|  |
| --- |
| Jan Fontanosa, Vyacheslav Perepelytsya, and Maasha Maheson |
| HVAC |
| Internet of Things Capstone Project |

|  |
| --- |
| Humber College  4-23-2018 |

# Declaration of Joint Authorship

We, Jan Fontanosa, Vyacheslav Perepelytsya, and Maasha Maheson, confirm that this work submitted for assessment is our own and is expressed in our own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. A list of the references used is included.

# Proposal

05

***Proposal for the development of HVAC***

Prepared by   
*Computer Engineering Technology Students*

**Executive Summary**

As a student in the Computer Engineering Technology program, I will be integrating the knowledge and skills I have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with the following sensors and actuators: . The database will store . The mobile device functionality will include the and will be further detailed in the mobile application proposal. I will be collaborating with the following company/department Humber Greenhouse. In the winter semester I plan to form a group with the following students, who are also building similar hardware this term and working on the mobile application with me . The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software Project. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project as a member of a 2 or 3 student group.

**Background**

The problem solved by this project is . A bit of background about this topic is .

Existing products on the market include [1]. I have searched for prior art via Humber’s IEEE subscription selecting “My Subscribed Content”[2] and have found and read [3] which provides insight into similar efforts.

In the Computer Engineering Technology program we have learned about the following topics from the respective relevant courses:

* Java Docs from CENG 212 Programming Techniques In Java,
* Construction of circuits from CENG 215 Digital And Interfacing Systems,
* Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
* Micro computing from CENG 252 Embedded Systems,
* SQL from CENG 254 Database With Java,
* Web access of databases from CENG 256 Internet Scripting; and,
* Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable me to build the subsystems and integrate them together as my capstone project.

**Methodology**

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the fall semester. My coursework will focus on the first two of the 3 phases of this project:  
 Phase 1 Hardware build.  
 Phase 2 System integration.  
 Phase 3 Demonstration to future employers.

*Phase 1 Hardware build*

The hardware build will be completed in the fall term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

*Phase 2 System integration*

The system integration will be completed in the fall term.

*Phase 3 Demonstration to future employers*

This project will showcase the knowledge and skills that I have learned to potential employers.

The brief description below provides rough effort and non-labour estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

**Concluding remarks**

This proposal presents a plan for providing an IoT solution for . This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects such as the initiative described by [3]. I request approval of this project.

# Abstract

This technical report is about the build progress and other documentation about our Internet of Things (IoT)-based Heating, Ventilation, and Air Conditioning (HVAC) monitoring system. This HVAC monitoring system controls the temperature, moisture, and sound generated from the system depending on the user’s specifications. The user can optimise the system in regards to their needs. The system is eco-friendly and optimizes energy and financial savings in a residential, commercial or industrial environment. These smart HVAC monitoring systems increase the efficiency of the working and production environment.

Table of Contents

[Declaration of Joint Authorship i](#_Toc508628689)

[Proposal iii](#_Toc508628690)

[Abstract vii](#_Toc508628691)

[List of Figures xi](#_Toc508628692)

[1. Introduction 1](#_Toc508628693)

[2. Project Description 3](#_Toc508628694)

[2.1 Project Requirements Specification 3](#_Toc508628695)

[2.1.1 Purpose 3](#_Toc508628696)

[2.1.2 Document Conventions 3](#_Toc508628697)

[2.1.3 Intended Audience and Reading Suggestions 3](#_Toc508628698)

[2.1.4 Product Scope 4](#_Toc508628699)

[2.2 Overall Description 4](#_Toc508628700)

[2.2.1 Product Perspective 4](#_Toc508628701)

[2.2.2 Product Functions 4](#_Toc508628702)

[2.2.3 User Classes and Characteristics 4](#_Toc508628703)

[2.2.4 Operating Environment 5](#_Toc508628704)

[2.2.5 Design and Implementation Constraints 5](#_Toc508628705)

[2.2.6 User Documentation 5](#_Toc508628706)

[2.2.7 Assumptions and Dependencies 5](#_Toc508628707)

