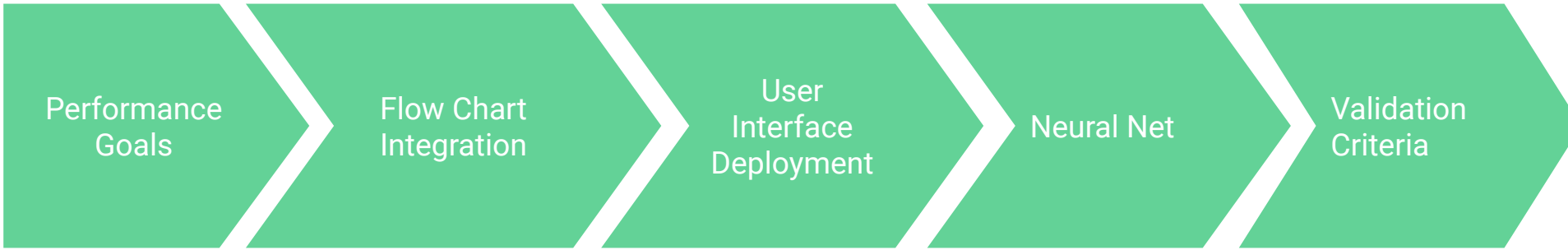
Burn Resuscitation and Management for Early Responders

BMExtra Group:
Jacob Ayers (BME), Hannah Kang (BME), Dominique Szymkiewicz (BME), Nora Ward (BME), Thomas Yates (BME), Eric Yeats (CompE)

Contact:
Avinash Kumar M.D.

Presentation Overview



Validation Stage Performance Goals

- Camera application/Neural Network integrated into app
  - Easy to use by end user
  - Successfully take photo and calculate TBSA from photo
- Patient information capture - recorded within App
  - Age, gender, weight
  - Burn type
- Neural Net Analysis
  - Image analysis
  - Burn Degree
- Required Qualifications
  - TBSA
  - Burn Degree
  - Transfer Recommendation
  - Fluid Resuscitation Recommendation

