Performance Livestock Analytics

Development Progress Presentation - DEC1708

Project Stakeholders

Michael Rhodas
Team Leader

Rachel Hartman
Communication Lead

Jacob Johnson
Key Idea Holder

Ken Kohl
Website Master

Jeff Murray
Key Idea Holder

Dane Kuper
CEO and Co-Founder

Dustin Balsley
COO and Co-Founder

Daji Qiao
Faculty Advisor

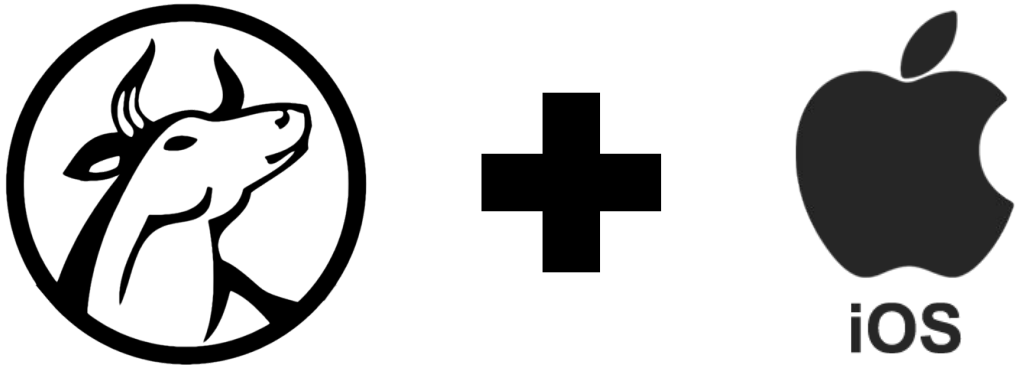
Performance Livestock Analytics

- Founded in 2015
- Started working with the Ag. Start-Up Engine in late 2016
- PLA wants to bring data science to farmers
- Interested in producing the most reliable solutions for farming

Problem Statement

We will create an intuitive iOS application for tracking veterinary medical information for cattle farmers.

This application will help our clients record and monitor medical treatment and recovery information for their animals and facilitate data-driven analysis and decision making to help our users make more informed economic and medical decisions.



Market Research

Unique to our application:

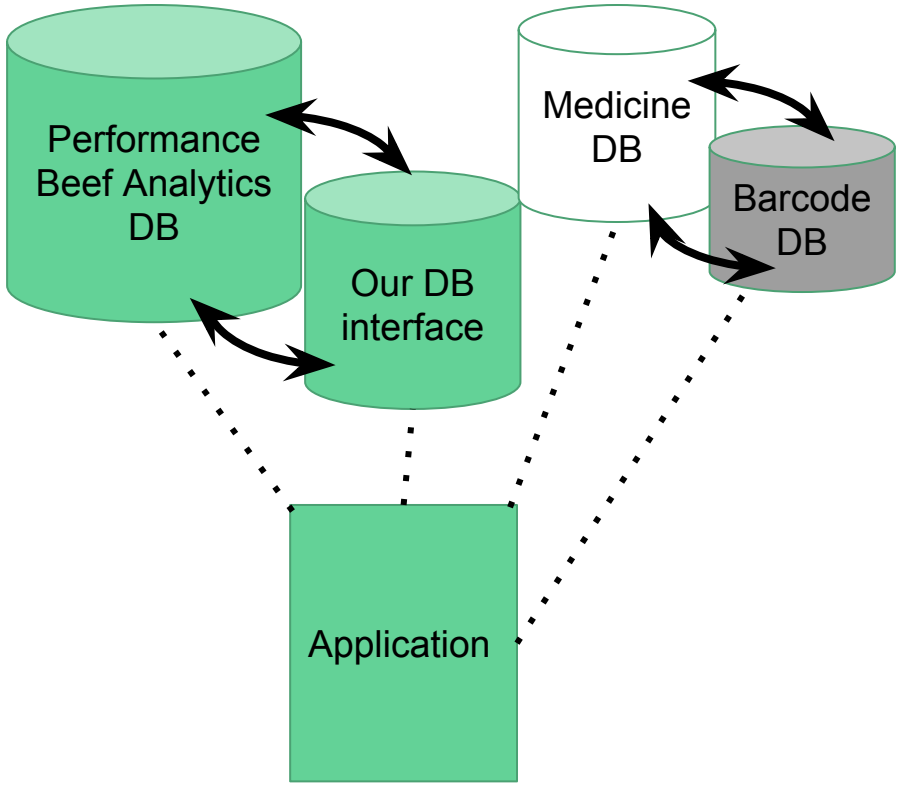
- Provide tracking for individual animals medications or doses of the drugs administered
- Keep track of most frequently used medications.
- Usability and speed a priority in the field
- Geared particularly for the area of cattle health.

AMAFERM®

CattleMax™

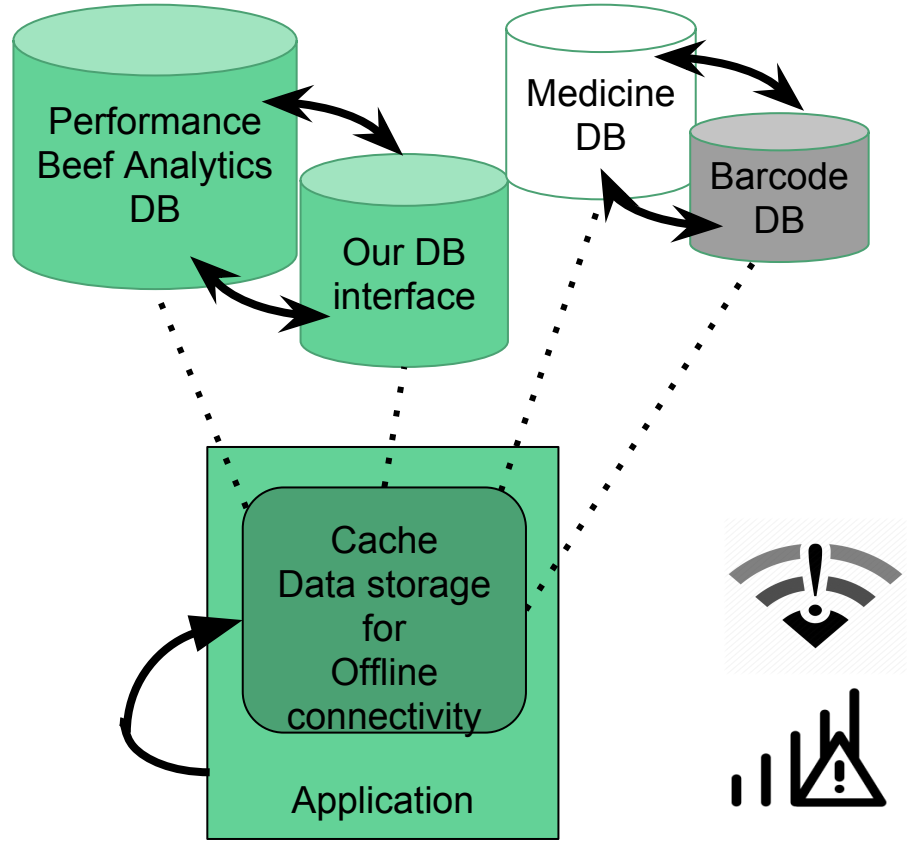
Functional Requirements

- Compile livestock medicine database
- Communication with Performance Beef database
- Systems for scanning heads of cattle and medical information
- Simple and intuitive user interface
- Systems of data analysis for analytics views and decision making aids