[2.3 External Interface Requirements 6](#_Toc508628708)

[2.3.1 User Interfaces 6](#_Toc508628709)

[2.3.2 Hardware Interfaces 7](#_Toc508628710)

[2.3.3 Software Interfaces 7](#_Toc508628711)

[2.3.4 Communications Interfaces 7](#_Toc508628712)

[2.4 System Features 8](#_Toc508628713)

[2.4.1 Temperature Reading 8](#_Toc508628714)

[2.4.2 Sound Reading 8](#_Toc508628715)

[2.4.3 Moisture Reading 9](#_Toc508628716)

[2.5 Other Nonfunctional Requirements 10](#_Toc508628717)

[2.5.1 Performance Requirements 10](#_Toc508628718)

[2.5.2 Safety Requirements 10](#_Toc508628719)

[2.5.3 Security Requirements 11](#_Toc508628720)

[2.5.4 Software Quality Attributes 11](#_Toc508628721)

[2.5.5 Business Rules 11](#_Toc508628722)

[3. Build Instructions 13](#_Toc508628723)

[Code: 19](#_Toc508628724)

[3.4 Android Application 22](#_Toc508628725)

[3.5 Website 22](#_Toc508628726)

[4.1 Conclusion 24](#_Toc508628727)

[4.2 Recommendation 24](#_Toc508628728)

[Bibliography 25](#_Toc508628729)

[Appendix A: Glossary 27](#_Toc508628730)

# List of Figures

[Figure 1 Android application concept GUI pictures. Consists of the initial design for the Login, Power Management, Register, Sound, and Status activity pages. 14](#_Toc506207647)

# Introduction

The HVAC monitoring system solves the technical problem of a more eco-friendly heating and cooling of a building. Other Smart HVAC systems do monitor temperature and moisture levels in the room but the noise generated from the HVAC systems is not monitored. The noise generating from HVAC systems can make it difficult for room occupants to hear, which can be a safety hazard as the listening to prolonged noise can damaged the occupants’ hearing. The noise can also make it difficult for occupants to hear noise generated from other monitoring systems (for example, an alarm system).

This IoT system will monitor and control the temperature, moisture and sound levels in the room using a Broadcom-based development platform (also known as the Raspberry Pi 3 [RPI]), temperature, moisture, and sound sensors, and will transmit the readings from the sensors to a cloud-computing-based service (Amazon Web Services [AWS]) and NoSQL-based database (DynamoDB). The database will interface with an Android application and a website for easy remote access for the user.

The report will detail the process for the concept, building, and feasibility of the project for other users to develop and improve upon the project. Instructions on where to find and how to build the HVAC system are included along with information on issues that we have encountered while working on the project.

# Project Description

# 2.1 Project Requirements Specification

## 2.1.1 Purpose

The purpose of this document is to detail the requirements for the HVAC Android application, and the website component, and the Amazon AWS/DynamoDB database component. The Android application’s purpose is to generate a user-friendly graphical interface for our HVAC hardware and to provide a status report containing the temperature readings, sound-based detection functionality of the system with the maintenance report and scheduler. The database’s purpose is to collect readings from the HVAC system for the Android application and website to fetch from.

## 2.1.2 Document Conventions

This document is written in English following software requirements specifications standards and occasionally uses bullet formatting as well as some concept images of the software product. Software development requirements featured in this document are listed with their own priority and weight.

## 2.1.3 Intended Audience and Reading Suggestions

The intended audience for this document is the project supervisors and application users. This document assumes that the audience has basic technical knowledge with Internet of Things (IoT) projects, Android mobile programming, WordPress/web programming, and AWS/DynamoDB.

## 2.1.4 Product Scope

The Android application scope is to help HVAC users to connect to, control and monitor the system remotely, and receive notifications about the system condition on the screen.

The database is there to store and retrieve information of the status of the hardware for every user-specified time interval.

# 2.2 Overall Description

## 2.2.1 Product Perspective

The product is the latest in the family of smart HVAC products, which monitor temperature and moisture. While maintaining traditional functionality it implements new technologies and provides new features and a new price.

