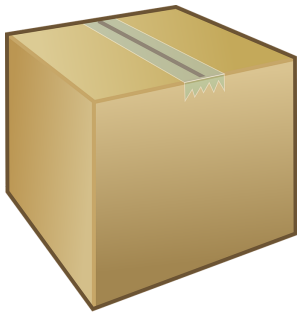


BOXIE

The home storage solution you didn’t know you needed.

The Box Club:
Alan Bird, Kevin Law, Tyler Grant, Eric Sims
September 2022

Table of Contents

1 Executive Summary **3**

2 Background and Technical Requirements **4**

2.1 General Summary 4

2.2 Other Projects 4

2.3 Idea Space 4

2.4 Technologies Leveraging 4

2.5 Software/Hardware Requirements 5

3 Requirements Analysis **5**

3.1 System Architecture 5

3.2 Personnel 7

3.3 System Features 7

Rank 1: Bare Essentials 7

Rank 2: Planned Features 7

Rank 3: Bells and Whistles 8

4 Software Engineering Tools and Techniques **8**

5 Timeline **9**

6 Appendix A: UI Sketches **10**

7 Appendix B: Use Cases

8 Changelog **12**

1 Executive Summary

Imagine that you and your family are about to embark on a fun camping trip. The car is all loaded up and you are about ready to head out, but you realize that a key item is missing - tent stakes! In the rush to leave, you run to the basement thinking you have a general idea of where they are at. Once there, you realize that all of the boxes have been re-organized since the Christmas decorations have been put back. It takes you a half an hour to open a number of boxes before finding camping related items. Then it takes more time to rummage through the camping boxes. Alas, you find the stakes and hurry to leave the room before looking back at the mess you made.

Wouldn't it be nice if you could hire a professional organizer to keep everything in a place where they should be? Wouldn't life be so much easier if we lived as wizards and could use the *ACCIO* (summon) spell to retrieve things? Unfortunately, for most people, those options are not available. But the great news is that we have developed a solution to finding things in long-term storage easily and quickly.

Boxie is an application that helps you keep track of where things are stored. With the use of text search or QR code scanning, we aim to help you find anything in less than a minute. You can organize your long-term storage in any way you like, and Boxie will track it. Adding an item to a box or a box to a room is as easy as adding a friend on Facebook. You can store as little or as much information about each item as you need. Do you have a collection of model trains? Then record the purchase receipts and their serial numbers for safekeeping!

With this powerful information in the palm of your hand, Boxie is able to do something incredible. Accidents happen, and sometimes there is a fire in the attic or the basement floods. Or sometimes a robber breaks in and steals your entire model train collection. Insurance companies can typically cover for the items lost, but you'll get a low-ball value estimate if you don't have a thorough inventory of your home. With Boxie, all of that information is available to you with the click of a single button!

Boxie makes managing long-term storage a low-cost and easy-to-learn solution. When you need it, you can rely on it for finding your items quickly and being able to report lost or damaged items in seconds.

2 Background and Technical Requirements

2.1 General Summary

Boxie is meant to enhance the lives of homeowners by reducing the time it takes to find items in long-term storage and by keeping a helpful inventory of what they own. This will be a cross-platform application that can keep the information in the cloud. Therefore, the information that users have gathered will be safe, even if they lose their phone or if it gets destroyed. We believe that we can achieve a better result than our competitors and create an app that people will want to use.

2.2 Other Projects

Elephant-Trax: This application only allows users to inventory items with pictures. There is no text description that you can add. Therefore, a user cannot quickly search for items. Elephant-Trax uses QR codes that a user can put on boxes, but they have to be ordered from the company.

Sortly: Sortly is similar to what we want to achieve, but aims their product at inventory for businesses instead of home use. Sortly is also expensive, and not a reasonable cost to the average homeowner.