Non-Functional Requirements

- Offline connectivity and usability
- Recover after an application failure
- Functionality on all scales, small to large
- Easy to use with a logical workflow
- Easily maintained and expanded



Technical Constraints & Considerations

- Operating within the Agile development process
- Must adhere to Apple’s development and release guidelines for the AppStore
- Policy is particularly strict when dealing with any medical information, regardless of whether it is intended for use by humans or otherwise
- To avoid issues with this particular guideline, we will avoid recommending specific medications or dosage information

Design Overview and Insights

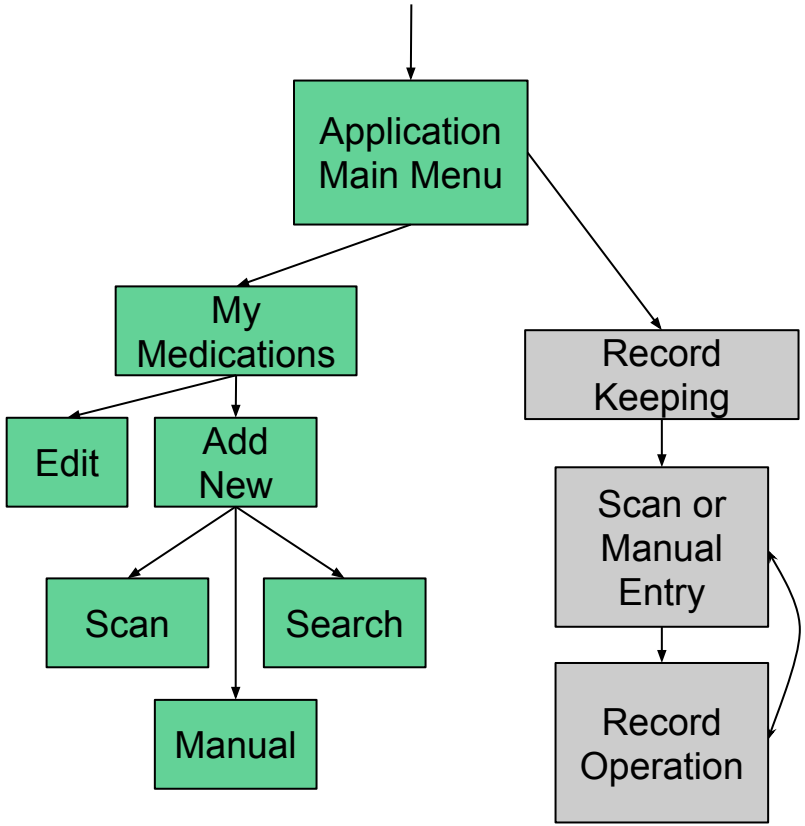
- Client Device
  - iOS application used for medication and medication administration data.
- System IO
  - Barcode recognition and association with medications.
- Local Cache
  - Stores personalized medication list and any administration done offline.
- Database
  - Currently utilizing Firebase, will be migrated to Performance Beef's SQL database.

Functional Decomposition

Two primary user stories:

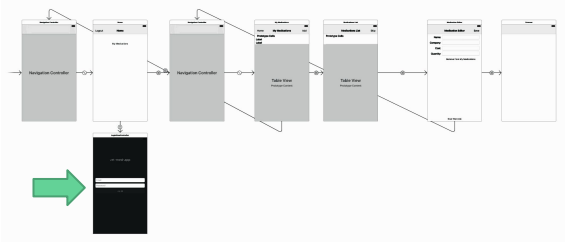
- My Medications
  - Enter medications by searching, scanning, or manual entry.
  - Medications entered are stored in a personalized medication list for speed of access in the future.

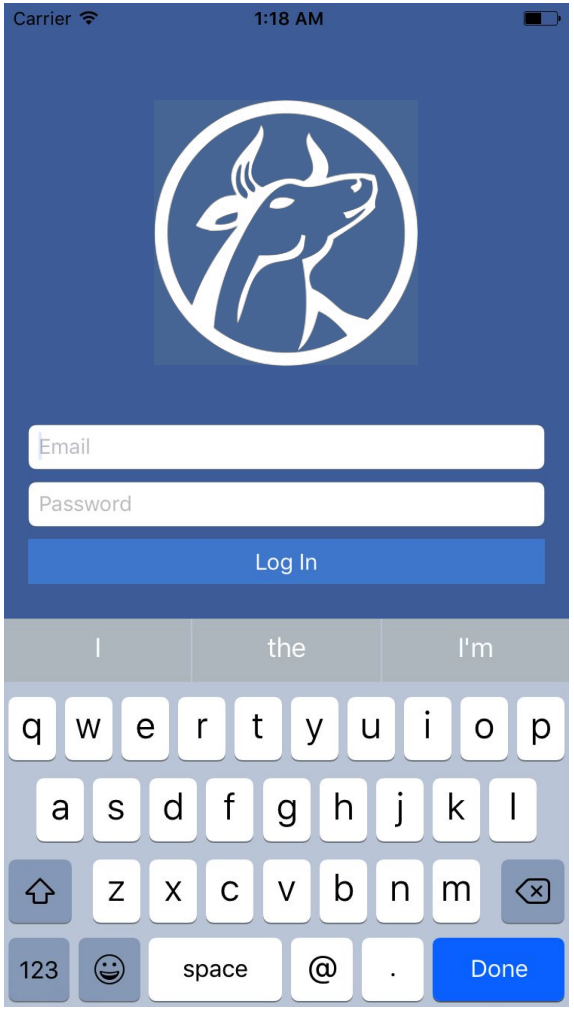
- Administer Medications
  - Cattle identifiers and medication quantity and cost are manually entered.
  - This data is stored for cost efficiency analytics.



Login View

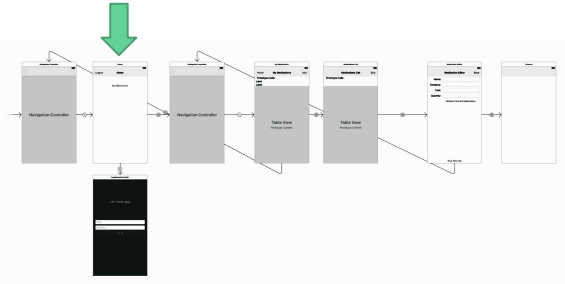
- Firebase OAuth support
- Local validation testing
- Server-side authentication
- Meaningful failure alerts
- Only required for initial login
- Once authenticated can continue offline