## 2.2.2 Product Functions

* Read Temperature level
* Read Moisture level
* Determine System Functionality via Sound Sensor
* Store Status Reports in the database for the user to look back on

## 2.2.3 User Classes and Characteristics

This product will be used in residential, commercial and industrial buildings. The product will most frequently be used by homeowners and industrial workers.

## 2.2.4 Operating Environment

The operating environment for the project will be the Raspberry Pi 3 with Raspbian Linux, the Android mobile operating system, AWS & DynamoDB for the database environment, and finally, WordPress/HTML for the website.

## 2.2.5 Design and Implementation Constraints

Hardware limitations in terms of the sound sensor sensitivity (cannot be used in an extremely noisy environment), needs embedded C, Internet access for database & website, minimum of Android API 5.0 (Lollipop), app encryption/security, hardware susceptible to extreme weather.

## 2.2.6 User Documentation

The technical report will be provided and the build instructions will be available online: <https://github.com/fntj0052/HVAC>

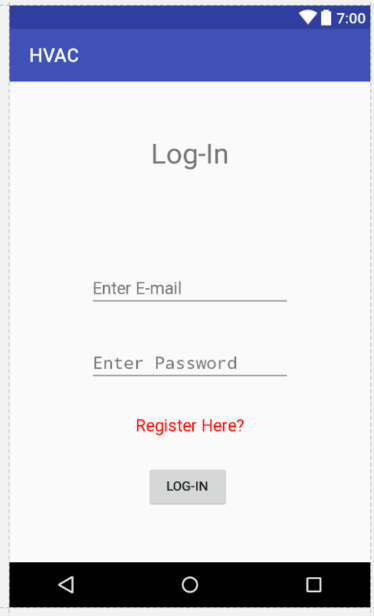
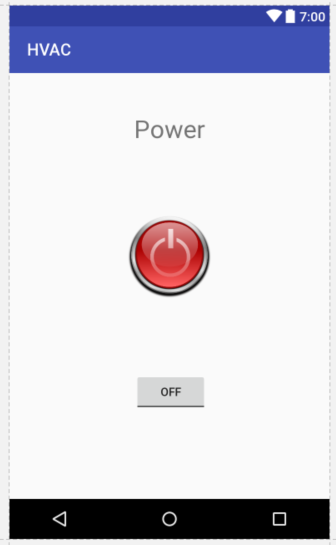
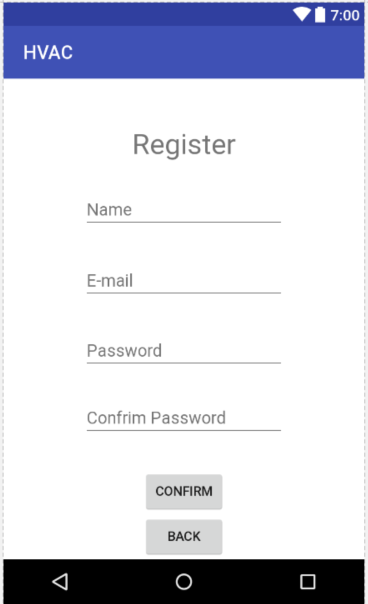
## 2.2.7 Assumptions and Dependencies

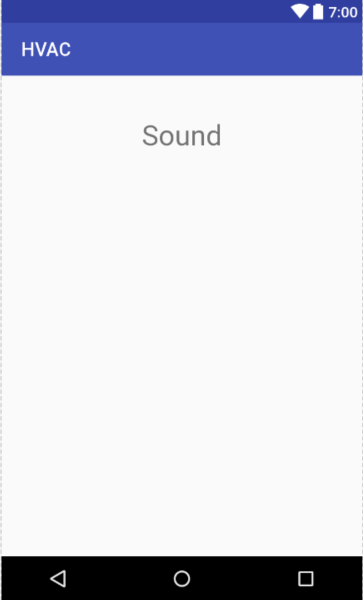
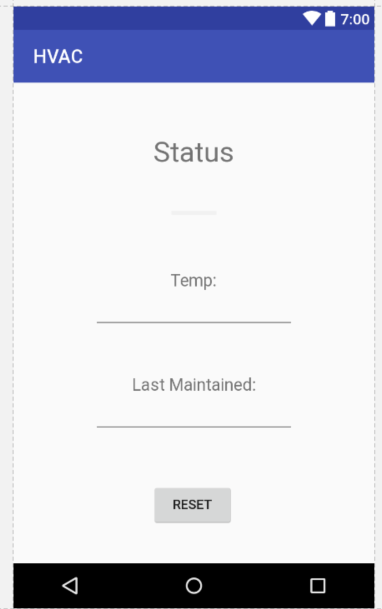
Updates and changes to AWS/DynamoDB software and policies, GitHub, and Android may affect the functionality of the product.