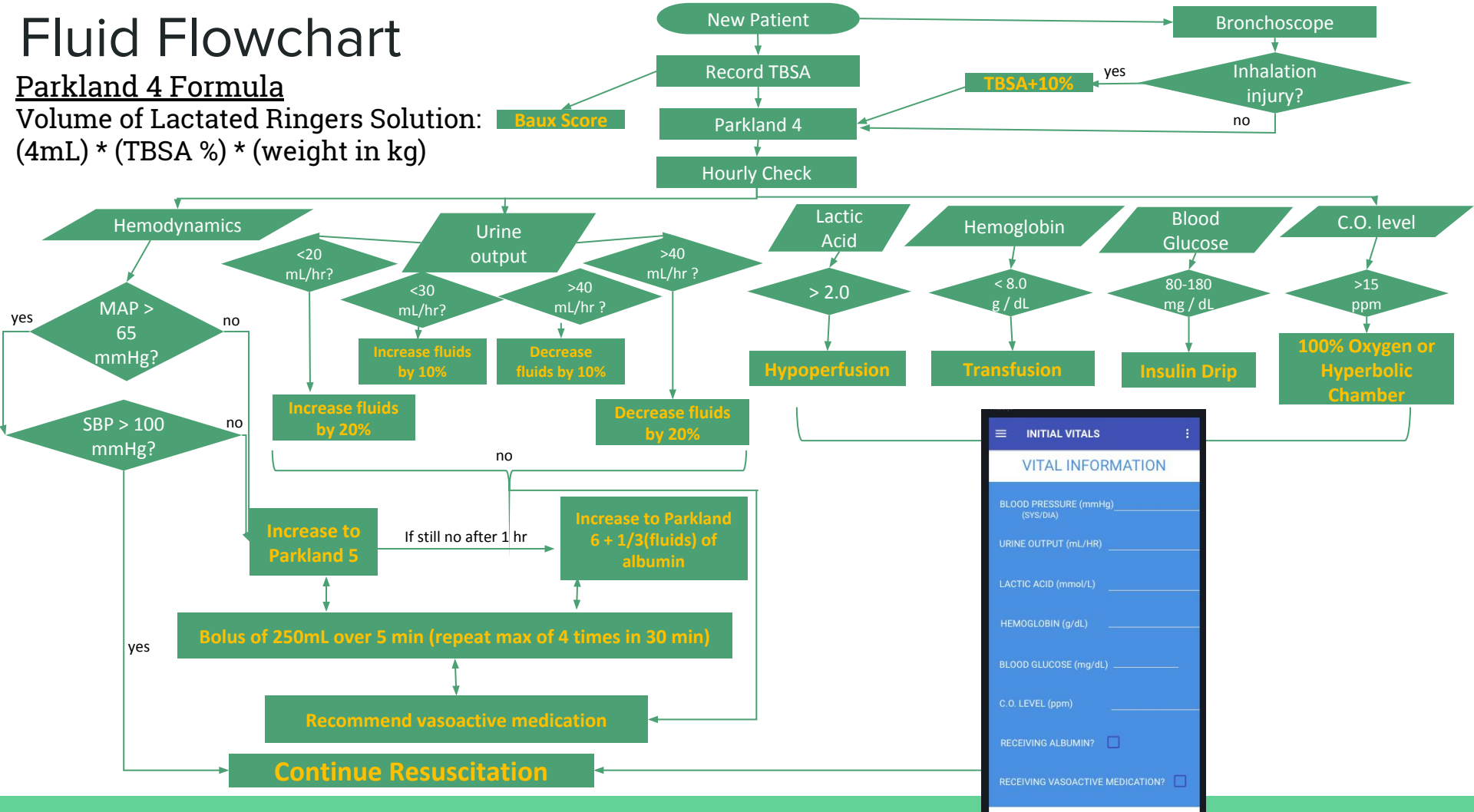
Important Information Capture

Information Needed	**User Interface Slide Location**
Patient Personal Information (name, age, gender, weight, initial conditions/existing medical considerations)	New Patient
Initial Measurements (Fluid administered before trauma center check in)	Initial Inputs
Initial Measurements (patient baseline vitals before administration of additional fluids)	Initial Vitals
Neural Net Photo Diagnostic Output (TBSA calculation, burn degree, initial fluid administration recommendation)	Solution
Old Vitals Information	Vital History - specifically 1-2 hours previously found in Vital History for X slide
Fluid Administration Log	Fluid History
Past/Initial Photos Viewed at a Later Time	Past Pictures
Vitals Update (capturing patient vitals and hour that information is collected)	Update Vitals