2.3 Idea Space

This is a product made to help anyone trying to be more organized, and by extension be more prepared for emergencies. We attempt to solve the problem of unorganization by providing an easy to use system to track the room an item is in and what it is contained in as well as tracking details like price, registration numbers, descriptions and receipts. This and any other provided information will be useful when making an insurance claim following a catastrophy or when moving and finding new places for all of your things.

We aim for this app to be easy to use and sensible for the average homeowner to use. If the app is too expensive or complicated, like the other projects we observed before, then nobody will want to use it. If we achieve the sweet spot, this app will help homeowners around the world have a better idea of what they have and where their stuff is.

2.4 Technologies Leveraging

React Native: React Native is a JavaScript-like language that translates to native components in mobile operating systems today. It allows us to build an Android and iOS application from one codebase, significantly cutting the amount of code required to

program a multi-platform application. None of us have used React Native before, but some of us have had experience with JavaScript and TypeScript, which we can leverage.

AWS Amplify: Amazon Web Services (AWS) offers a solution called Amplify, which is a one-stop place for the most relevant AWS services for developing a mobile app. Amplify has a command line interface for our development environments. When we make a change on the AWS console, we deploy that and then use an amplify pull command to use those features in the application. AWS Amplify also houses our bucket data storage as well as our database management. We do have experience with AWS, but not Amplify. Luckily there isn't too much of a learning curve.

PDFKit: PDF is a javascript library that we will use to generate custom forms. These forms will contain a list of items and their descriptions that a user would like to report to their insurance agencies. The reason we chose it is because this library has many contributors on GitHub and is regularly updated.

2.5 Software/Hardware Requirements

This application will be a downloadable app for an iOS or Android phone with the latest supported operating systems. To use the full functionality of this app, a working camera is also recommended. This app also requires an internet connection to use.

Among future possible features we have the ability to locally store information allowing this app to be run without an internet connection but currently all data is stored in the cloud requiring internet for synchronization.

3 Requirements Analysis

3.1 System Architecture

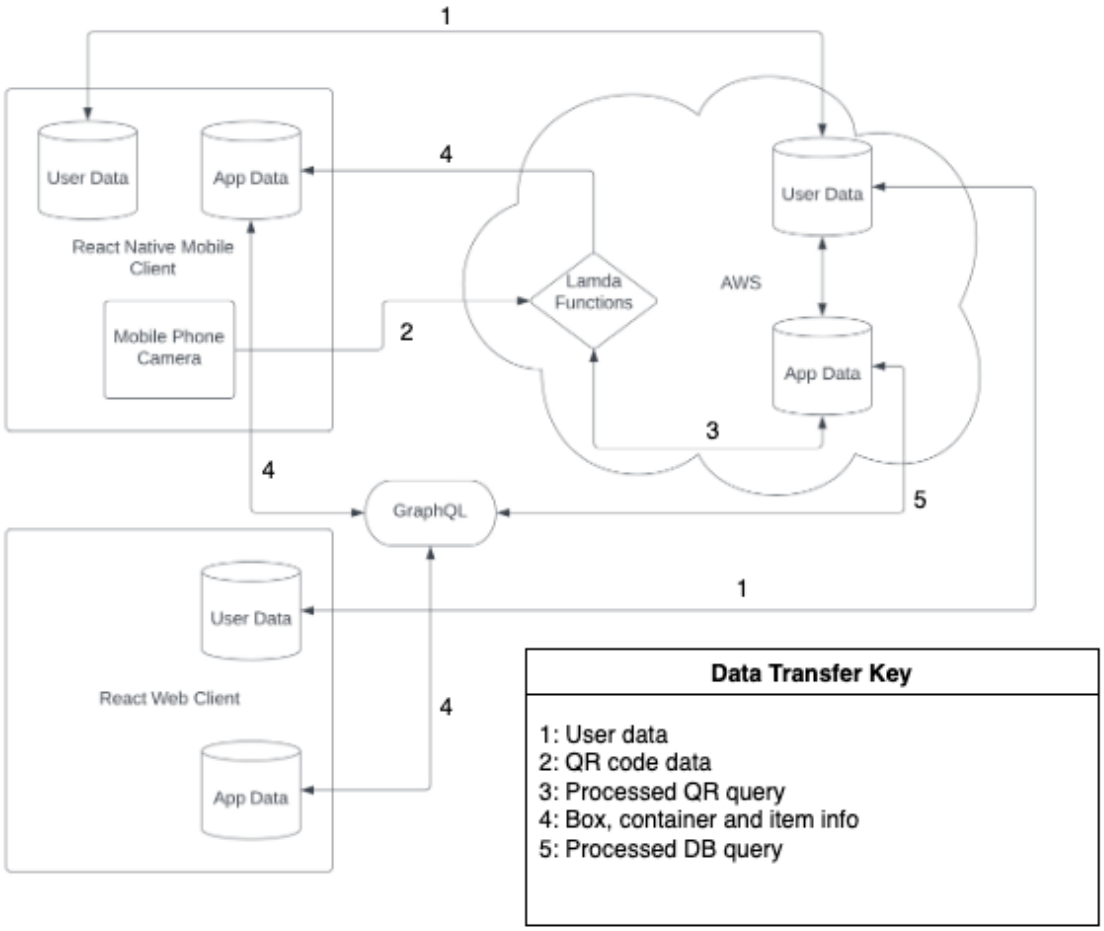
The system architecture utilized in the project is in AWS Amplify and React Native. Our AWS setup makes use of "Lambda Functions" which allow us to do a "serverless" architecture on the back-end. The front end is written in React Native, which jointly deploys native apps to both iOS and Android ecosystems.

