Performance Health

A Modern Vet Med Web Application

Michael Rhodas, Jeff Murray, Jacob Johnson, Ken Kohl, Rachel Hartman

Team: DEC1708

Advisor: Daji Qiao (ECpE)

Client: Performance Livestock Analytics

Problem Statement:

Create a MVP web application that will help our clients record and monitor medical information for their animals and facilitate data-driven analysis and decision making from the PLA application to help our users make more informed economic decisions.

Solution:

We have created a modern HTML5 web application with React component design, Bootstrap UI, and Firebase real-time data synchronization and authentication.

Intended Users & Uses:

- Designed with farmers in mind
  - Intuitive user interface and interaction scheme
  - Real farm insights provided by the PLA team
- Built for usage in the field
  - Bootstrap UI components support a great experience on both web and mobile platforms
  - Real-time data synchronization and reactive architecture make multitasking easy
- Data integrity as a top priority
  - Firebase cloud storage and native APIs helps ensure secure and reliable data for PLA consumption

Project Testing:

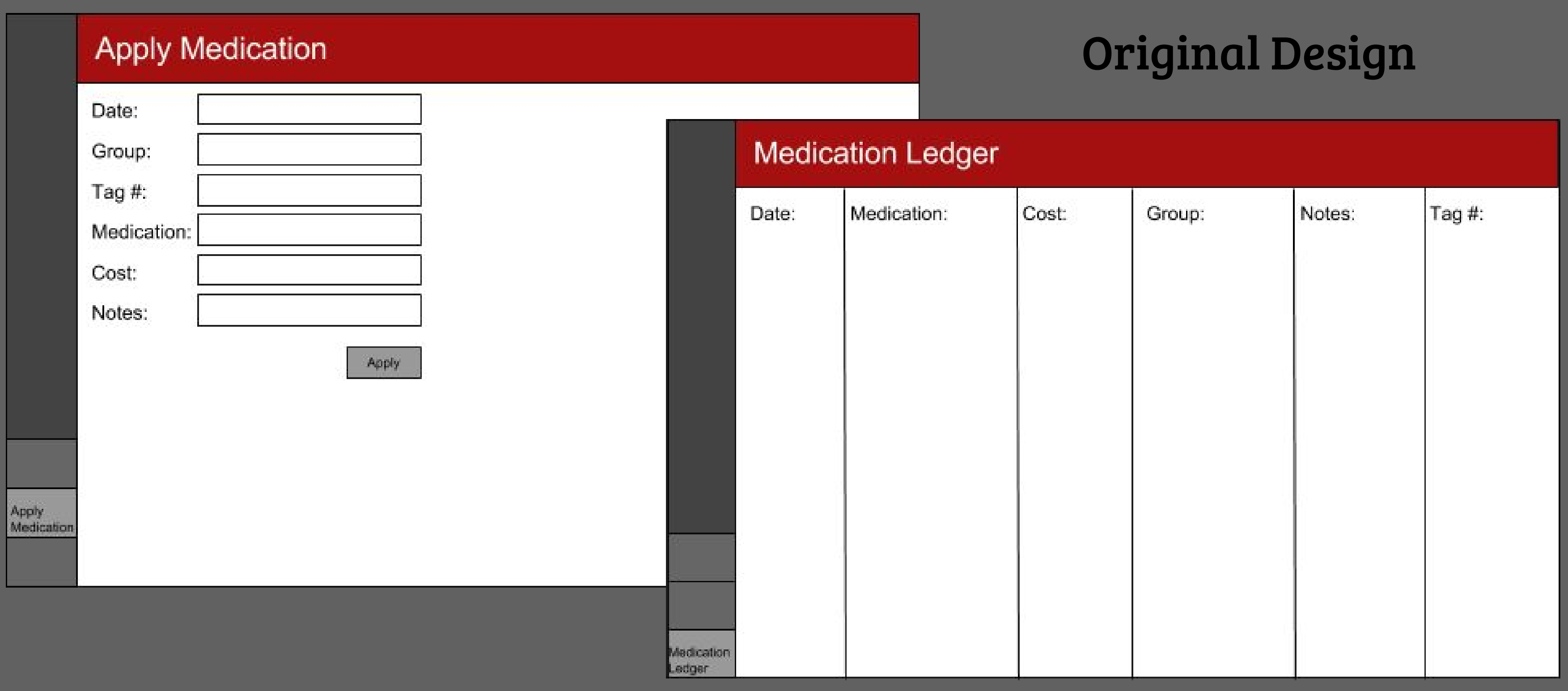
- Jest testing environment integration (image below)
- Tests primarily focused on ensuring proper rendering of key components and views