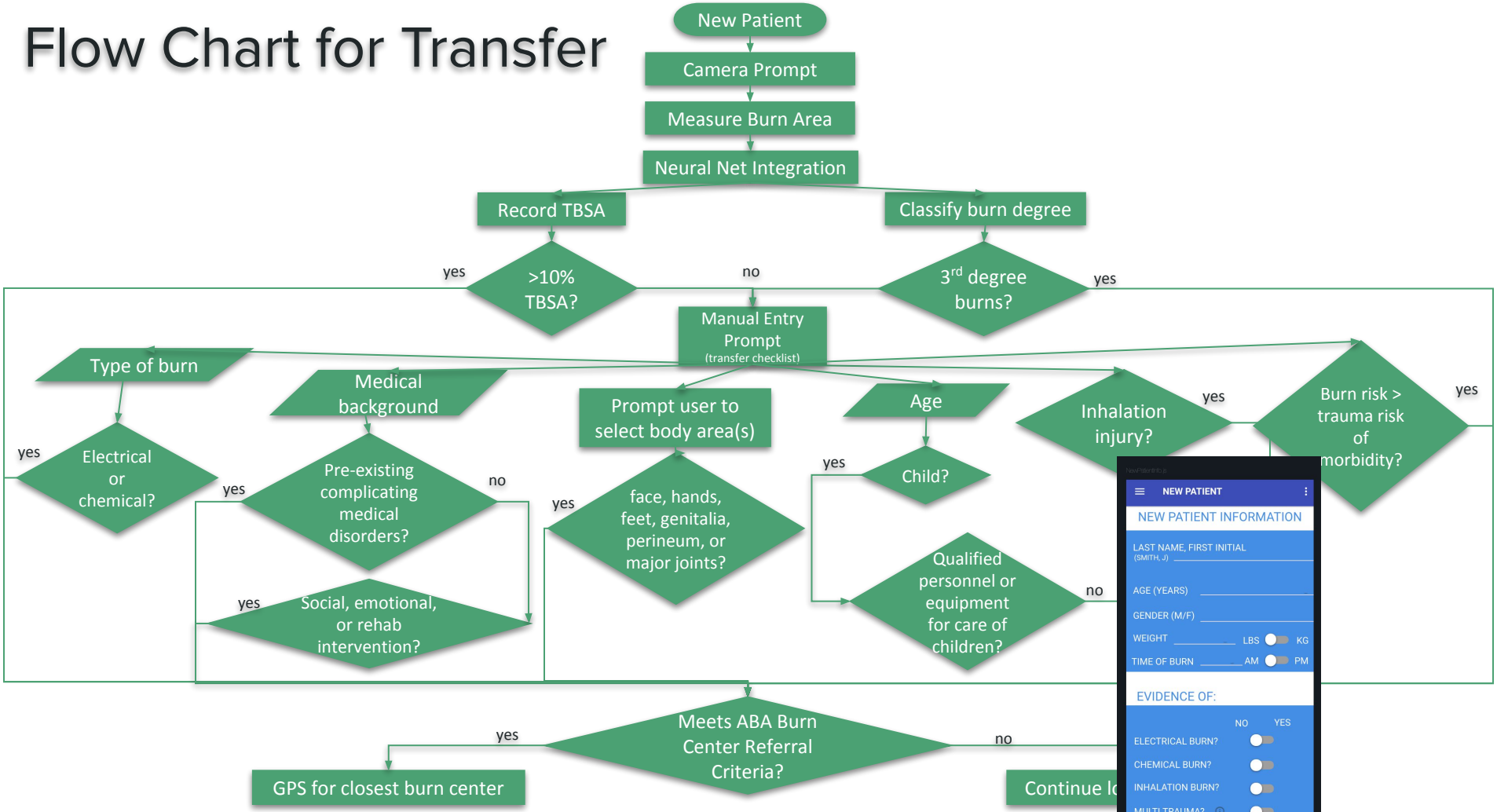
Fluid Flowchart

Parkland 4 Formula

Volume of Lactated Ringers Solution:
(4mL) * (TBSA %) * (weight in kg)