These decisions make the application's architecture, more or less, system agnostic, able to be developed on Mac, Linux, or Windows, and able to be run on iOS and Android operating systems.

Interfacing between the AWS backend and React Native frontend is done primarily through an AWS service called AWS Amplify. Amplify allows us to take advantage of many prebuilt AWS services such as database administration and management, as well as working with users in the forms of account creation, user login/logout, and secure authentication. This setup allows us to funnel most everything through Amplify, without having to juggle too many AWS services separately. This is important because, as an integrated solution, it cuts down on bugs that come from cobbling systems together from many disconnected parts.

In addition to Amplify, the app's React Native UI is able to query the database through an architecture called GraphQL. GraphQL is a powerful JSON-based architecture for interfacing with our cloud-based database. It gives us the tools for querying and updating the database from the app's client-side code.

The following diagram shows the high level architecture of the app:



3.2 Personnel

Alan Bird is the primary designer for the UX/UI elements along with the implementation into the react native code base. Alan has also been primarily in charge of the app navigation architecture and how data propagates through the app once received from the AWS backend.

Kevin Law is primarily responsible for UX/UI development and front-end development of the mobile application in React Native. He has assisted in many parts of the main app flow, but has mainly contributed to the app's UI (thus far) in the sign in/sign up process and interfacing the app's UI with the AWS backend through Cognito. Going forward, in addition to general usability, Kevin will be tasked with UI polish in the form of styling, animations, and improved application flow.

Tyler Grant is responsible for managing/setting up the database and the communication between the front-end and backend for the database.

Eric Sims is responsible for configuring and maintaining the AWS Server. Eric is also generating AWS Lambda functions which are bits of server logic that take input from an API call from an app, and can combine information from the database, general storage and other services to produce and return a result. This functionality includes all of the logic for QR code generation and for generating an insurance claims document.