# 2.3 External Interface Requirements

## 2.3.1 User Interfaces

Android GUI standards will be implemented for the mobile application. The head developer of the Android application graphical user interface is Jan Fontanosa.



Figure Android application concept GUI pictures. Consists of the initial design for the Login, Power Management, Register, Sound, and Status activity pages.

## 2.3.2 Hardware Interfaces

For the mobile application, any Android device running Android 5.0 or up will be able to use the application. The software on the Raspberry Pi 3 can be used through the LCD and physical buttons on the keypad. All project developers will be working on the project hardware.

## 2.3.3 Software Interfaces

For the web interface, a connection to the website through a browser will be necessary. The database can be viewed through the AWS website using an AWS account. The operating system used will be a Raspbian Linux. The HVAC system database will be developed and maintained by Maasha Maheson. The product will rely on analogRead to get the analog data and will be processed on C to display to the user through text/images.

## 2.3.4 Communications Interfaces

The product website can be accessed through any modern browser that uses the standard HTTP protocol. The website will be protected through free WordPress protection system but may be vulnerable to security issues. The project website will be developed by Vyacheslav Perepelytsya.

# 2.4 System Features

## 2.4.1 Temperature Reading

2.4.1.1 Description and Priority

The temperature reading is an essential component of the HVAC which will provide a reading of the room temperature to the user and determine whether the HVAC system needs to turn on or off. It’s of HIGH priority. The lead developer of the temperature reading sensor will be Maasha Maheson.

2.4.1.2 Stimulus/Response Sequences

The user is entitled to read the temperature as well as stop and start or restart the system.

2.4.1.3 Functional Requirements

The temperature sensor needs to be functioning as well as the Raspberry PI and the database. In case of a specific error there will be a specific error message to the user, in case of a power outage error or a system shutdown the user will experience a blank screen.

REQ-1: TEMP

## 2.4.2 Sound Reading

2.4.2.1 Description and Priority

The sound reading is MEDIUM priority component of the HVAC which will provide a reading of the noise level of the system environment to the user and determine if the HVAC system is operating at a safe sound level. The lead developer of the sound reading sensor will be Vyacheslav Perepelytsya.

2.4.2.2 Stimulus/Response Sequences

The user is entitled to read the sound level status as well as stop and start or restart the system.

2.4.2.3 Functional Requirements

The sound sensor needs to be functioning as well as the Raspberry PI and the database. In case of a specific error there will be a specific error message to the user, in case of a power outage error or a system shutdown the user will experience a blank screen.

REQ-1: SOUND

## 2.4.3 Moisture Reading

2.4.3.1 Description and Priority

The moisture reading is MEDIUM priority component of the HVAC which will provide a reading of the room moisture levels to the user and determine whether the HVAC system needs to turn on or off. The lead developer of the moisture reading sensor will be Jan Fontanosa.

2.4.3.2 Stimulus/Response Sequences

The user is entitled to read the moisture sensor status as well as stop and start or restart the system.

2.4.3.3 Functional Requirements

The moisture sensor needs to be functioning as well as the Raspberry PI and the database. In case of a specific error there will be a specific error message to the user, in case of a power outage error or a system shutdown the user will experience a blank screen.

REQ-1: MOIST

# 2.5 Other Nonfunctional Requirements

## 2.5.1 Performance Requirements

The Android device should be capable of running Android 5.0 (Lolipop) and have a good internet connection for the AWS/Dynamo DB database to transfer information properly.