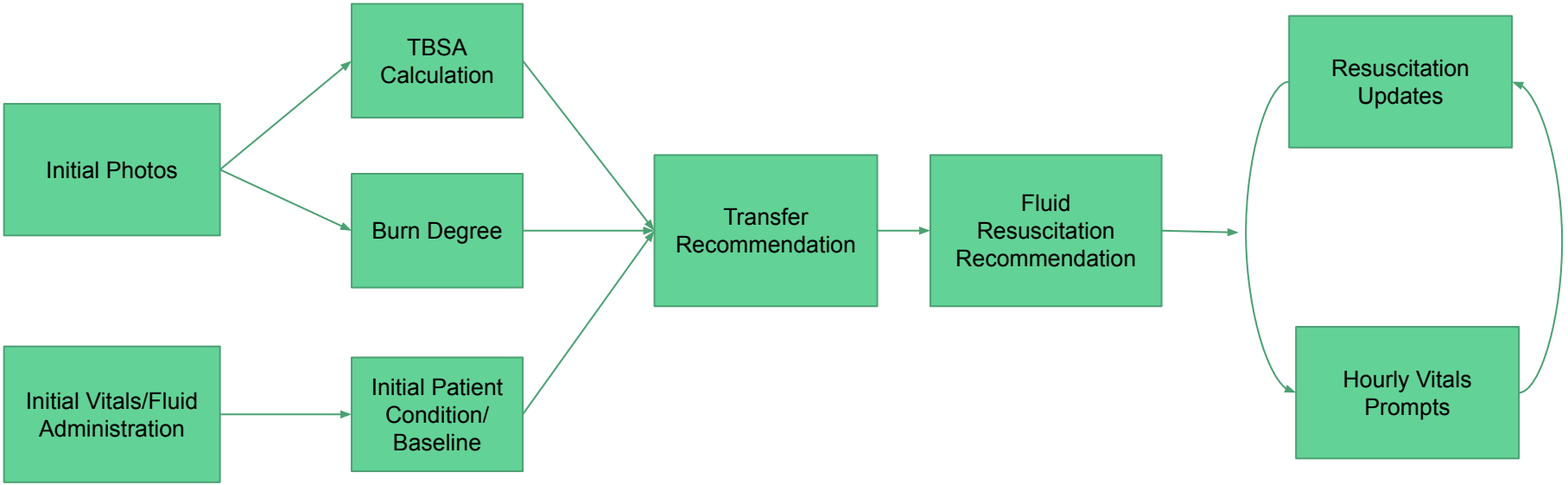
Flow Chart for Transfer

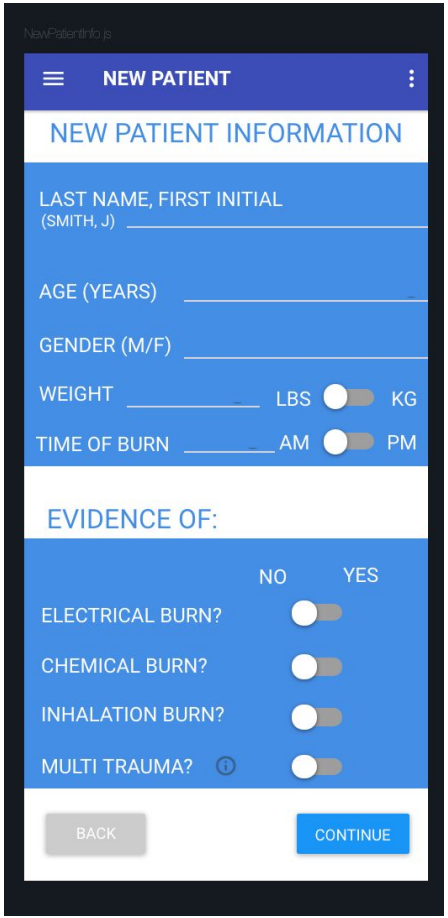


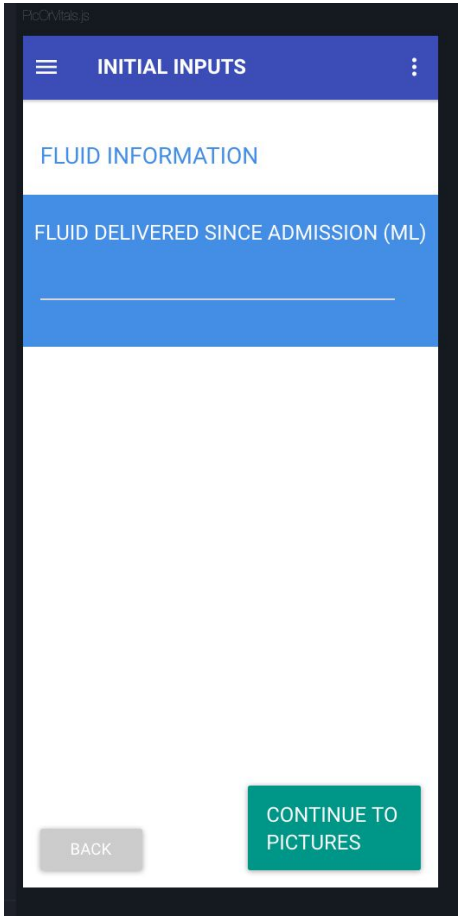
User Interface - Beginning Prototype

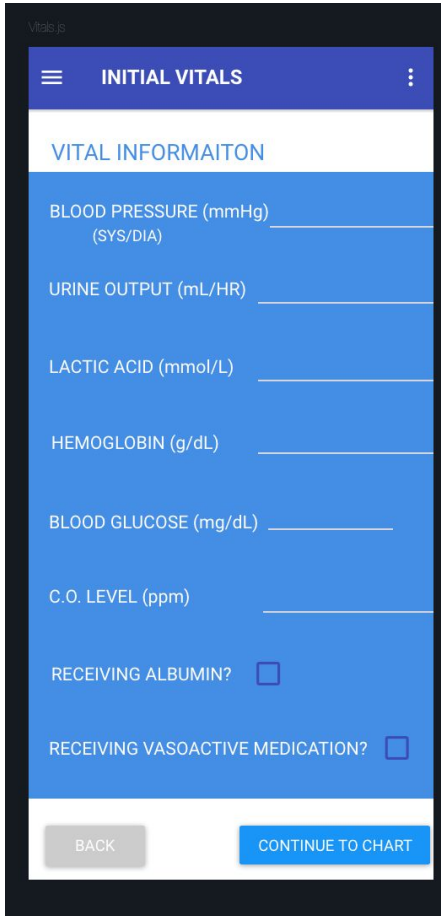
User Interface - Basic Design Ideas

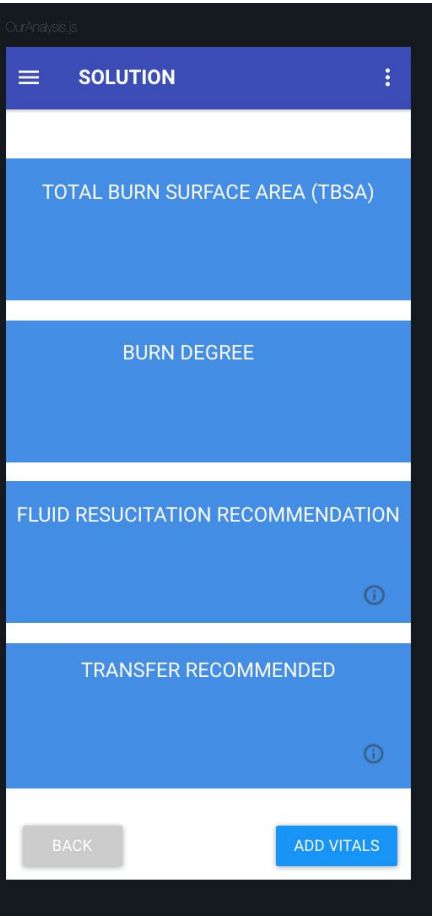
User Interface - Basic Design Ideas

