Hao Hsu - Developer
April 29, 2017

eHealth App Spec

1. Purpose

This document will go over the functionality of eHealth App. It will go over the requirements, the technology stack, the wireframes, architecture flow, components, and the use cases. These may be subject to change as this is in flux.

2. References

- eHealth Software Design Description - High Level Design of Cloud-Based Medical Management System
- eHealth App Wireframes
- https://github.com/facebook/react-native/blob/master/PATENTS
- http://www.valuecoders.com/blog/technology-and-apps/vue-js-comparison-angular-react/
- https://medium.com/the-vue-point/vue-2-0-is-here-ef1f26acf4b8

3. Audience

The target audience of this document is for developers, testers, project managers.

4. Requirements

The application is to be the client application of the eHealth SDD. The application is to be used for Independent Physicians and their customers. The application will have the following services:

4.1. Phase 1

- User, Admin, Staff and Owner Accounts - simplify process of managing, registering new users, authentication and authorization
- Practice Management - simplify the process of authoring, approving and distributing documentation; and capture attestation. It has a dashboard that describes the states of customer documentation, alerts, and reminders. It also has several wizards and forms to simplify processes.
- Electronic Health Record - It will provide reports and charts/graphs for customer records.
- Monitoring and Logs - Be able to save warnings and errors in application to a host database.

4.2. Phase II

- E-Commerce - Store Fronts for different Independent Physicians and their organizations. It will provide payment processing and invoice.
- Archive - TBD

4.3. Phase III

- Telemedicine (Virtual Patient Visits with Video/AR) - TBD
- Virtual Office - TBD
- Medical Wearable - TBD

5. Technology Stack

Several technologies would be evaluated for front-end work.

5.1. Front-End - HTML5 or JSX, Bootstrap, Typescript/Javascript, Lodash, Bluebird/async, Mobile Native/React Native, Vue.js/React, Webpack/Browserify, Origami/MockupPlus/Balsamiq

5.1.1. HTML5 or JSX

JSX is a Javascript extension syntax that can quote HTML. This is optional in Vuejs as HTML can be used. In React, either JSX or Javascript is supported by Facebook.

5.1.2. Typescript or Javascript

- Optional Static Types - clearer errors, less typos, early error detection
- Newer ECMAScript - supports newer syntax from Javascript organization
- Readable Compiled/Transpiled Javascript unlike Babel
- Can run Javascript code as is unlike CoffeeScript
- Less error when sharing data structure and code
- Enhanced IDE support like Intellisense
- Strict null checks - easier to debug
- Require more setup …etc (negative)

5.1.3. Lodash

- lambda transformations like map
- chaining transformations
- utility functions

5.1.4. Mobile Native or React Native

- Lawyers needed to look at the Facebook patent in React
- Natural, monetary motive to support Java ecosystem in Android and Swift in iOS
- Delay in React Native support comparing to Java or Swift when problem occurs
- Many dependencies in React Native compare to Swift and Java
- Does not support older Android versions (>= Android 4.1 (API 16) and >= iOS 8.0)

5.1.5. Vuejs or React

- Lawyers needed to look at the Facebook patent in React
- Vuejs also has Virtual DOM and is faster
- Reusable HTML and CSS with JSX Optional in Vuejs
- Easier to write reusable components and less code to write

- Less opinionated
- Intuitive structure with template, script, styling

- Not the most popular and less components support (negative)
- No mature mobile native support (negative)
- Current team has two people who know React (negative)

5.1.6. Webpack/Browserify
Webpack probably would be used.

5.1.7. Origami/MockupPlus/Balsamiq
Need one of these tools to generate modified jpg in spec. This can be used to generate interactive mockup in HTML/CSS.