## 2.5.2 Safety Requirements

If the device is installed in an extreme weather environment it could damage or harm equipment and people. For example in an earthquake situation the device may affect or damage nearby equipment or people if it is installed improperly or unsafely. Floods and fires may also have unpredictable consequences for the system and nearby devices/people.

## 2.5.3 Security Requirements

The user does not need to provide any personal information aside from the email and password for the Android application.

## 2.5.4 Software Quality Attributes

Sensor accuracy, reliability and a user-friendly GUI are the main additional quality characteristics.

## 2.5.5 Business Rules

The owner of the product and the people they share the device access with will be the only users of the product and as such perform all the possible user roles. In cases of malfunction an additional person may be involved for the repairs.

# 3. Build Instructions

The Build Instructions section describes items purchased, the steps taken and any necessary modifications made during the construction of the smart HVAC monitoring system.

3.1 Bill of Materials

This subsection lists the parts needed to build the HVAC system:

* Raspberry Pi 3 – The Raspberry Pi 2 can be used but an external USB Wi-Fi adapter would need to be used with the Pi in order to maintain the HVAC system’s portability. A USB keyboard and mouse will be needed for the initial setup of the Pi. We have used a Raspberry Pi 3 kit similar to this: <https://www.amazon.ca/CanaKit-Raspberry-Micro-Supply-Listed/dp/B01E4HDIO4/ref=sr_1_9/144-7110608-6031153?ie=UTF8&qid=1520631689&sr=8-9&keywords=raspberry+pi+3>
* LCD Touchscreen Display – Any touchscreen can be used but we have chosen to use the Kuman 7” LCD Display due to its driver support of the Raspberry Pi 3. The screen is not currently sold on Amazon anymore but here is an alternative display that can be used: <https://www.amazon.ca/Kuman-Resistive-800x480-Display-Raspberry/dp/B01F4RSIA2/ref=sr_1_4?m=A3IRH1M32QHQ71&s=merchant-items&ie=UTF8&qid=1520632138&sr=1-4&keywords=raspberry+pi+touch+screen>
* Touch Sensor – This is used as an emergency power off switch for the smart HVAC monitoring system: <https://www.sainsmart.com/products/ttp223b-digital-touch-sensor>
* Moisture Sensor – This is used to monitor the moisture levels in the HVAC system: <https://www.sainsmart.com/products/water-sensor-with-free-cables>
* Sound Sensor – This is used to monitor the sound levels generated from the HVAC system. We have used two sound sensors for the HVAC system but the system can be calibrated to work with one sound sensor also: <https://www.sunfounder.com/sound-sensor-module.html>
* Temperature Sensor – This sensor is used to monitor the temperature levels of the system. We have decided to use the BME280 supplied for the Humber Student Sense Hat mentioned in the Sense Hat subsection. Here is the sensor from the manufacturer’s website: <https://www.sparkfun.com/products/13676>

3.2 Hardware Setup

This section details the setup of the hardware components of the HVAC monitoring system project. It consists of a Broadcom development platform (also known as the Raspberry Pi 3), multiple sensors (such as the temperature sensor, moisture sensor, sound level detector sensor, and the touch sensor), along with a touchscreen LCD interface. The smart HVAC monitoring system was originally written in Python code for the touch and moisture sensors by Jan Fontanosa but converted to Embedded C for better integration of sensor codes. The lead developer for the sound sensors is Vyacheslav Perepelytsya and the lead developer for the temperature sensor is Maasha Maheson.

3.2.1 Raspberry Pi

For this project, we have set up the Raspberry Pi according to the instructions hosted on the supplier’s website: we have used CanaKit’s *Quick Start Guide*, which is listed in the bibliography (CanaKit Corportation). We have also used Raspbian Stretch (Raspbian 9) for operating system as it has support for the libraries needed to use the smart HVAC monitoring system. This operating system and the software needed to operate the Sense Hat were configured according to the instructions listed on the six0four *Student Sense Hat* GitHub repository (Medri & Tian).