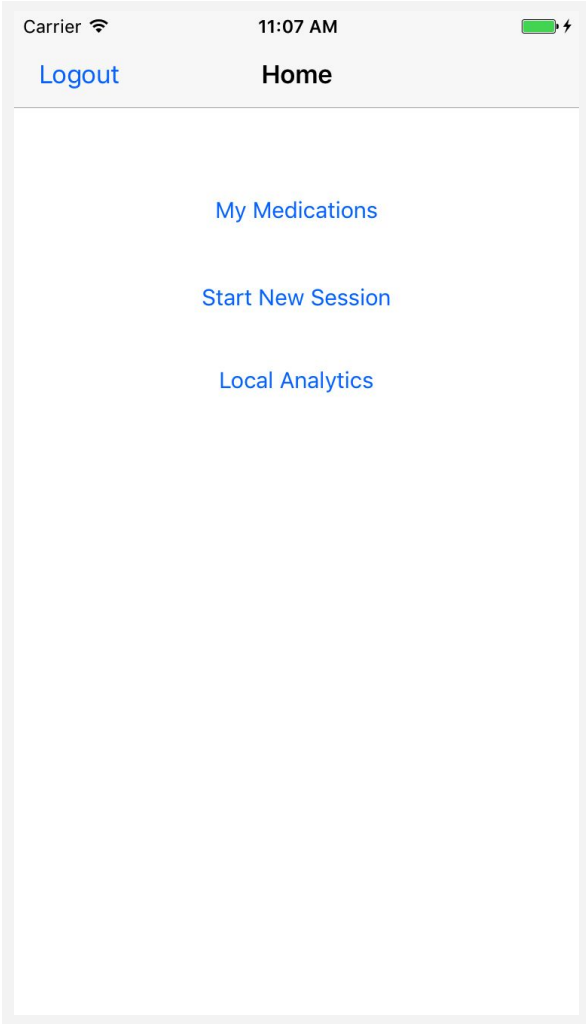




Home View

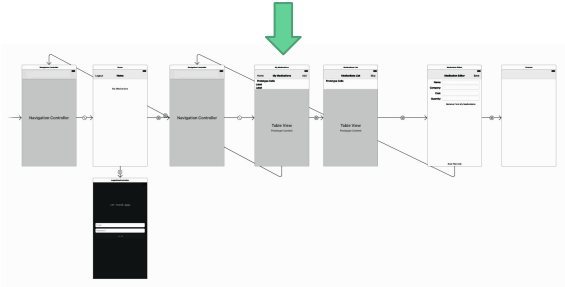
- Navigation controller root node
- Currently a simple menu view
- Will expand with ‘recent’ populated data
- This view will likely grow in complexity as the project expands

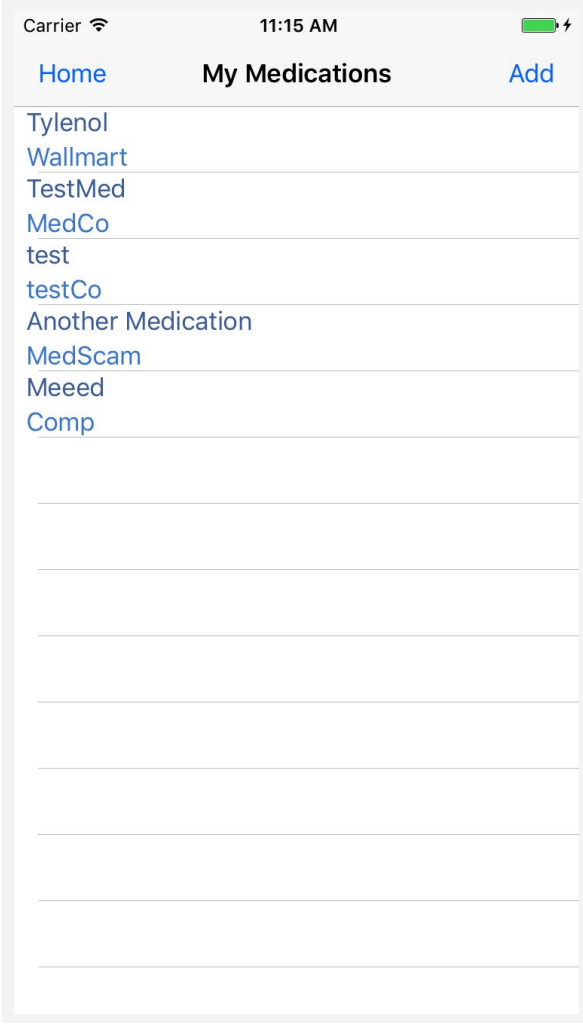




My Medications View

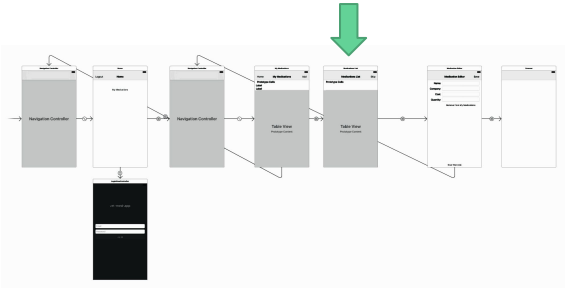
- Allows farmers to create a favorites list
- Supports offline functionality quickly
- Helps keep iterative user stories fast
- Supports custom, searched, and scanned additions
- All stored locally and synced to Firebase

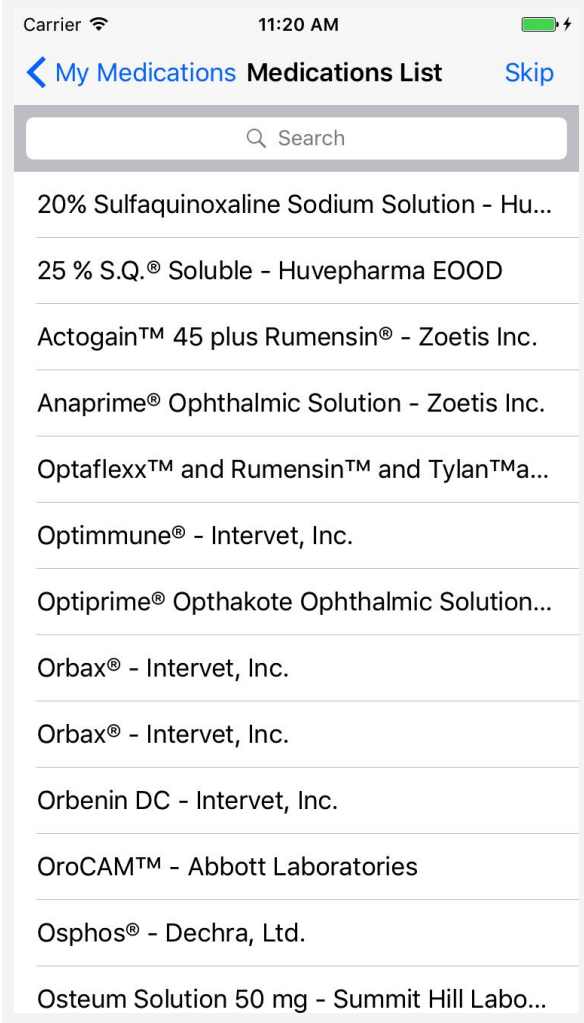




Master Medications View

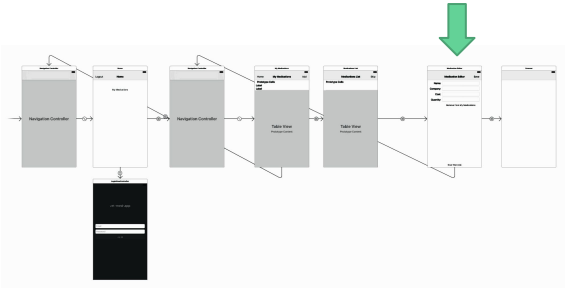
- Allows farmers to find known medications
- Compiled information we populated ourselves
- Search functionality included
- Can also be skipped for custom additions
- Data is housed server-side and pushed to devices
- Master list is only editable by administrators