5.2. Gulp/Grunt/SCONS/Make
One of these tools to do dependency checks and create dist targets.

5.3. Host server and host db
This is tightly coupled with the FrontEnd and Cloud to access host specific alarms, static data like documentation, and host specific customization

5.4. Loadable constants in JSON manifests for HTML/CSS in components - PostgresSQL/MySQL with JSON type for storing customer specific JSON manifests. These can be reused across web and mobile.

5.5. Our Front-end application will interact with Redis/Mongoose/Elastic Search for high availability. Clustered PostgresSQL/MySQL as the single source of truth will ensure consistency and reliability.

6. Deployment of Application on Cloud Hosting

Our application would create dist files that would be pushed to a hosting site like **AWS** and **Heroku**. For certain static data like documentation, they can be accessed on host database.

7. Wireframes

The dimensions of these wireframes should be responsive. We should look at mock plus, balsimiq and Origami to create interactive wireframes and export to HTML and CSS.

8. Front-end Architecture of Web App

Main HTML element is bound to a Vue instance. Each reusable Vue component is bound to a HTML element. A component file would load a generated HTML file and CSS file. (See Components section). Mockup tool generates <component>_mockup.html and <component>_mockup.css. With the mockup files and a specific JSON manifest (<component>_<version>.json), a variant <component>_template.html can be generated.
Each page has an associated route URL. Each route URL is composed of multiple components. Each component is bound to a specific element id.
Data is to pass between components using props.

9. Components

Components can be defined and reused. A component has 3 parts: template (HTML), styling (CSS), and script (JS). In template section of a component, the template file (ex. login.html) can be loaded in a component .vue (login.vue), and the template file (login_template.html) can be generated from a mockup HTML file (login_mockup.html) and a JSON manifest (login_red.json) of a component.

9.1. Component login.vue

Developer should write the three sections. The template and styling sections can be written by loading some files.

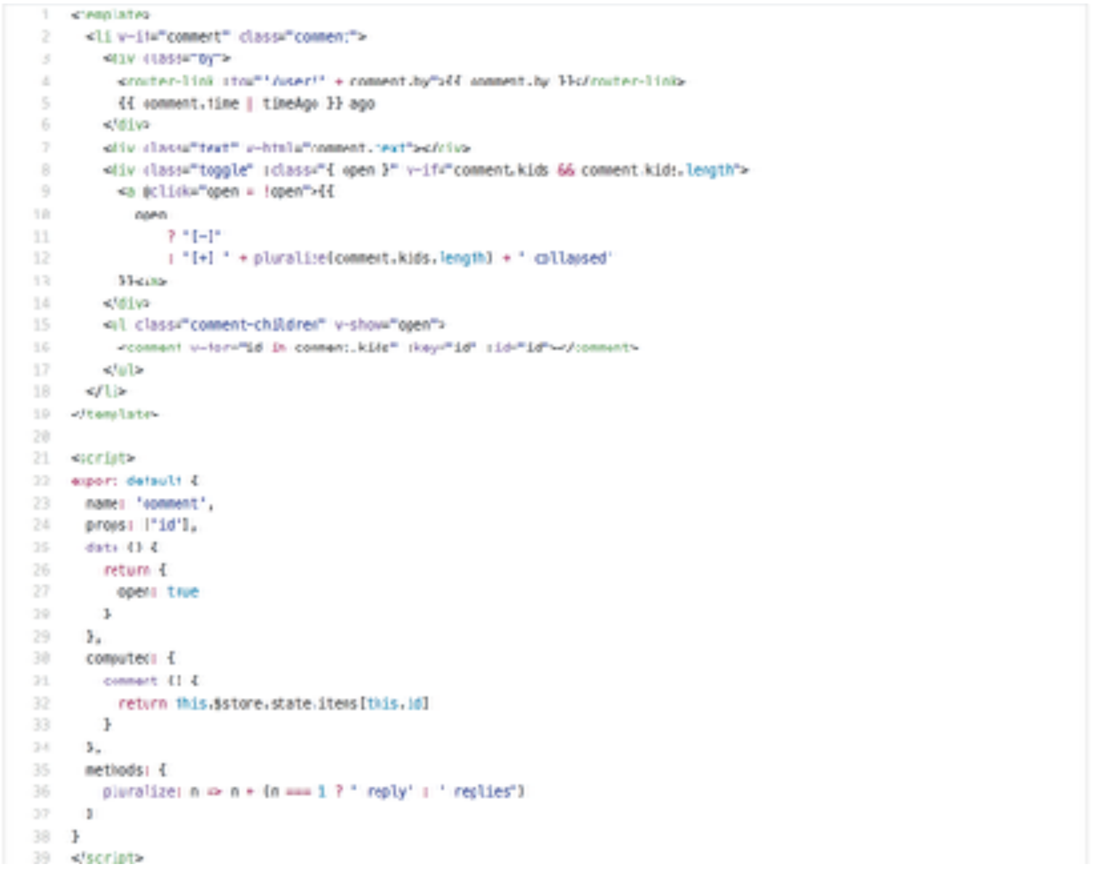
- Template section - load login.html which is generated from login_mockup.html and login.json
- Styling section - load login.css which is generated from login_mockup.html? and login.json
- Script section - javascript section