Operating Environment:

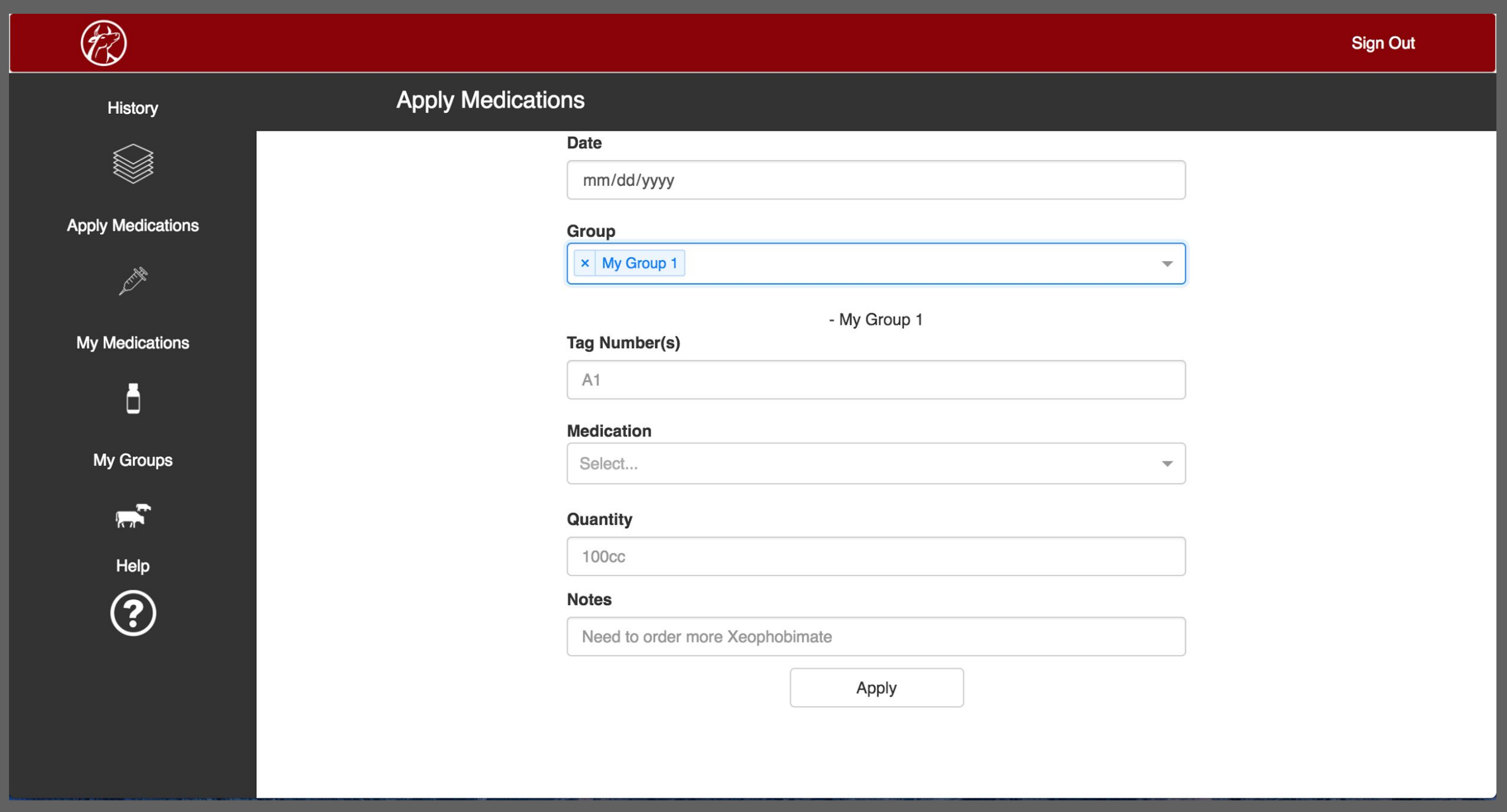
- Website application environment
  - Supports both web and mobile consumption via HTML5 and ES6 JavaScript compliant browsers
- Internet connection required
  - However, application can be adapted for Firebase offline data store and cached synchronization