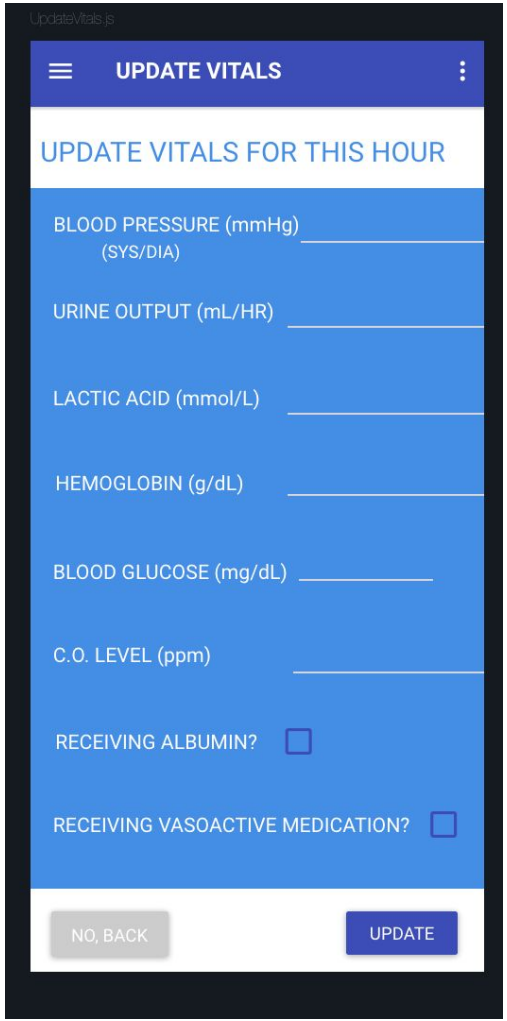


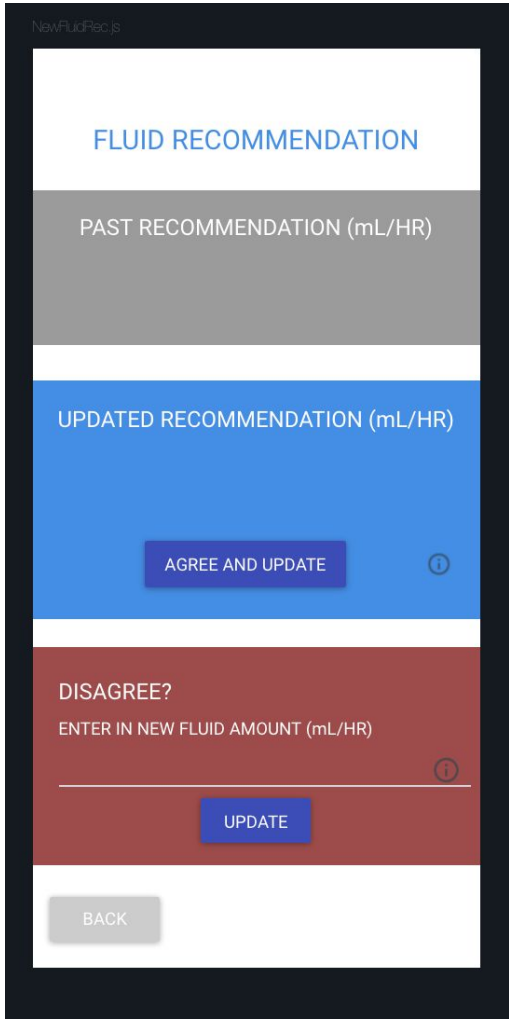

NewPatientInfo.js
NEW PATIENT
NEW PATIENT INFORMATION
LAST NAME, FIRST INITIAL
(SMITH, J)
AGE (YEARS)
GENDER (M/F)
WEIGHT
LBS
KG
TIME OF BURN
AM
PM
EVIDENCE OF:
NO
YES
ELECTRICAL BURN?
CHEMICAL BURN?
INHALATION BURN?
MULTI TRAUMA?
BACK
CONTINUE


FluidVitals.js
INITIAL INPUTS
FLUID INFORMATION
FLUID DELIVERED SINCE ADMISSION (ML)
BACK
CONTINUE TO PICTURES


Vitals.js
INITIAL VITALS
VITAL INFORMAITON
BLOOD PRESSURE (mmHg)
(SYS/DIA)
URINE OUTPUT (mL/HR)
LACTIC ACID (mmol/L)
HEMOGLOBIN (g/dL)
BLOOD GLUCOSE (mg/dL)
C.O. LEVEL (ppm)
RECEIVING ALBUMIN?
RECEIVING VASOACTIVE MEDICATION?
BACK
CONTINUE TO CHART