9.1.1. Mockup template - login_mockup.html - generated from tool

```
<form id=“login_form”>
  <div class="imgcontainer">
    <img src="img_avatar2.png" alt="Avatar" class="avatar">
  </div>

  <div class="container">
    <label><b>Username</b></label>
    <input id=“username” type=“text" placeholder="Enter Username or
Email" name="uname" required>

    <label><b>Password</b></label>
    <input id=“password” type=“password" placeholder="Enter Password"
name="psw" required>

    <button id=“loginButton” type=“submit">Login</button>
    <input type="checkbox" checked="checked"> Remember me
  </div>

 <div class="container" style="background-color:#f1f1f1">


    <button type="button" class="cancelbtn">Cancel</button>

    <span class="psw">Forgot <a href="#">password?</a></span>
```

```
        </div>
      </form>
```

9.2. Component Example



```
<template>
  <li v-if="comment" class="comment">
    <div class="by">
      <router-link :to="'/user/' + comment.by">{{ comment.by }}</router-link>
      {{ comment.time | timeAgo }} ago
    </div>
    <div class="text" v-html="comment.text"></div>
    <div class="toggle" :class="{ open }" v-if="comment.kids && comment.kids.length">
      <a @click="open = !open">{{
        open
            ? '[-]'
            : '[+] ' + pluralize(comment.kids.length) + ' collapsed'
      }}</a>
    </div>
    <ul class="comment-children" v-show="open">
      <comment v-for="id in comment.kids" :key="id" :id="id"></comment>
    </ul>
  </li>
</template>

<script>
export default {
  name: 'comment',
  props: ['id'],
  data () {
    return {
      open: true
    }
  },
  computed: {
    comment () {
      return this.$store.state.items[this.id]
    }
  },
  methods: {
    pluralize: n => n + (n === 1 ? ' reply' : ' replies')
  }
}
</script>
```

10. JSON manifest for Components

JSON manifest files would have the constants/overrides that would be used in the template (HTML) and styling (CSS) sections of Vuejs components. These files can be reused to replace data in the <component>_mockup.html.

In order for this work, the following requirements should be met:

- the mockup tool generates an id for each HTML node in the document tree or a script needs to be created to do this
- store specific text, colors, icons, images that distinguishes the look and meaning of the elements in the UI
- login_mockup.html + login_red.json -> login_template.html to be included in login.vue