Concept Sketches & Finished Product:

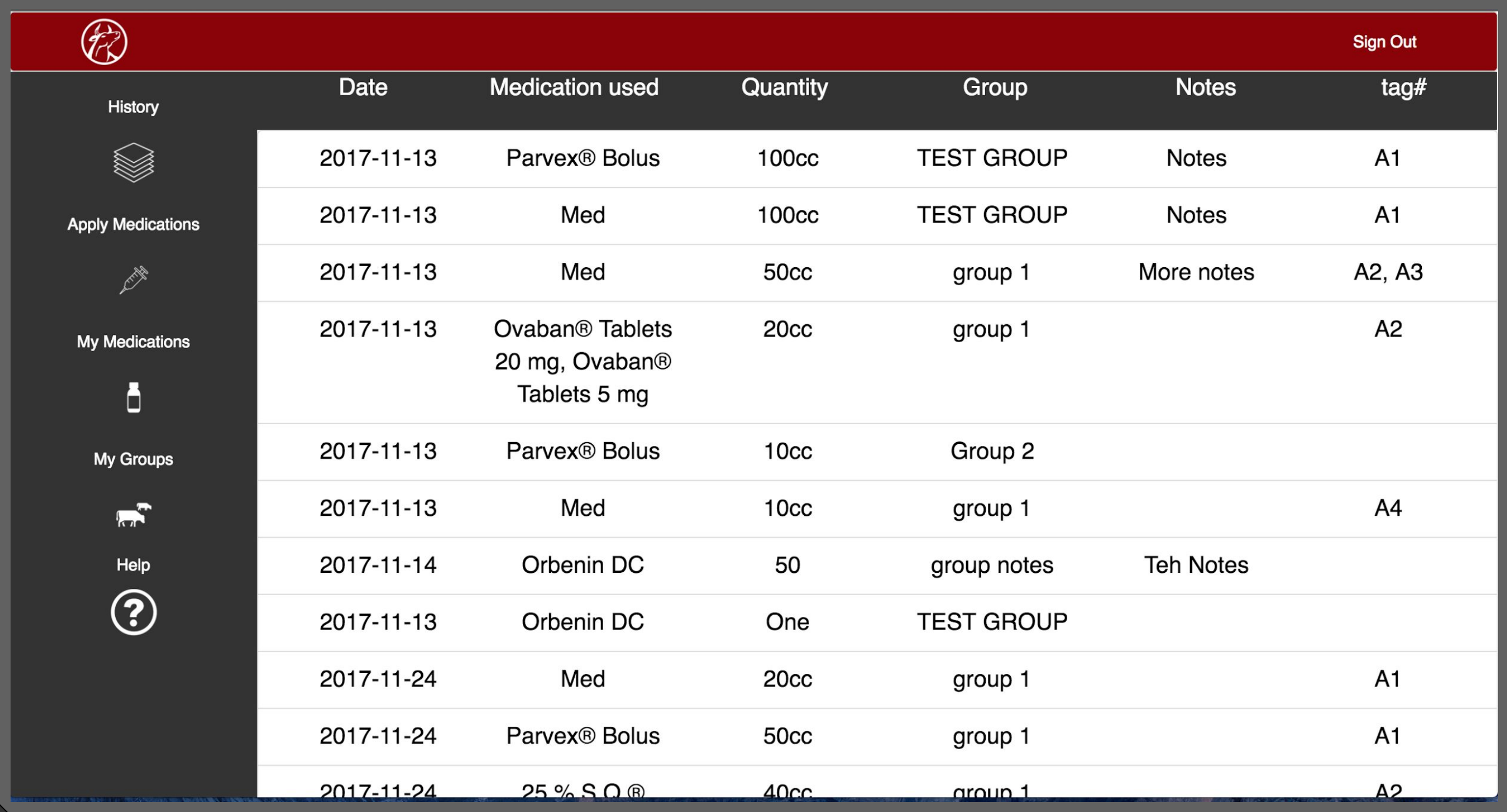
Original Design



Reactive Bootstrap components make data forms more effective, adaptive, and attractive. Bootstrap also makes adaptive styling much easier for mobile consumption on tablets & phones.