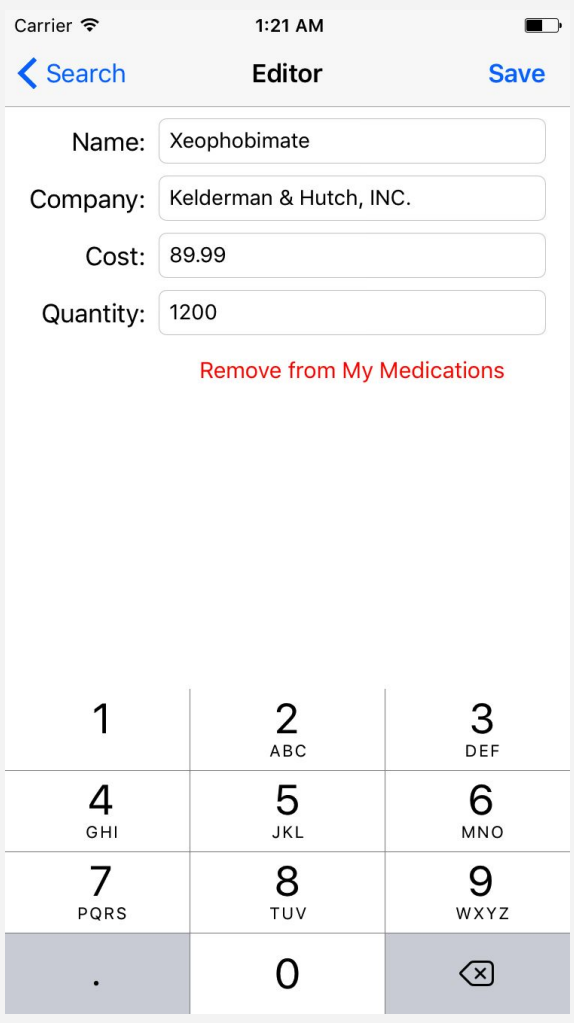




Medication Editor View

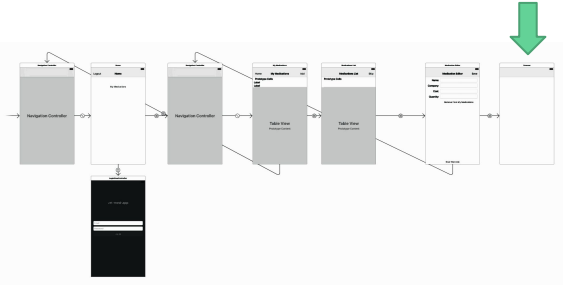
- Allows farmers to manually edit and expand data
- Currently track: name, company, cost, and quantity
- Fields like cost and quantity can be updated with usage and will drive analytics portion of application
- Custom medication class can easily be expanded
- Allows additions, edits, and removal of medications

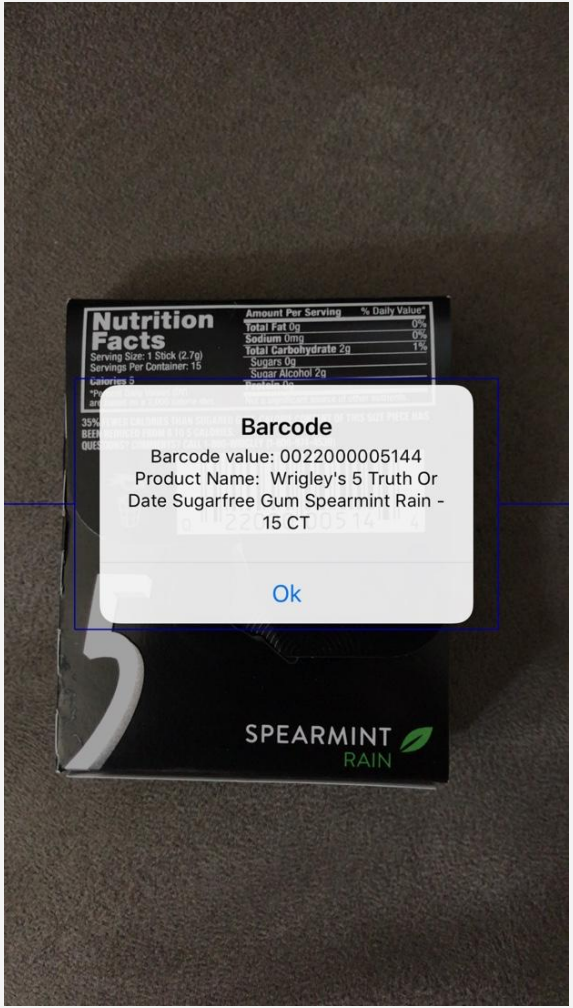




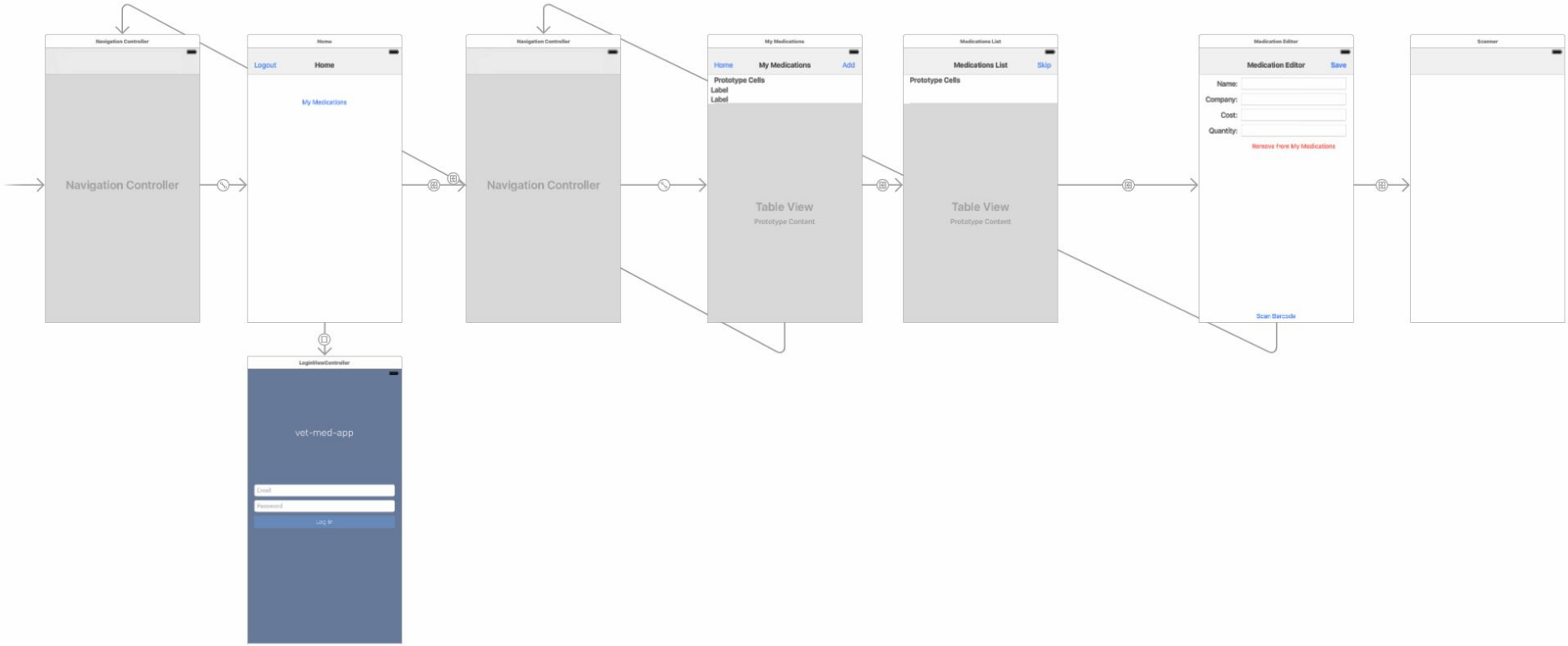
Medication Scan View

- Proof of concept for later iterations and designs
- Allows farmers to quickly find their medications
- Supported through external lookup API
- Modular design and use makes extension and changes easy without breaking data constraints