10.1. Login json manifest - login.vue, login_red_template.html, login_mockup.html, login_red.json

```
{ “size”: 4,
“elements”: [ { “id” :  “username”,
            “element”: “input”,
            “placeHolder” : “Enter Username or Email”,
            “color”: “white”,
            “background”: “red”
           },
           { “id” : “usericon”,
              “element”: “i”,
              “icon”: “glyphicon glyphicon-user”
           },
           { “id”: “password”,
              “element”: “input”,
              “placeHolder” : “Enter Username or Email”,
              “color”: “white”,
              “background”: “red”
            },
           { “id”: “loginButton”,
             “element”: “button”,
             “color”: “white”,
             “background”: “red” } ] }
```

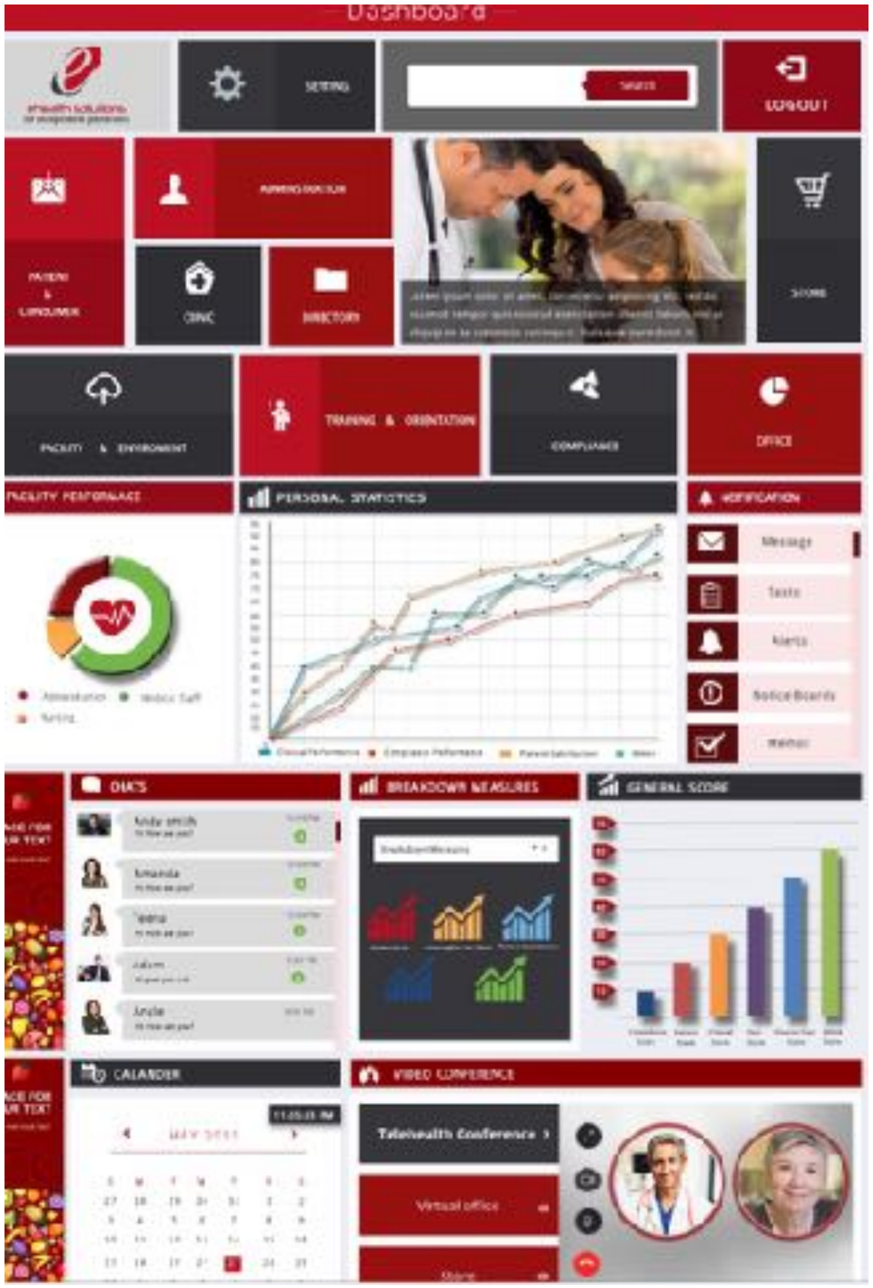
11. URL Flows with vue-routing

/{category}/{page}

11.1./dashboard

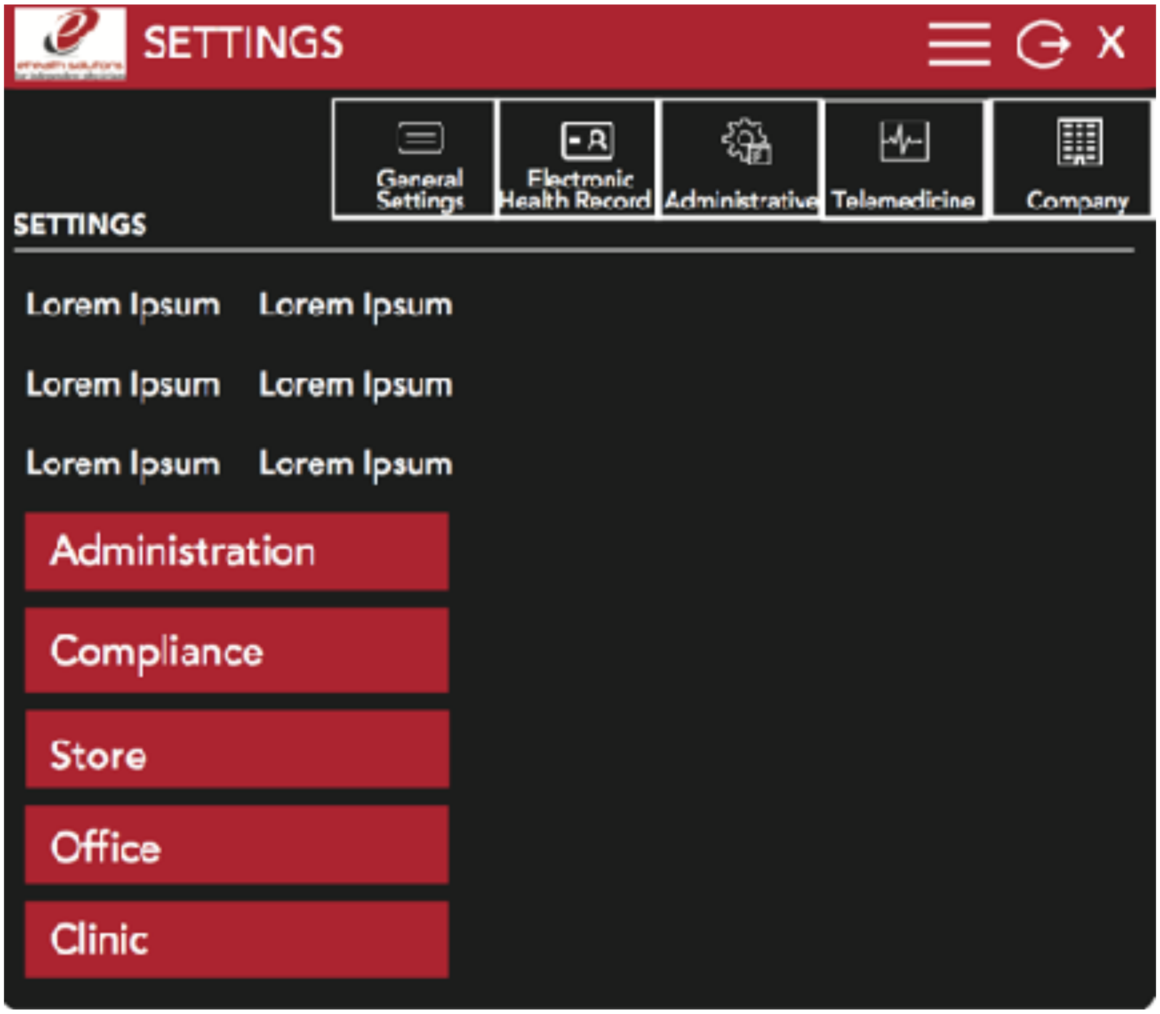
The layout of the main dashboard is not for web app but for mobile??