Achieved Design

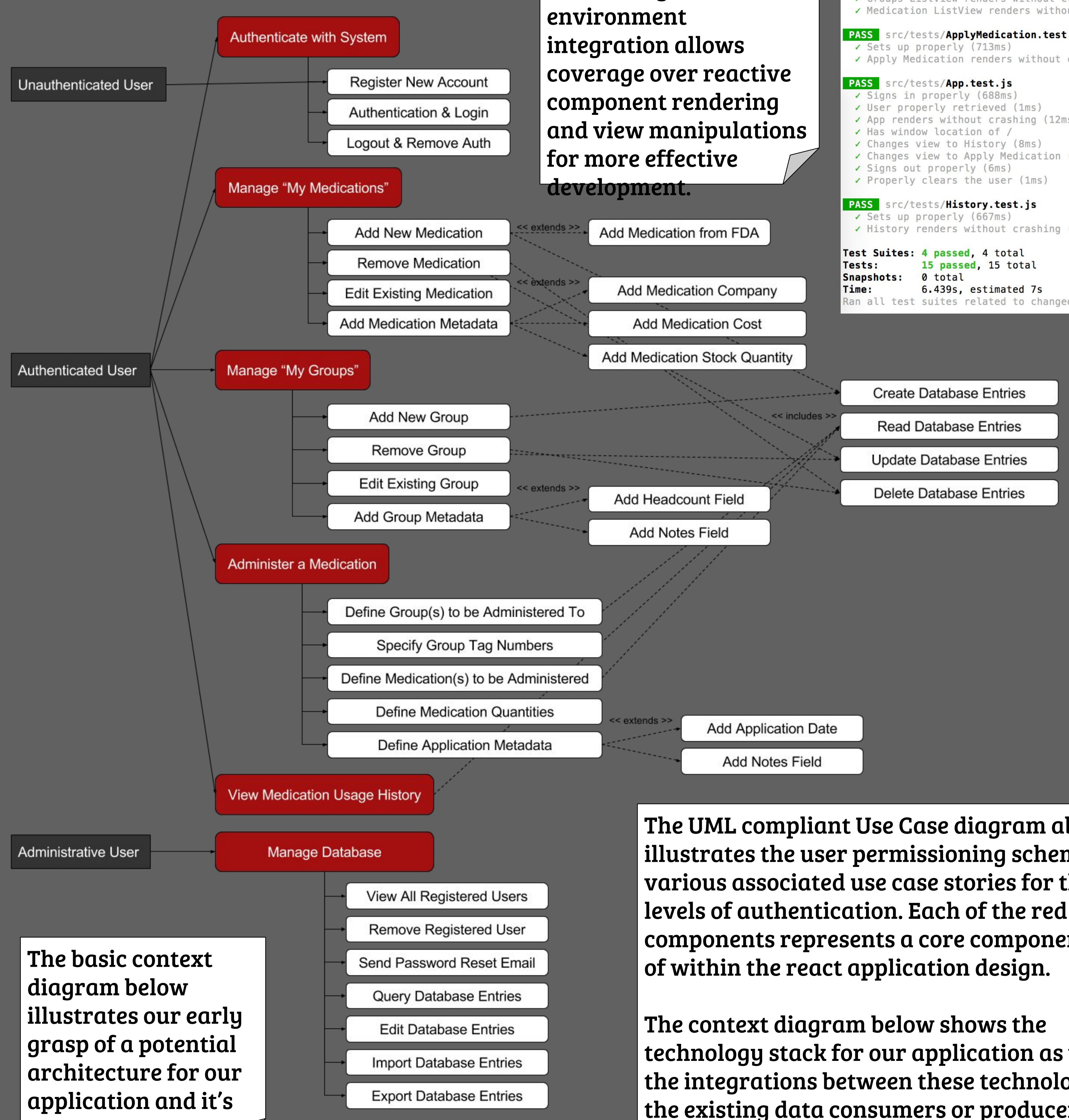


Date	Medication used	Quantity	Group	Notes	tag#
2017-11-13	Parvex® Bolus	100cc	TEST GROUP	Notes	A1
2017-11-13	Med	100cc	TEST GROUP	Notes	A1
2017-11-13	Med	50cc	group 1	More notes	A2, A3
2017-11-13	Ovaban® Tablets 20 mg, Ovaban® Tablets 5 mg	20cc	group 1		A2
2017-11-13	Parvex® Bolus	10cc	Group 2		
2017-11-13	Med	10cc	group 1		A4
2017-11-14	Orbenin DC	50	group notes	Teh Notes	
2017-11-13	Orbenin DC	One	TEST GROUP		
2017-11-24	Med	20cc	group 1		A1
2017-11-24	Parvex® Bolus	50cc	group 1		A1
2017-11-24	25 % S.O.®	40cc	group 1		A2