Navigation & Testing



Design Decisions

Risk: General lack of experience and steep technology learning curve

Mitigation: Extensive research phase and demo development

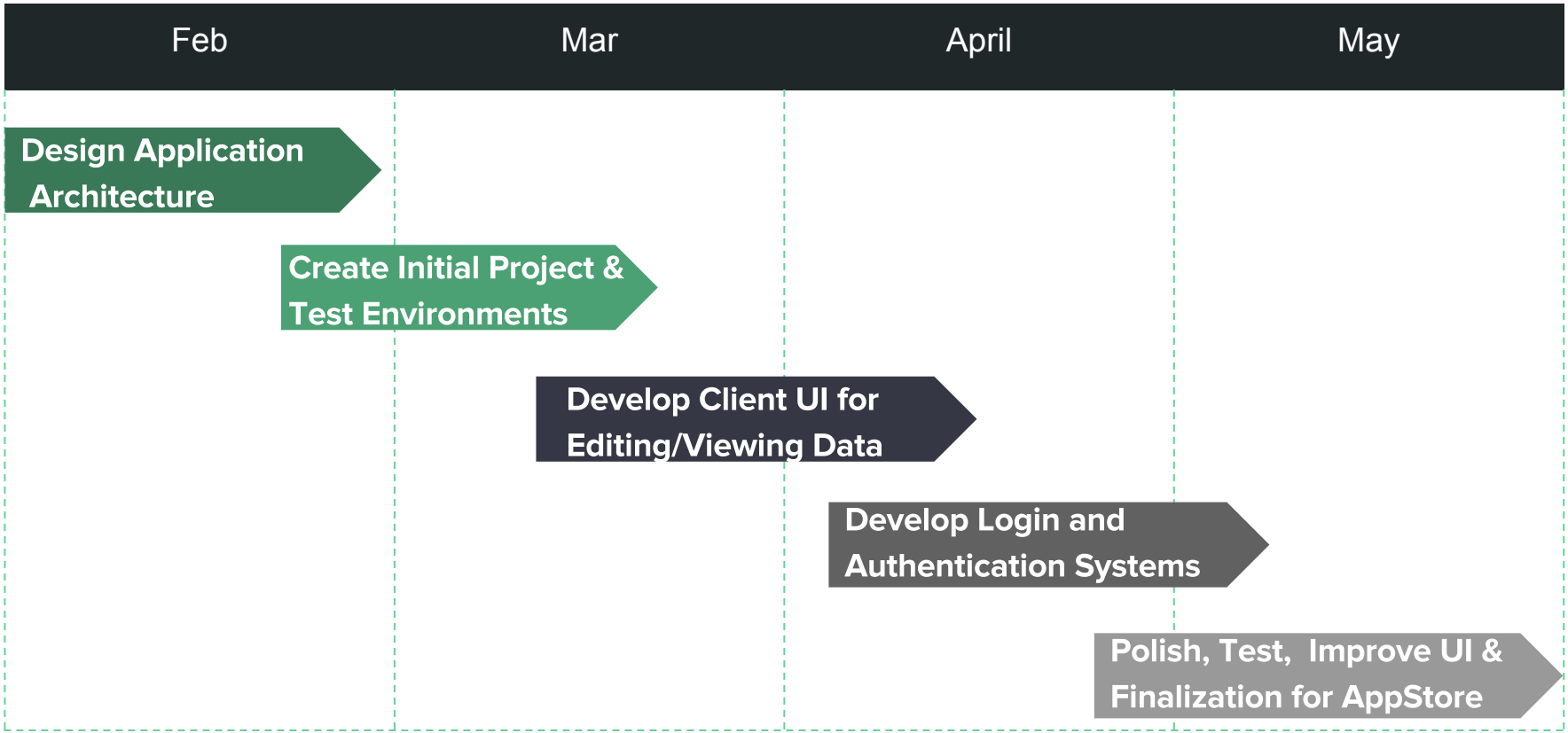
Risk: Integration with existing Performance Livestock systems