3.2.1.1 Raspberry Pi Setup & Operating System Installation

First, connect the HDMI cable to the appropriate connections ports on the Pi and the touchscreen LCD, or connect the cable to an HDMI port on any other display for initial setup. Then, connect an USB keyboard and mouse to the Pi, and optionally, the Ethernet cable to the correct ports on the Raspberry Pi. The Ethernet cable can be used for initial setup with the Pi 3 and is necessary for the Pi 2, if it does not have an external Wi-Fi adapter. Since we have used the Raspberry Pi 3, we have used the onboard Wi-Fi instead to connect to an open wireless network in order to download and install the root certificates needed to connect to an enterprise Wi-Fi network. This process is detailed later in the subsection titled *Enterprise Wi-Fi*. Afterwards, insert the microSD card preloaded with Raspbian Stretch (or NOOBS operating system, depending on what the supplier has included in their kit) and connect the Pi to the microUSB power charger to power it on. Once Raspbian has loaded up the desktop, install the following programs and enable the correct protocols detailed on the six0four *Student Sense Hat* GitHub instructions for the *Humber Parts Crib Raspberry Pi Image Creation*: <https://github.com/six0four/StudentSenseHat/blob/master/cribpisdcard.md>

3.2.1.2 Touchscreen LCD

First, connect the HDMI cable to the appropriate connection ports on the Pi and LCD. Then connect the microUSB to USB cable to the touch port on the LCD and the USB port on the Pi. Afterwards, go to the LCD screen manufacturer’s website and follow the documentation provide by the manufacturer to install the drivers for the screen to the Pi operating system that you are using. For our project, we used a Kuman 7 inch touchscreen LCD and thus we used their website and downloaded and installed the drivers as instructed (Kuman Ltd). We had to select the SC7B link in order to access the specific instructions for the chosen display. We had to modify the installation script included with the manufacturer’s drivers to use the root partition we were actually using for the Raspberry Pi 3. Without this modification, the Raspberry Pi would not boot correctly.

3.2.1.3 Enterprise Wi-Fi

In order to connect to an enterprise Wi-Fi network, we had to download and install the root certificate hosted by the network administrators. Once the root certificate was downloaded, we had to change the file type from .cer to .crt and moved the certificate to the /usr/share/ca-certificates directory. Then we executed sudo dpkg-reconfigure ca-certificates in order for to install the new certificate. After we had to edit /etc/wpa\_supplicant/wpa\_supplicant.conf in order to configure the enterprise Wi-Fi as followed on the *Student Sense Hat* GitHub (Medri & Tian) and listed below:

network={

ssid="myWi-Fi@Humber"

priority=999

proto=RSN

key\_mgmt=WPA-EAP

pairwise=CCMP

auth\_alg=OPEN

eap=PEAP

identity="STUDENT ID"

password="PASSWORD"

phase1="peaplabel=0"

phase2="auth=MSCHAPV2"

}

3.2.2 Sensor Hat PCB

We have used the six0four *Student Sense Hat* GitHub instructions to build and configure the PCB & sensors and did not use surface resistor. Vyacheslav Perepelytsya has stripped down the PCB by removing the transistors and resistors in order to provide proper voltages needed for the sound sensors to correctly work. The main difference is that the connection to the PCF is now direct, changing the voltage supply from 5V to 3.3V that is better suited for the project. Resistors R4-R7 are removed, and instead direct paths are provided (Through soldering cables or re-design). Q1 and Q2 (transistors) components are also unnecessary. After the sensor hat as been made, it can be mounted on top of the Raspberry Pi.

3.2.2.1 Temperature Sensor

We have used the temperature sensor provided in the Sense Hat kit.

3.2.2.2 Moisture Sensor

First, connect the – (minus) pin to the ground pin on the Raspberry Pi 3 (Pin 6), the + (plus) pin to the 5 V on Pi (pin 2), and the S-indicated pin to pin 18 according to the Raspberry Pi 3 GPIO pin diagram [GPIO24]. Then, connect the sensor to the Sense Hat, which should be connected to the Pi.