3.3 System Features

Rank 1: Bare Essentials

- B1: Login functionality using AWS Cognito
- B2: Basic UI elements and pages
- B3: Ability to log out
- B4: Viewing rooms, containers (or boxes), items
- B5: Database connected to front-end

Rank 2: Planned Features

- P1: Add pictures and descriptions to items and save them on the server
- P2: When a container is made, a QR code is generated and saved on the server
- P3: Scanning QR codes takes you directly to an individual box to view the contents
- P4: Insurance document is generated based on items listed in app
- P5: App flows smoothly and has a general positive user experience

Rank 3: Bells and Whistles

- W1: Family accounts where multiple users residing in the same house can access the same data
- W2: A companion website where users can access their home inventory on a desktop computer
- W3: Ability to search for items

4 Software Engineering Tools and Techniques

Versioning: We used GitHub for version control. Every team member made a branch whenever they worked on a new feature. We merged those branches during our spring planning meetings or other times we were working together.

Communication: We set up a Discord server to communicate via text, voice and video. It was very convenient to have one place to host all of our communication reliably. We set up channels for the different parts of our system (i.e. Server, Database, Front-End UI, etc.).

Agile/Scrum Workflow: We decided to have weekly sprint planning meetings on Wednesdays at 3:30PM. In these meetings, we reported on what we've accomplished, we made a goal for where we want our project to be next week, and then we made plans for what each of us would do to achieve that goal. Because our schedules did not permit, we did not have daily stand-ups, but kept in open communication throughout the week through Discord text and voice chat often.

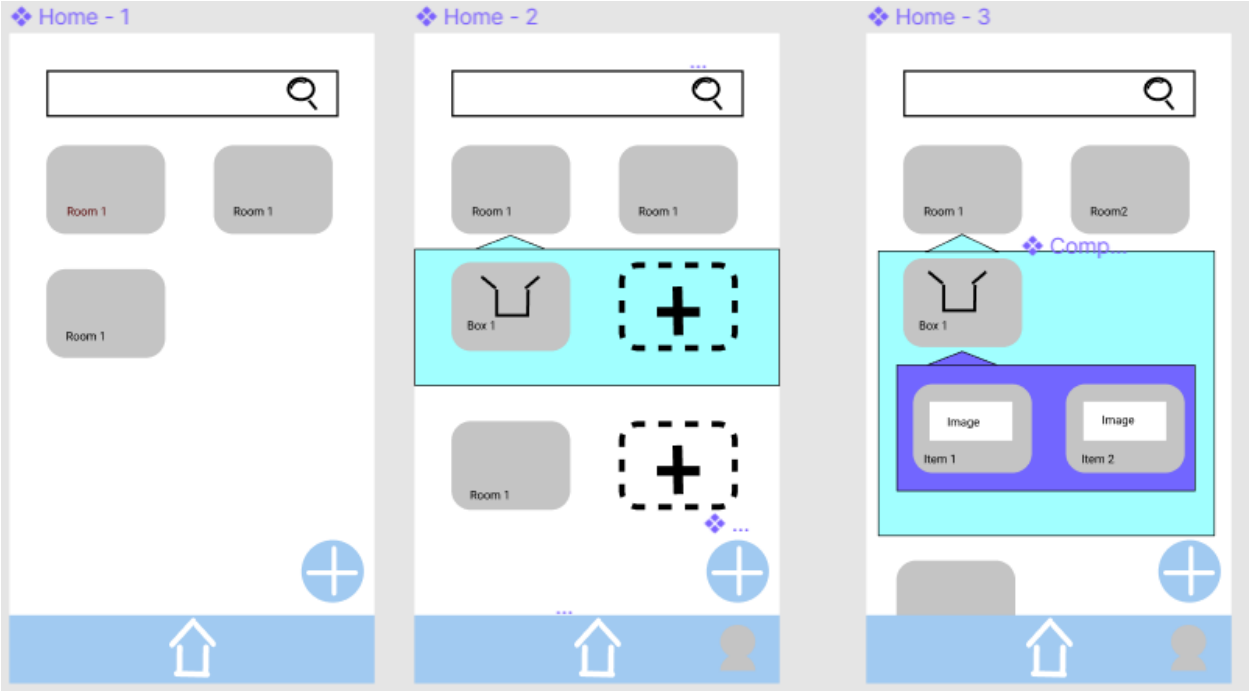
Visual Studio Code: Visual Studio code has been a powerful IDE to allow us to manage our react native files and frameworks, as well as interface with AWS. Since we have all agreed to use the same IDE, it allows us to set up environments quicker than if we used different IDE's individually. There are some react native and AWS extensions that have been helpful as well.

