one command deployment of live updates to app code

• Firebase hosting for React.js admin panel

Multi-language with react-native-i18n

• Swedish, Norwegian, English, Danish, Finnish

• Replaced all static text in the app with strings variable pointing to custom json translation file

• Created language based endpoints in firebase for dynamic editing the Faq in the Admin panel

Bug Tracking and Analytics

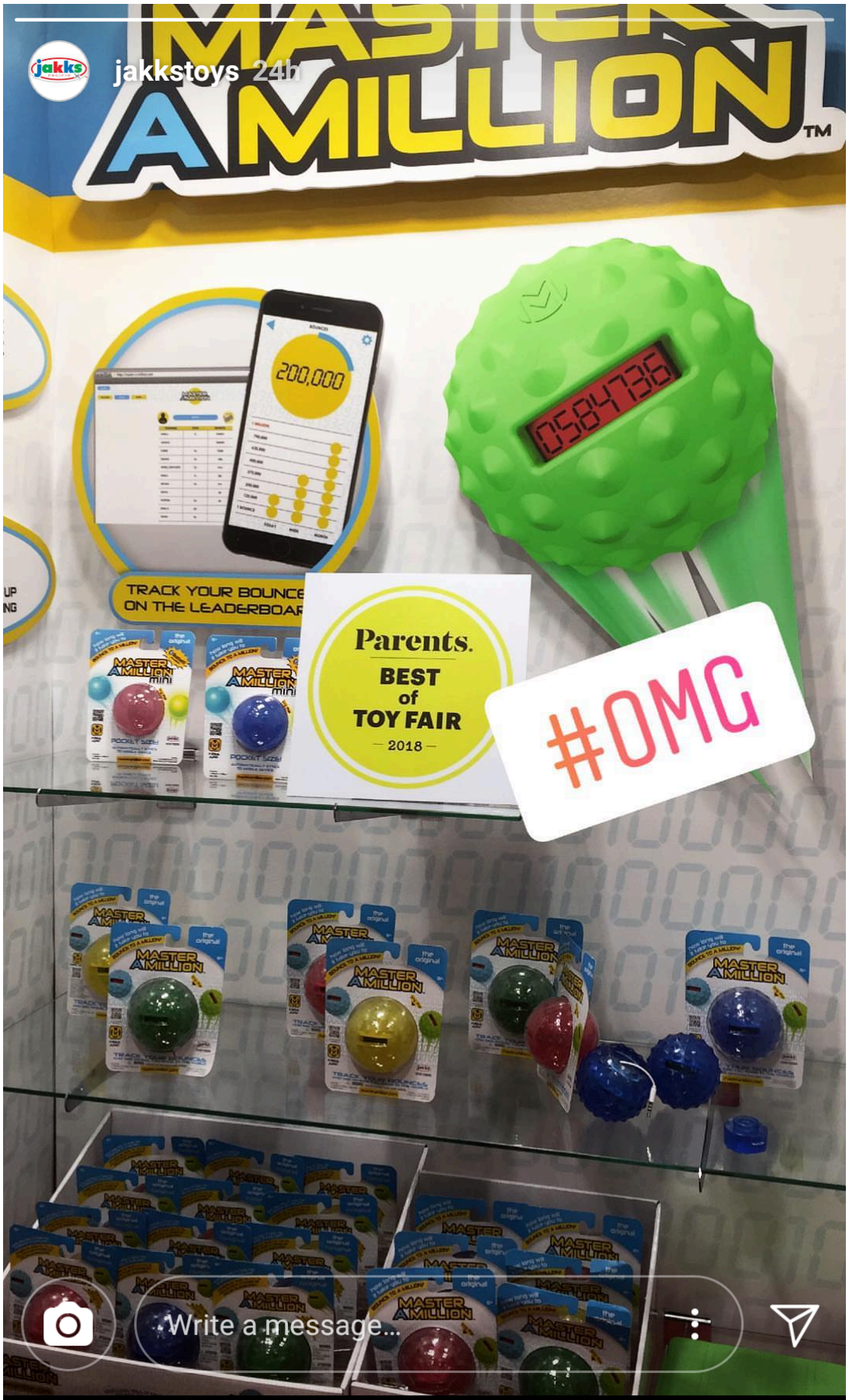
• Google Analytics - Added custom triggers for all screen navigation events, app interaction points, and all hardware sync events

• BugSnag - Added tracking of logs, user crash data by user, and breadcrumb events using the Google Analytics events

• Created custom tables and charts in React.js admin website displaying google analytics data

Backend

• Authentication with Firebase Auth + Firebase Realtime DB


jakkstoys 24h
MASTER A MILLION™
200,000
0584736
TRACK YOUR BOUNCE
ON THE LEADERBOAR
Parents.
BEST
of
TOY FAIR
2018
#OMG
Write a message...