Mitigation: Temporary Firebase backend with PHP/SQL replacement imminent

Risk: Apple AppStore release process and validation testing

Mitigation: Following preferred practices and available guidelines

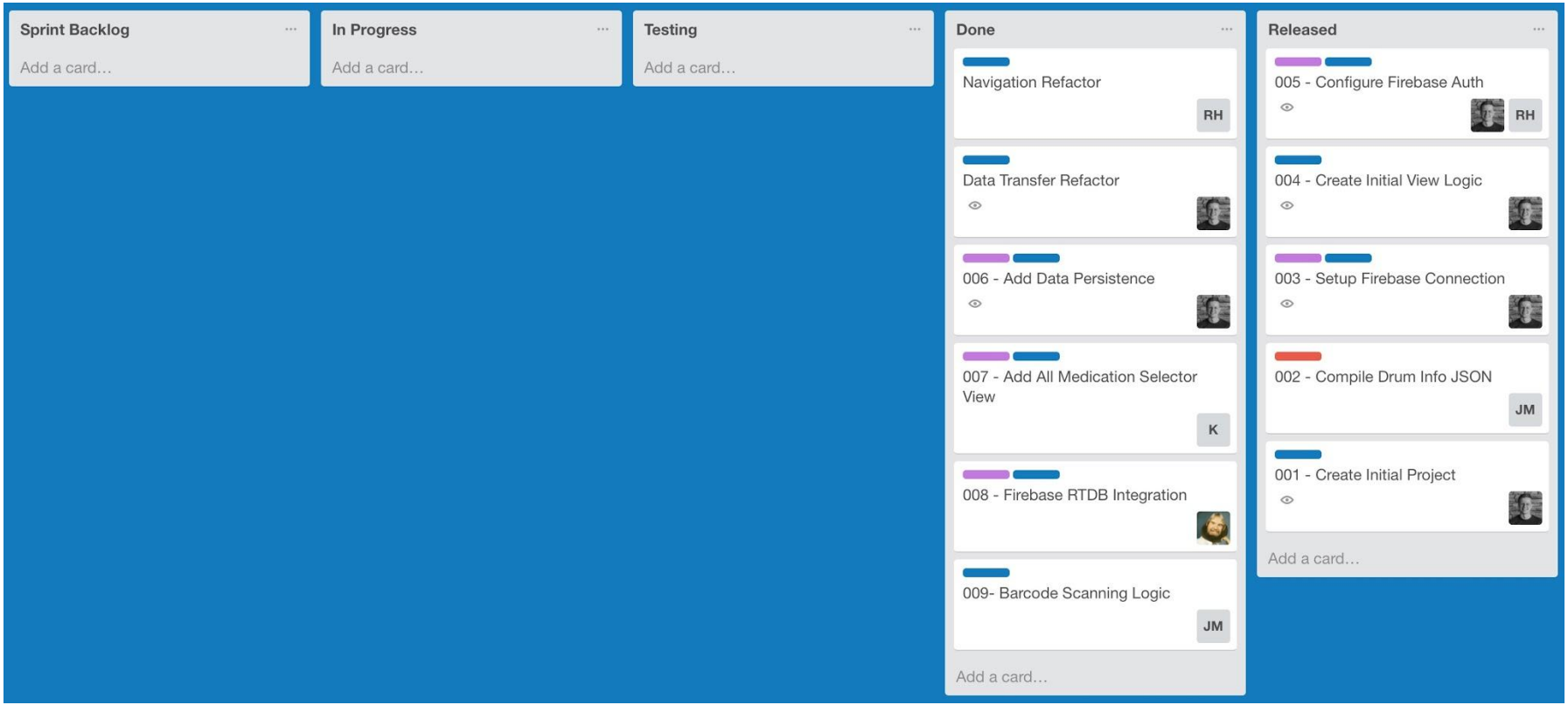
Project Timeline



Current Project Status

- Alpha build completed with full firebase support.
- This includes the “my medications” user story as well as higher level scene management and navigation logic.
- Served as a self contained and encapsulated method for learning iOS’s controller, delegate, and navigator systems.
- Also aided in the understanding of client-side data persistence and firebase integration for authentication and real-time database.

Feature Contributions

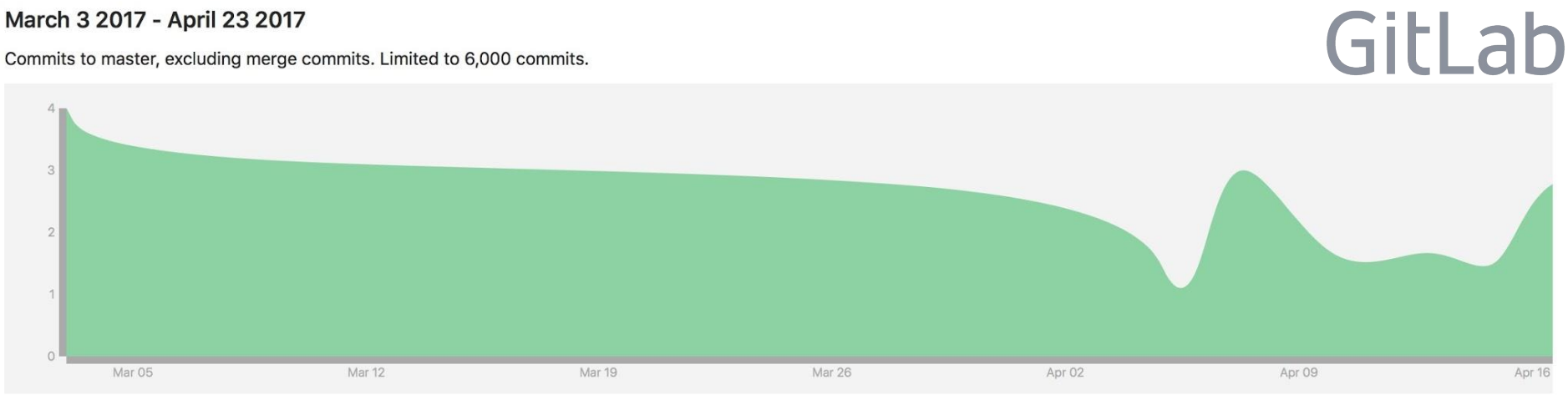


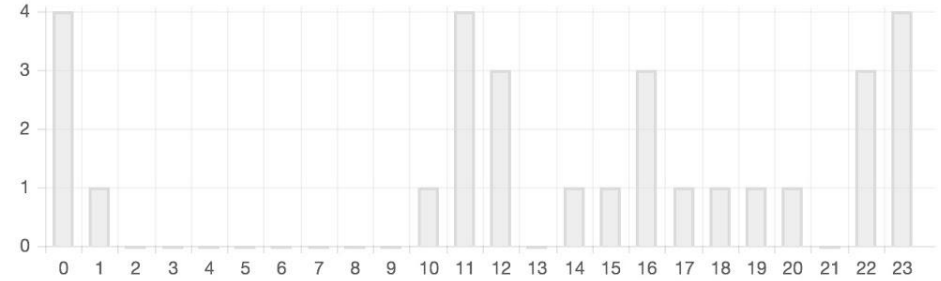
Next Semester

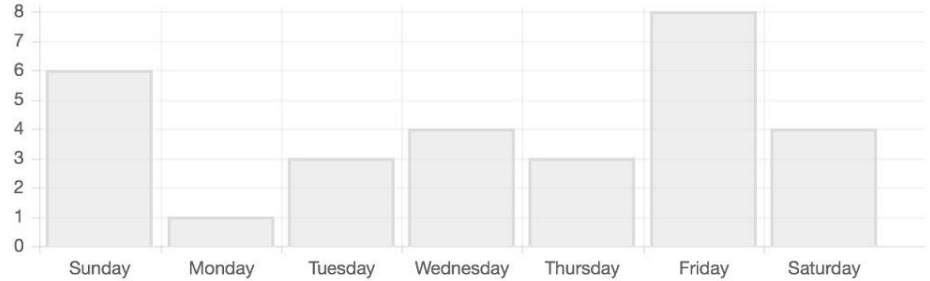
The second semester of the project will be dedicated toward improvements product design and feature expansion.

- Barcode scanning for medication identification
- Bluetooth communication for medicine application
- Analytics for tracking and data-driven decision making
- Medical diagnostics through RFID or other external devices
- Extended or highly polished user interface
- Release validation and testing

Contributions



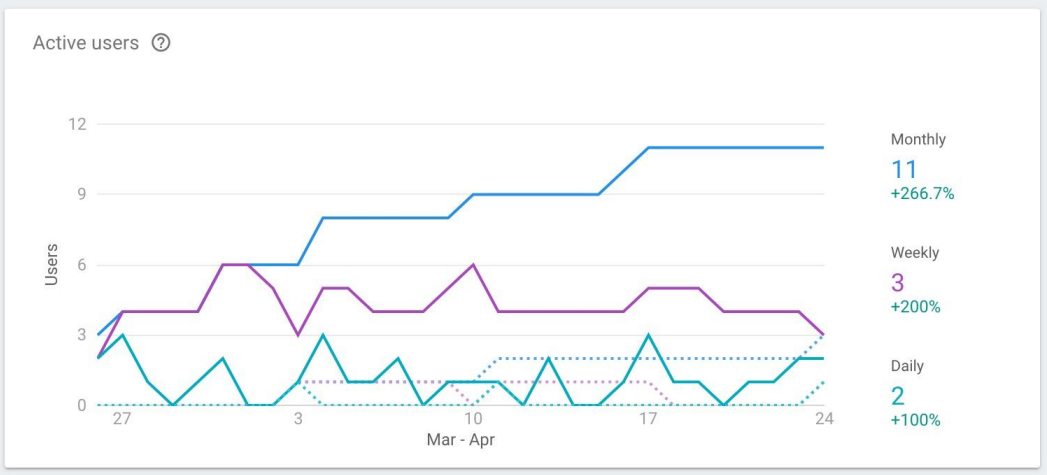




Barcode Scanning

- Barcode lookup API is through upcdeal.us
- Utilizes the AVCaptureSession and AVCaptureDevice classes available for iOS development to capture camera data
- This input is then listened to, and looks for a metaDataObject that can be read as an AVMetadataMachineReadableCodeObject or barcode
- Upon finding this code, the numerical value is passed to a method that calls the API
- After this call returns, the data is stored within the controller for the scanner, and accessed by medication editor’s controller to populate input fields