5 Timeline

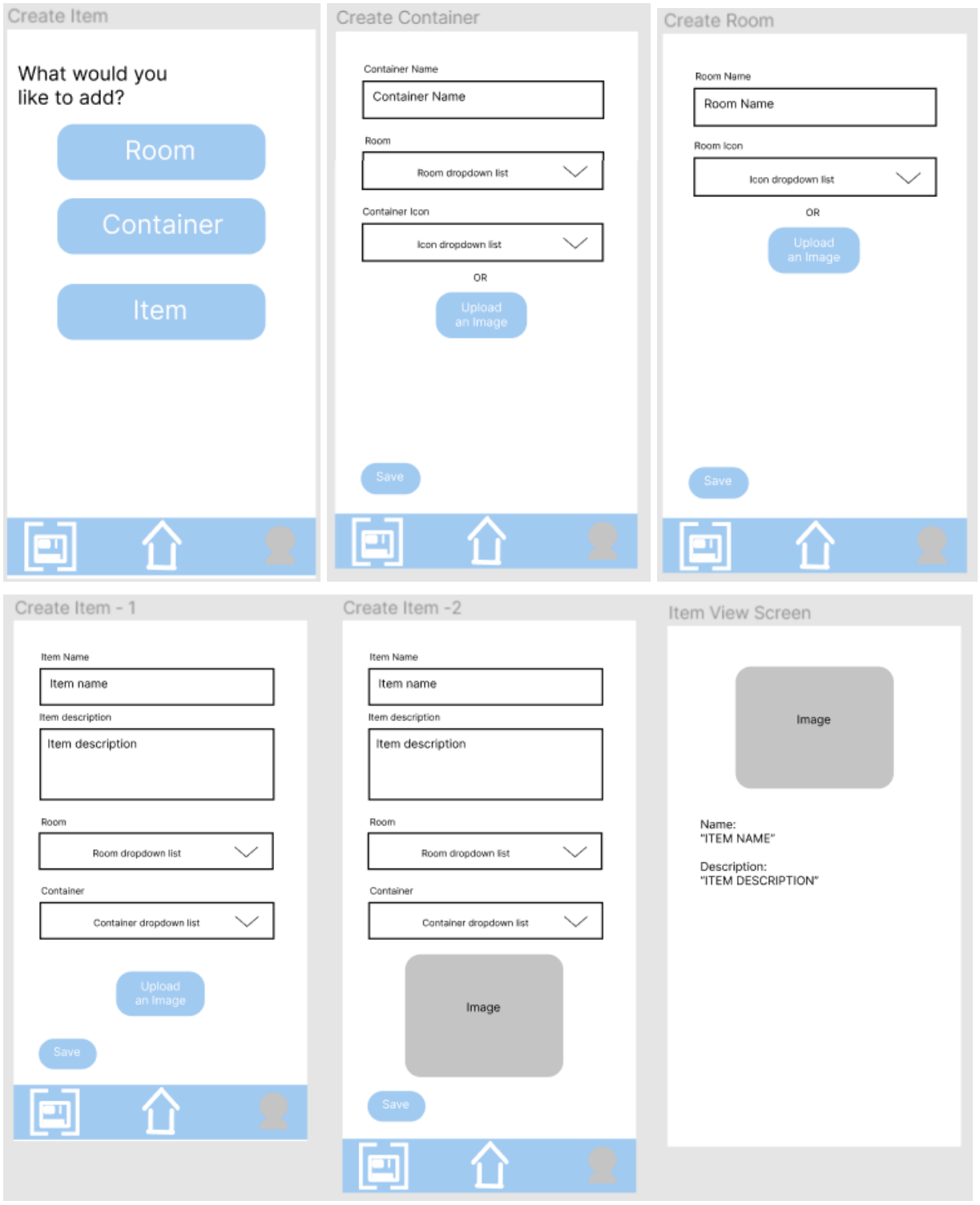
* See corresponding items in Section 3.3 “System Features”