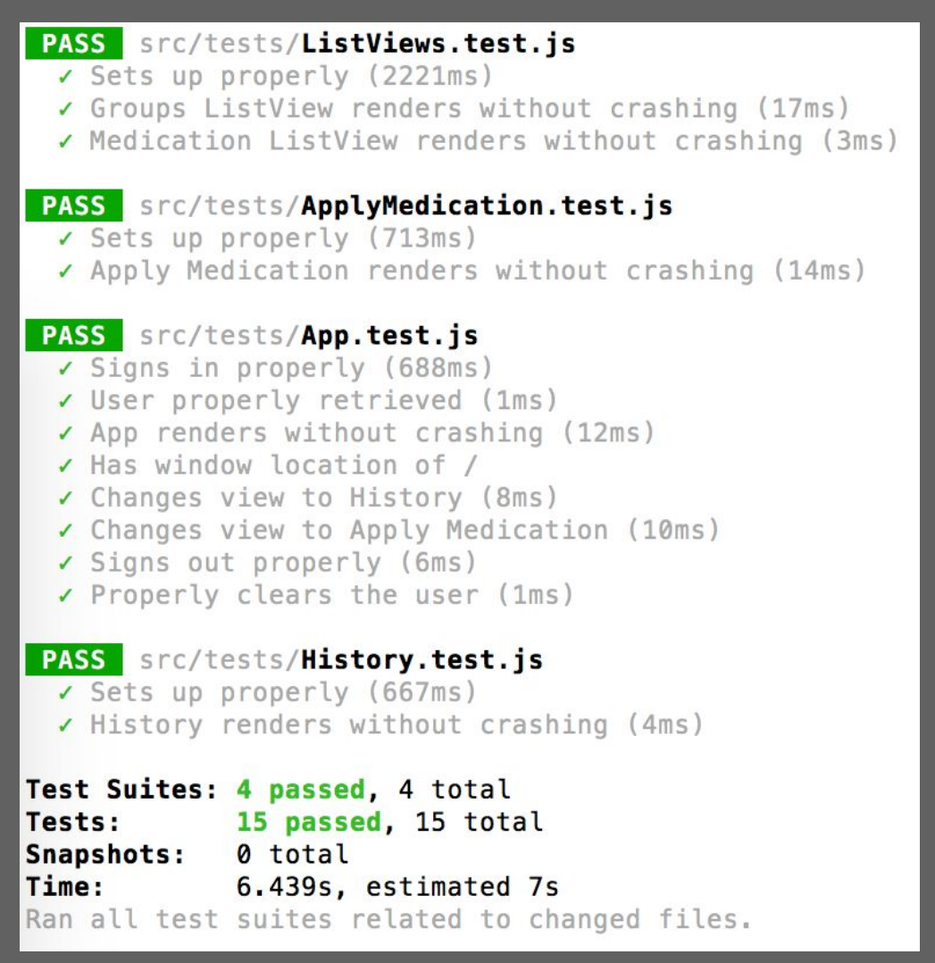
React component design makes modular development much easier and allows assets to be reused throughout the application. This makes for faster development time and a more uniform user interface.

Product Architecture:

UML Use Case Diagram



Jest testing environment integration allows coverage over reactive component rendering and view manipulations for more effective development.