Firebase

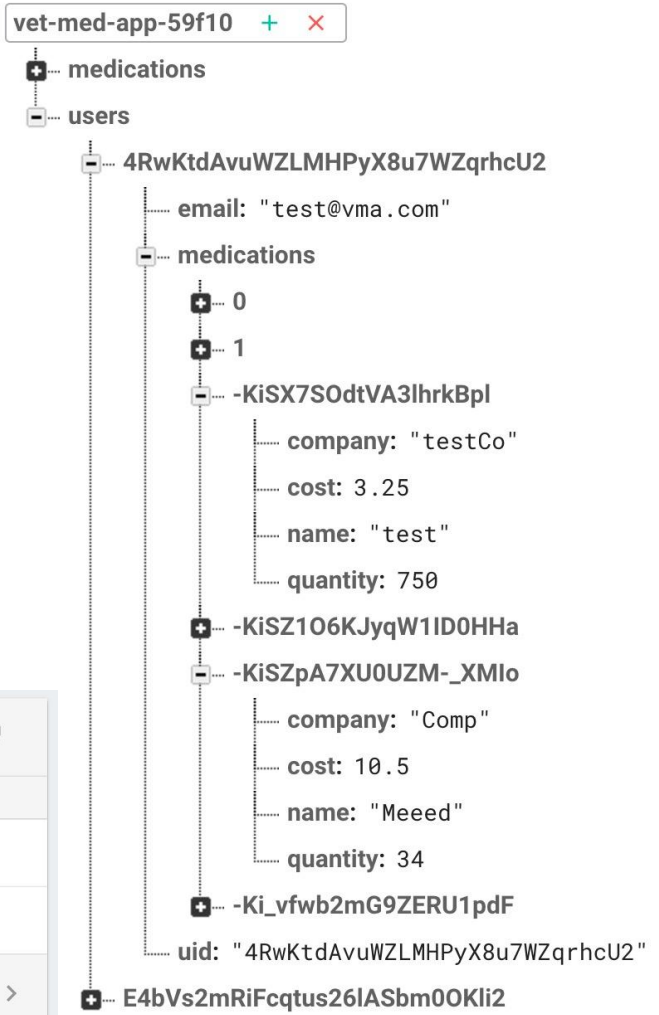


Search by email address or user UID

ADD USER

Email	Providers	Created	Signed In	User UID ↑
test@vma.com	✉	Apr 4, 2017	Apr 23, 2017	4RwKtdAvuWZLMHPyX8u7WZqrhc..
testuser@test.com	✉	Apr 23, 2017	Apr 23, 2017	E4bVs2mRiFcqtus26lASbm0OKli2

Rows per page: 50 1-2 of 2



View Controllers

- Developed entirely in Swift
- Firebase integrations shown
- Features Swift's optionals paradigm and nil checks
- @IBAction method definitions
- Notice segue logic

Swift

```
//
//  HomeViewController.swift
//  vet-med-app
//
//  Created by Jacob Johnson on 4/4/17.
//  Copyright © 2017 Performance Livestock Analytics. All rights reserved.
//

import UIKit
import Firebase
import os.log

class HomeViewController: UIViewController {

    //MARK: ViewDidLoad
    override func viewDidLoad() {
        super.viewDidLoad()
        // check for logged in user
        if FIRAuth.auth()?.currentUser?.uid == nil {
            // if no active user perform logout sequence immediately
            logoutAction(self)
        }
        else {
            // otherwise process user data as needed
            print("Logged in with user: \(FIRAuth.auth()?.currentUser?.email! ??
                "Error getting user information.")")
        }
    }

    //MARK: Actions
    @IBAction func logoutAction(_ sender: Any) {
        do {
            // logout of Firebase if possible
            try FIRAuth.auth()?.signOut()
            // show the login screen
            performSegue(withIdentifier: "logoutSegue", sender: self)
        } catch let logoutError {
            // print logout error if encountered
            fatalError(logoutError as! String)
        }
    }

}
```

Navigation Systems

- Segue prepare funct
- Outlet usages
- Firebase user data references and alterations
- Database tree structure