#include <wiringPi.h>

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

int RCpin = 5;

int dry = 0;

int wet = 1;

int Rpin = 2;

wiringPiSetup();

pinMode(RCpin, INPUT);

void RCtime (RCpin){

int reading = 0;

time.sleep(0.1);

int sensorState = digitalRead(RCpin);

while(1){

if (sensorState == 0){

reading += 1;

}

if (reading >= 1000) {

return 0;

}

if (sensorState != 0){

return 1;

}

}

return 0;

}

void buzz\_on(pin){

pinMode(Rpin, INPUT);

pinMode(Rpin, HIGH);

}

void buzz\_off(pin){

pinMode(Rpin, INPUT);

pinMode(Rpin, LOW);

}

int main (void){

printf("Waiting for wetness..");

while(1){

time.sleep(1);

if (RCpin == wet);

printf ("Sensor is wet");

while(1){

time.sleep(1);

if (RCpin == wet);

printf("Sensor is still wet...");

buzz\_on(2);

continue;

if (RCpin == dry);

buzz\_off(2);

printf("Sensor is dry again...");

break;

}

return 0;

}

return 0;

}

* + - 1. Sound Level Detector Sensor

Sound sensors were connected each to a 3.3V voltage output and ground on the PCB, as well as 1 PCF input slot for each, all of this was connected to the modified PCB (See modifications under PCB heading) from CENG Hardware Project. The time needed was no more than 2 hours of soldering, no more than 2 hours of programming and calibration, and was around 30 minutes of connecting parts.

The program needed to test the sound sensor is included below and in the HVAC GitHub link under the name NoiseDetector.c. Run the code with administrator permissions: sudo ./NoiseDetector. Compile the code with gcc –Wall –o NoiseDetector NoiseDetector.c –lwiringpi -lm

Each of the sound sensors should be tested and calibrated appropriately to have close to equal sound sensitivity. If not, the PCF pin slots or the sound sensor hardware itself may need to be replaced. Changing cables and the PCF input slots for the sound sensors can provide a different result.

## Code:

#include <stdio.h>  
#include <stdlib.h>  
#include <wiringPi.h>  
#include <pcf8591.h>  
#include <math.h>

#define PCF 120

int main (void) {   
int value1, value2;   
int data1, data2;   
int counter = 0;   
int step = 1;   
//1 int i = 0;  
 int offset = 85;   
//85 int deviation = 10;   
//10 int noiseCounter = 0;  
int noiseDuration = 5;   
//20 int noiseOffCounter = 0;  
 int noiseOffDuration = 1000;  
 //1000 int noise = 0;

if (wiringPiSetup () == -1) {  
 printf("Error at wiringPiSetup()");  
 return 1;  
 }

pinMode (0, OUTPUT) ;   
// aka BCM\_GPIO pin 17 pinMode (1, OUTPUT) ;   
// aka BCM\_GPIO pin 18

// Setup pcf8591 on base pin 120, and address 0x48   
// printf("%d\n", ++step);

if (pcf8591Setup (PCF, 0x48) == -1) {  
 printf("Error at pcf8591Setup()");  
 return 1 ;  
 }  
 printf("Lowest noise level \t\t\t%d\n", offset);  
 printf("Data has been generalized with weight value of %d\n", step);

while(1) // loop forever   
{ data1 += analogRead (PCF + 0);  
 //white - local data2 += analogRead (PCF + 1);  
 //yellow - remote counter++;  
 if(noise) { noiseOffCounter++;  
 }

if(counter == step ) {   
counter = 0;  
 value1 = round((double)data1 / step);  
 value2 = round((double)data2 / step);  
 data1 = 0;  
 data2 = 0;

if ((value1 < offset) || (value2 < offset)) {

if( abs (value1 - value2) < deviation) {  
 if (++noiseCounter > noiseDuration) {  
 noiseCounter = 0; noiseOffCounter = 0; noise = 1;   
} } else {  
 if(noise) {  
 if(noiseOffCounter > noiseOffDuration) {  
 noiseOffCounter = 0; noise = 0;   
} } else if( --noiseCounter < 0) {  
 noiseCounter = 0;   
} }