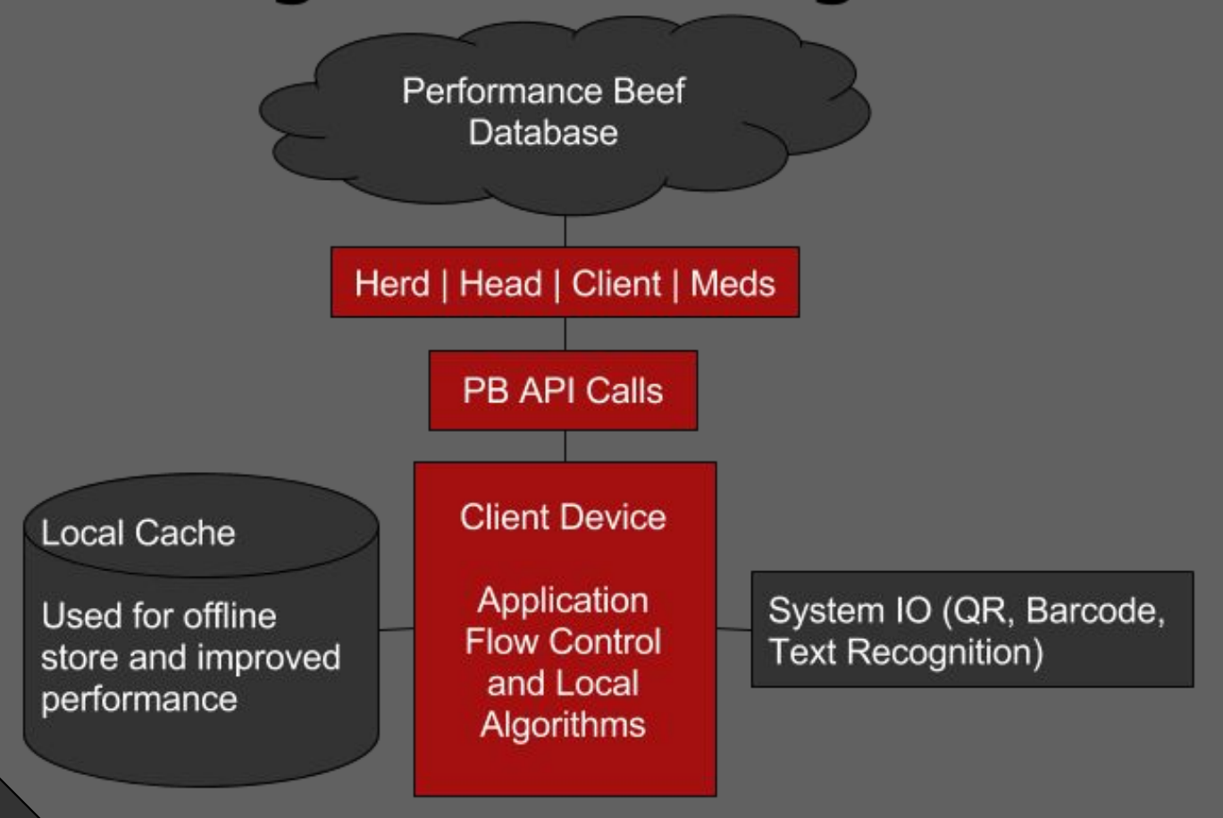
```
PASS  src/tests/ListViews.test.js
  ✓ Sets up properly (2221ms)
  ✓ Groups ListView renders without crashing (17ms)
  ✓ Medication ListView renders without crashing (3ms)

PASS  src/tests/ApplyMedication.test.js
  ✓ Sets up properly (713ms)
  ✓ Apply Medication renders without crashing (14ms)

PASS  src/tests/App.test.js
  ✓ Signs in properly (688ms)
  ✓ User properly retrieved (1ms)
  ✓ App renders without crashing (12ms)
  ✓ Has window location of /
  ✓ Changes view to History (8ms)
  ✓ Changes view to Apply Medication (10ms)
  ✓ Signs out properly (6ms)
  ✓ Properly clears the user (1ms)

PASS  src/tests/History.test.js
  ✓ Sets up properly (667ms)
  ✓ History renders without crashing (4ms)

Test Suites: 4 passed, 4 total
Tests:       15 passed, 15 total
Snapshots:   0 total
Time:        6.439s, estimated 7s
Ran all test suites related to changed files.
```

The UML compliant Use Case diagram above illustrates the user permissioning scheme and various associated use case stories for those levels of authentication. Each of the red parent components represents a core component view of within the react application design.