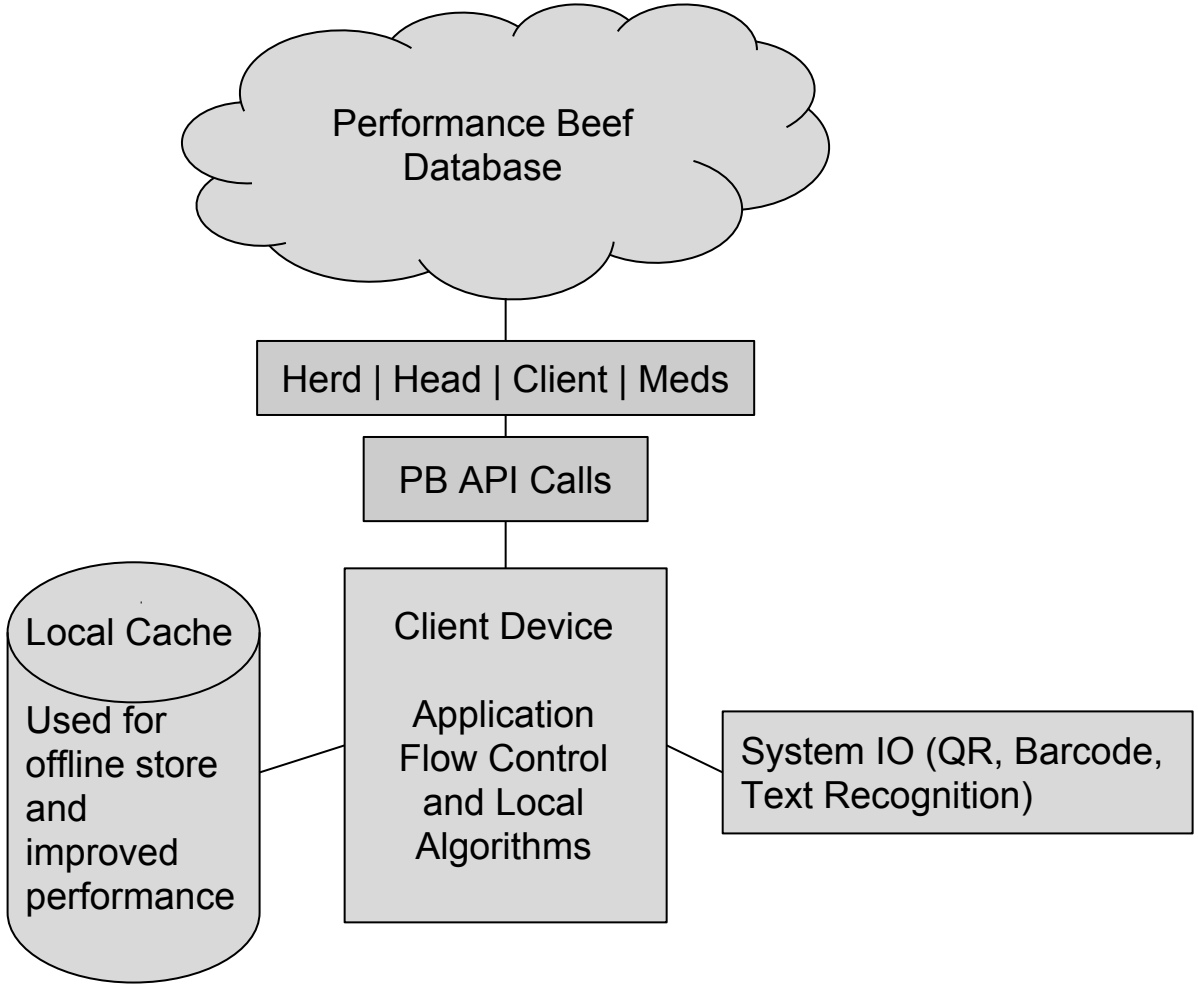
Swift

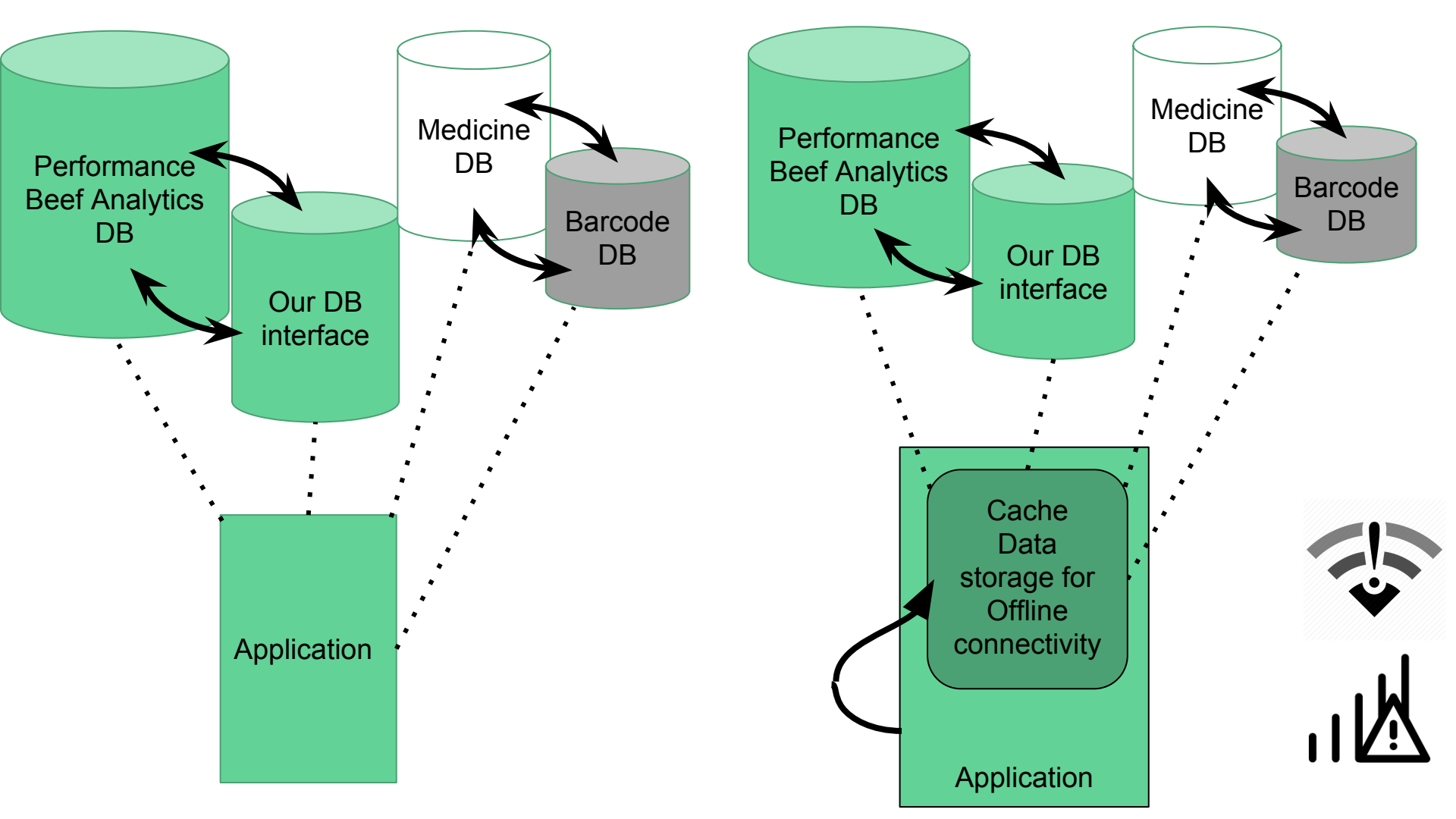
```
// called just before the undwind sequence when an exit segue is triggered
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    super.prepare(for: segue, sender: sender)
    // check which button was pressed to trigger (should be delete or save)
    if sender is UIBarButtonItem {
        let button: UIBarButtonItem = sender as! UIBarButtonItem
        // if save button pressed
        if(button === saveButton) {
            // grab field text values
            let name: String = nameTextField.text ?? ""
            let company: String = companyTextField.text ?? ""
            let cost: Double = Double(costTextField.text!) ?? 0.0
            let quantity: Int = Int(quantityTextField.text!) ?? 0
            // set the medication to be passed to medicationTableViewController
                after the unwind segue
            medication = Medication(name: name, company: company, cost: cost,
                quantity: quantity)

            //Add data to Firebase
            ref = FIRDatabase.database().reference()

            let user = FIRAuth.auth()?.currentUser
            if user != nil {
                let uid = user!.uid
                let medRef =
                    self.ref.child("users").child(uid).child("medications").
                    childByAutoId()
                medRef.setValue(["name" : name, "company" : company, "cost" :
                    cost, "quantity" : quantity])
            } else {
                // Go to user login
            }

        }
    }
    else {
        // delete button pressed
        didDelete = true
    }
}
```


Performance Beef Database
Herd | Head | Client | Meds
PB API Calls
Local Cache
Used for offline store and improved performance
Client Device
Application Flow Control and Local Algorithms
System IO (QR, Barcode, Text Recognition)


Performance
Beef Analytics
DB
Medicine
DB
Barcode
DB
Our DB
interface
Application
Performance
Beef Analytics
DB
Medicine
DB
Barcode
DB
Our DB
interface
Cache
Data
storage for
Offline
connectivity
Application

Cost Analysis

Item	Cost
Apple Developer License	$99 a year
Bluetooth Transmitter w/ EID reader	$50
Barcode API	$99 a month

References:
http://www.cals.iastate.edu/news/releases/performance-livestock-analytics-establishes-presence-iowa-state-university-research

Images:
https://www.adiants.com/wp-content/uploads/2016/07/app-ios-png-4.png

https://1.bp.blogspot.com/-YIfQT6q8ZM4/Vzyq5z1B8HI/AAAAAAAAAAc/UmWSSMLKtKgtH7CACEIUp12zXkrPK5UoACLcB/s1600/image00.png

https://cdn2.iconfinder.com/data/icons/agriculture-1/512/Vaccination-512.png

http://www.performancelivestockanalytics.com/uploads/2/5/6/2/25626603/4569492.jpg?1438359277

http://www.filemaker.com/solutions/customers/stories/images/1866-image4.jpg

http://www.isupark.org/file/6568