Item	**Date**	**Person**
B1	Already completed	–
B2	Already completed	–
B3	Already completed	–
B4	Already completed	–
Design Document 1	9 September	Eric Sims
B5	23 September	Alan Bird
P1	30 September	Tyler Grant
P2	30 September	Eric Sims
Demo 1 (Alpha)	30 September	Kevin Law
P3	14 October	Eric Sims
User Guide 1	21 October	Tyler Grant
Design Document 2	21 October	Alan Bird
P5	28 October	Kevin Law
Demo 2 (Beta)	28 October	Eric Sims
P4	4 November	Eric Sims
W1	18 November	Kevin Law
W2	2 December	Alan Bird
W3	2 December	Eric Sims
Design Document 3	2 December	Tyler Grant
User Guide 2	2 December	Eric Sims
Demo 3 (Final)	9 December	Alan Bird

6 Appendix A: UI Sketches



**These three screens illustrate the traditional flow of the home screen allowing you to view rooms, their containers, and the items (and photos) in those containers.*



From the create item flow the user will be able to make any item, room or container specifying any required details.

7 Appendix B: Use Cases

Use case: I am a user with a storage room full of decorations for different seasons. Having my house decorations match the current season is a big priority for me but it is a huge hassle to find the decorations for next when all of my boxes look the same. With the Boxie app I can label each box and see exactly what I have in each box so when I need just the pumpkins from my halloween box and the fake turkeys from the Fall decorations I can find the exact box with the click of a button.

Use case: I am a long time user of the Boxie app and have everything in both of my storage rooms saved along with their receipts from when I bought them. One of our storage rooms flooded, ruining the contents of many of my boxes beyond recognition. Because of my records in the Boxie app I am able to make an exact insurance claim for everything I have lost allowing me to replace everything I have covered.

Use case: I am preparing to move to a new house! I have over 50 boxes of various shapes and sizes that all get off the moving truck waiting to find a new home. I know where individual items should go in the house. I just need to know what boxes have what items, luckily all of my boxes were stored in the Boxie app before I loaded them onto the truck. With the Boxie scanner I can quickly tell what are the contents of a given box and what room it should be headed to.

8 Changelog

9/2/22: Updated architecture graph to contain key to describe what data is passed between modules.

9/7/22: Updated the Project Timeline to reflect feature release dates as well as semester due dates

9/9/22: Update formatting for Project Timeline and assigned primary responsibility

9/16/22: Update for project timeline. Removed following:

Deliverables	**Due Date**	**Owner**
Updated Design Document Due	9-9–22	Eric Sims
QR Code Generates and returns code to be printed	9-23-22	Eric Sims
Database hooked up, enabling the addition/deletion of rooms/boxes/items	9-23-22	Alan Bird
QR Scanner is able to return ID of box	9-30-22	Eric Sims
Images of boxes/items stored in D3	9-30-22	Tyler Grant
Alpha Demo	9-30-22	Kevin Law
Website basic UI	10-7-22	Alan Bird
Website login	10-21-22	Kevin Law
User Guide Draft Due	10-21-22	Tyler Grant
Insurance Claim Packet available in PDF form	11-4-22	Eric Sims
Website UI full functionality	11-25-22	Kevin Law
Item search functionality in app and website	12-2-22	Tyler Grant
Project Demo Day	12-9-22	Alan Bird

9/16/22: Changed Section 3.3 to more specific deliverables and corresponded features with the timeline in Section 5