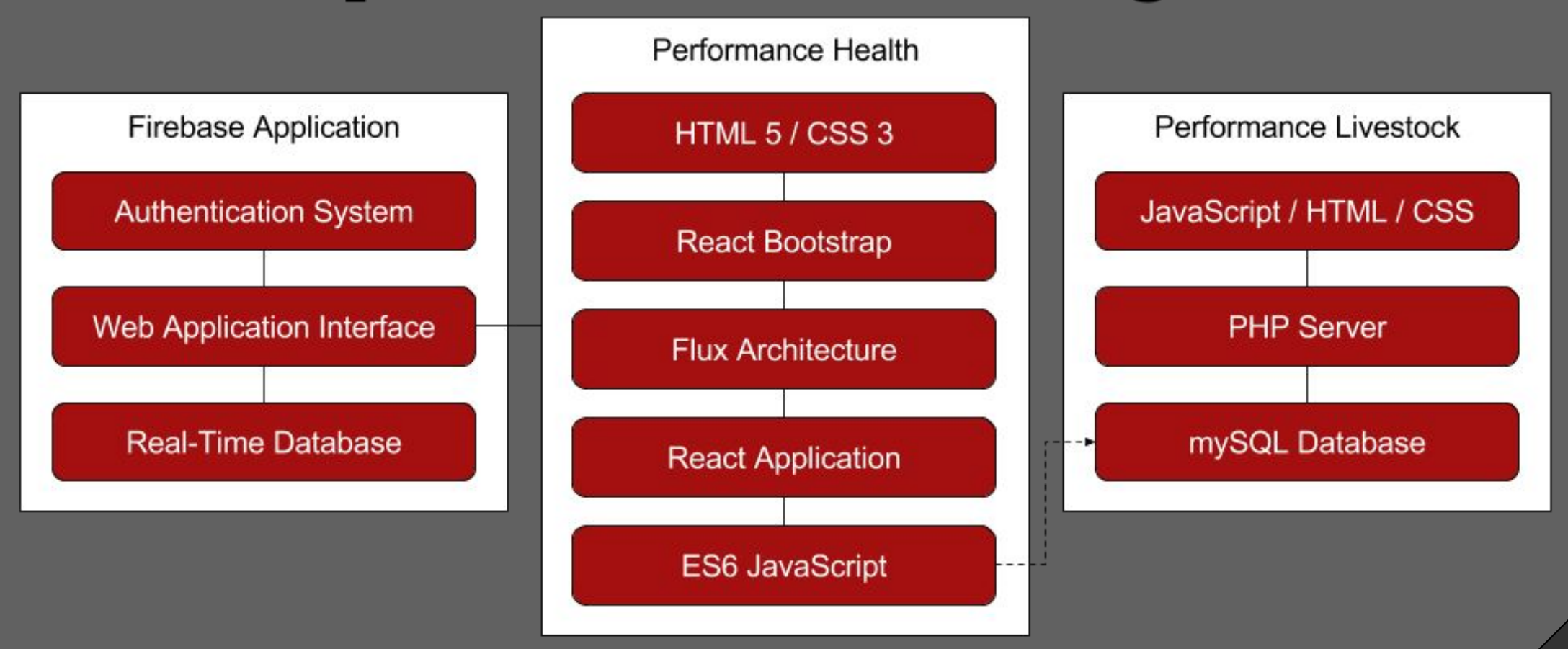
The context diagram below shows the technology stack for our application as well as the integrations between these technologies and the existing data consumers or producers.

The basic context diagram below illustrates our early grasp of a potential architecture for our application and it's eventual evolution.

Early Context Diagram


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

Implemented Context Diagram



Functional Reqs:

A user can:

- Authenticate and register with the system
- Manage their "my medications" list
- Manage their "my groups" list
- Administer a medication to a group or subgroup
- View history of medication usage and associated metadata
- Define metadata for medication applications such as quantity per head, tag selection information, and user specific note strings

Technical Details:

- HTML5, designed for the modern web and all compliant devices
- Javascript, fully ES6 compliant and built upon the latest JS standards means our application can immediately be integrated with countless web technologies
- React, component driven architecture makes project scaling vector much higher with minimized development effort moving forward
- Flux, reactive three-dimensional data synchronization via React's virtual DOM
- Bootstrap, component based user interface design with mobile adaptivity in mind makes for an intuitive, cohesive, and attractive user interface design
- Node.JS + NPM, package management and local server development made easy with the aid of Node, package management, and FaceBook's create-react-app
- Firebase, cloud based data storage with the modern advancements of real-time database synchronization, built-in authentication support, and JSON syntax

Non-Functional Reqs:

- Data must remain integrable when a user enters it manually and be consistent with automatically populated data
- Many users and accounts should be able to interact with database simultaneously
- Many instances of a user account should be able to interact with the system simultaneously with correct interleaving
- Database CRUD operations should never fail unsafely resulting in data loss
- User authentication and database operations should use encryption protocol