OurAnalysis.js
SOLUTION
TOTAL BURN SURFACE AREA (TBSA)
BURN DEGREE
FLUID RESUCITATION RECOMMENDATION
TRANSFER RECOMMENDED
BACK
ADD VITALS


UpdateVitals.js
UPDATE VITALS
UPDATE VITALS FOR THIS HOUR
BLOOD PRESSURE (mmHg)
(SYS/DIA)
URINE OUTPUT (mL/HR)
LACTIC ACID (mmol/L)
HEMOGLOBIN (g/dL)
BLOOD GLUCOSE (mg/dL)
C.O. LEVEL (ppm)
RECEIVING ALBUMIN?
RECEIVING VASOACTIVE MEDICATION?
NO, BACK
UPDATE


NewFluidRec.js
FLUID RECOMMENDATION
PAST RECOMMENDATION (mL/HR)
UPDATED RECOMMENDATION (mL/HR)
AGREE AND UPDATE
DISAGREE?
ENTER IN NEW FLUID AMOUNT (mL/HR)
UPDATE
BACK

```
import React, { Component } from "react";
import Button117 from "../symbols/button117";
import { Center } from "@builderx/utils";
import Button612 from "../symbols/button612";
import { View, StyleSheet, Text } from "react-native";

export default class EndRes extends Component {
  render() {
    return (
      <View style={styles.root}>
        <Center vertical>
          <View style={styles.rect} />
        </Center>
        <Center horizontal>
          <Button117 style={styles.button117} />
```

```
export default class SecurityAuth extends Component {
  render() {
    return (
      <View style={styles.root}>
        <Image
          source={require("../assets/a3d677392ee343199bc8e0bfbba7037f_(1).jpeg")}
          style={styles.logo}
        />
        <View style={{...styles.gray, flex: 0.6, flexDirection: 'column', justifyContent: 'flex-start', alignItems: 'stretch'}} >
          <View style={{flex: 1, flexDirection: 'row', alignItems: 'center'}}>
            <Text style={styles.text}>USERNAME</Text>
            <DisabledTextbox style={styles.DisabledTextbox} />
          </View>
          <View style={{flex: 1, flexDirection: 'row', alignItems: 'center'}}>
            <Text style={styles.text}>PASSWORD</Text>
            <DisabledTextbox style={styles.DisabledTextbox} />
          </View>
          <View style={{flex: 1, flexDirection: 'row', justifyContent: 'center', alignItems: 'center'}}>
            <Button101
              style={{...styles.button}}
              root={() => {
                this.props.navigation.push("Patient");
              }}
              onPress={() => {
                this.props.navigation.push("Patient");
              }}
            />
          </View>
        </View>
```

TBSA Algorithm and ConvNet on App

```java
@ReactMethod
public void calcTBSA(String imgb64, Callback errorCallback, Callback successCallback) {
    try {
        // convert Base64 to single-precision floating point
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inDither = true;
        options.inPreferredConfig = Bitmap.Config.ARGB_8888;

        byte[] decodedString = Base64.decode(imgb64, Base64.DEFAULT);
        Bitmap image = BitmapFactory.decodeByteArray(decodedString, offset: 0, decodedString.length);

        int[] results = {0, 0, 0};
        // cut up image into 50x50 images, should already be BGR
        // feed each image to the neural network
        for (int row = 0; row < image.getHeight() - INPUT_SIZE; row+=INPUT_SIZE) {
            for (int col = 0; col < image.getWidth() - INPUT_SIZE; col += INPUT_SIZE) {
                // slice image and get bytebuffer
                Bitmap subimage = Bitmap.createBitmap(image, col, row, INPUT_SIZE, INPUT_SIZE);
                ByteBuffer byteBuffer = convertBitmapToByteBuffer(subimage);
                float[][] result = new float[1][labels.size()];
                tflite.run(byteBuffer, result);
                int ind = argMax(result);
                results[ind] += 1;
            }
        }
        successCallback.invoke( ...args: Double.toString( d: (double)results[0]/(results[0]+results[1])));
    } catch (Exception e) {
        errorCallback.invoke(e.getMessage());
    }
}
```