- panels in the dashboard - stats, notification, chats, score, measures
- url links associated to this page - /logout, /setting, /search, /training, /company /office, /directory



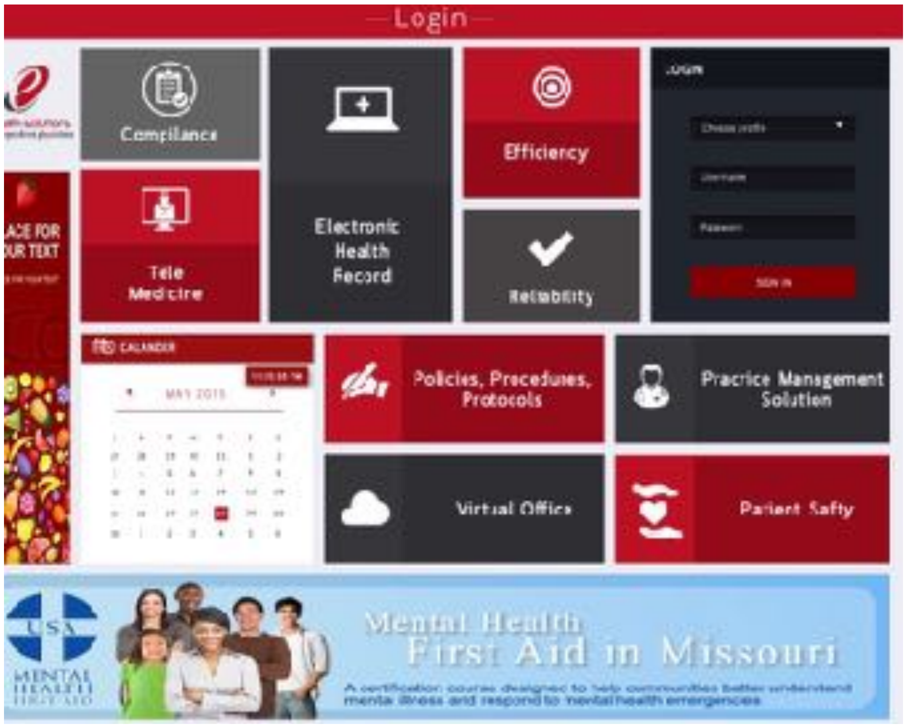
11.2./settings

- bar with links - /ehr, /admin, disabled /telemedicine, /company - this may be redundant
- Menu with links - /admin, /compliance, /store, /office, /clinic



11.3./admin
11.4./directory
11.5./company
11.6./logout
11.7./login

This seems to do too much. We only need the login screen and some public links. Users cannot access confidential data without logging in so most of these links should not be here.


Login
Compliance
Electronic Health Record
Efficiency
Reliability
Tele Medicine
CALANDER
MAY 2015
Policies, Procedures, Protocols
Practice Management Solution
Virtual Office
Patient Safty
Sign in
Mental Health First Aid in Missouri
A certification course designed to help communities better understand mental illness and respond to mental health emergencies
MENTAL HEALTH FIRST AID

11.8./practice-management
11.9./ehs
11.10./training

12. States with vuex or redux

Vuejs can use Redux or Vuex to manage its states.

13. Use Cases

13.1. Patient makes an appointment (wireframes - tbd)
13.2. Register new user (wireframes - tbd)
13.3. User login
13.4. Roles and Authorization
13.5. Operation with no Network?
13.6. Approve document
13.7. Patching