Neural Network Model

- Convolutional Neural Network
  - 50x50x3 Images ➜ Softmax Output

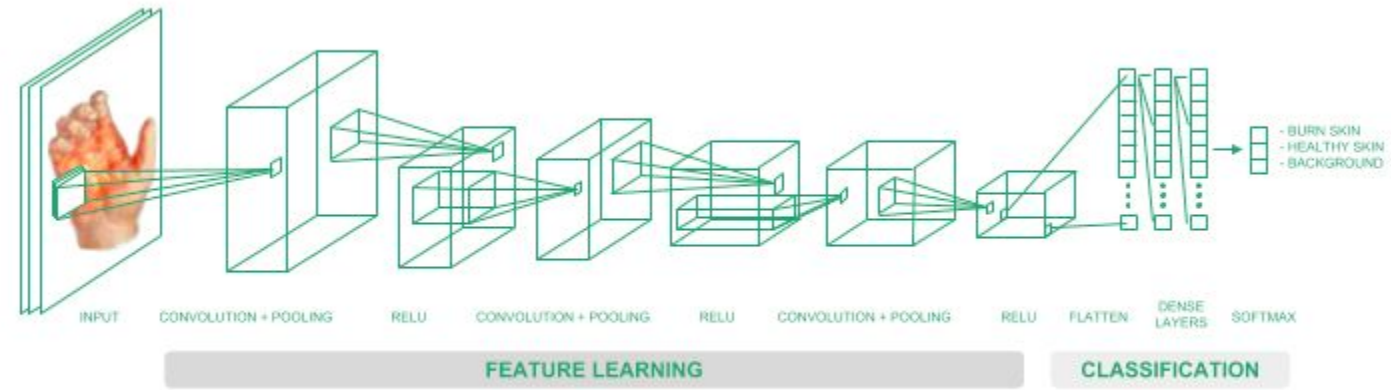


Image Classification Update

- Dr Kumar provided an additional 100+ images for classification, training
  - Burned Count: 12,758
  - Healthy Count: 14,097
  - Background Count: 28020
- Total: 54,875

Neural Net Update

- Neural Net Trainer
  - Through 10 Epochs with 54,875 subimages:
    - Maximum Accuracy = ~85%
    - Minimum Loss = ~.447
  - Features = 100/Conv Layer
  - Saved into .pb file for mobile version
- Neural Net Predictor
  - Input any image
  - Image is split into 50x50 subimages and classified
  - Reconstructed and colored based on classification
  - Control BSA Error: 7%-17%
    - This error translates to .3%-3% error in TBSA

Validation Metrics

- Compare our app to current gold standard
  - Sensitivity (Burn Images)
  - Specificity (Background and Healthy Skin)
  - Similarity Metric
    - Compare the “masks” created by code and doctors
- Determine effect on patient care
  - Average time
  - Compare fluid recommendations
  - Quality of Care

Capturing Validation Metrics

- Excel spreadsheet
  - Widely used within the Burn ICU to calculate TBSA
  - Easily formatted to record/analyze data
- Provider Survey
  - Feedback on integration/ease of use
  - Look for potential errors
  - Qualitatively determine effect of this app

Next Steps

- Continued Work on App Deployment
  - Integration of subparts
  - Performance testing
- Validation/Testing
  - Creation of capture forms
  - Discussion on validation timeline