if(noise) {  
 printf("\n\tNOISE !!!! #1:%3d #2:%3d", value1, value2);  
 //both microphones are sensing noise - red digitalWrite (0, LOW);  
 digitalWrite (1, HIGH);  
 } else {   
printf("\n\tVoice In!! #1:%3d #2:%3d", value1, value2);   
// Voice in at any of microphones - green digitalWrite (0, HIGH); digitalWrite (1, LOW); }

for(i = offset; (i > offset - value1) && (i > 0); i--) {  
 printf(" ");   
} for(i = offset; i > value1; i--) {  
 printf("-");  
 }   
printf("|");  
 for(i = offset; i > value2; i--) {  
 printf("+");  
 } } else {  
 if(!noise) { digitalWrite (0, LOW);  
 digitalWrite (1, LOW); } } } }

return 0; }

3.2.2.4 Touch Sensor

First, connect the G-indicated pin to the ground pin on the Raspberry Pi 3 (Pin 9), V is to 3.3 V on Pi (pin 1), and the out to pin 16 according to the Raspberry Pi 3 GPIO pin diagram [GPIO23].

#include <wiringPi.h>

#include <stdlib.h>

#include <stdio.h>

int padPin =4;

int alreadyPressed =0;

int main(void);

{

wiringPiSetup();

pinMode(padPin, INPUT);

while(1)

{

int padPressed = digitalRead(padPin);

if(padPressed)

{

int padPressed = digitalRead(padPin);

if(padPressed)

{

if(alreadyPressed){}

else

{

printf("Pressed.\n");

alreadyPressed = 0;

}

}

else

{

alreadyPressed =0;

}

delay(100);

}

return 0;

}

3.3 AWS & DynamoDB Database

Maasha Maheson was in charge of the AWS IoT and DynamoDB integration. After signing up for the free-tier AWS account, click the IoT services button. We followed the Connecting to the Raspberry Pi AWS API guide to configure the IoT services on the Pi. We then followed the Embedded C SDK guide to install the SDK onto Raspberry Pi: we added the source code for the external programs into the appropriate folders and then executed the make command on the SDK parent directory to compile both the external dependencies for the SDK and the actual Embedded C SDK. To test AWS IoT, we have used the sample programs in folder to change the config.h file for the program to connect to the Rest API link listed in the AWS IoT website.

To setup DynamoDB, we have used the created AWS account to create a DynamoDB table with the default options configured.

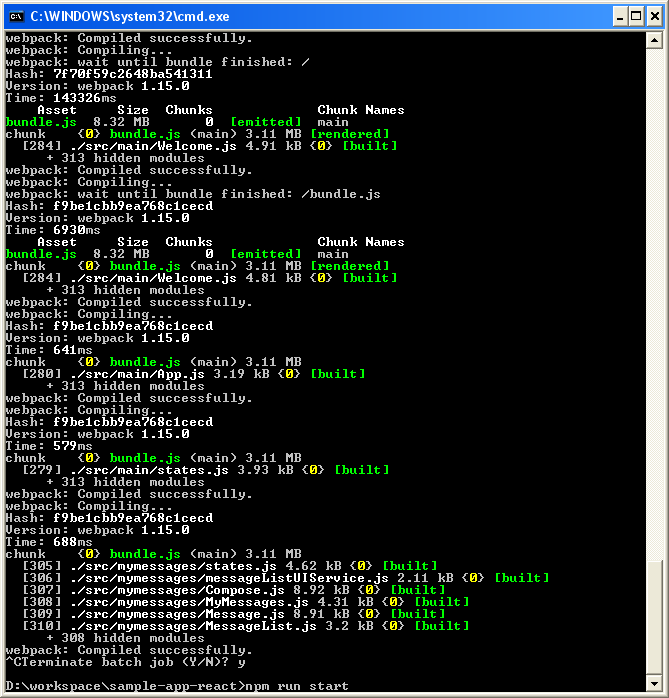
# 3.4 Android Application

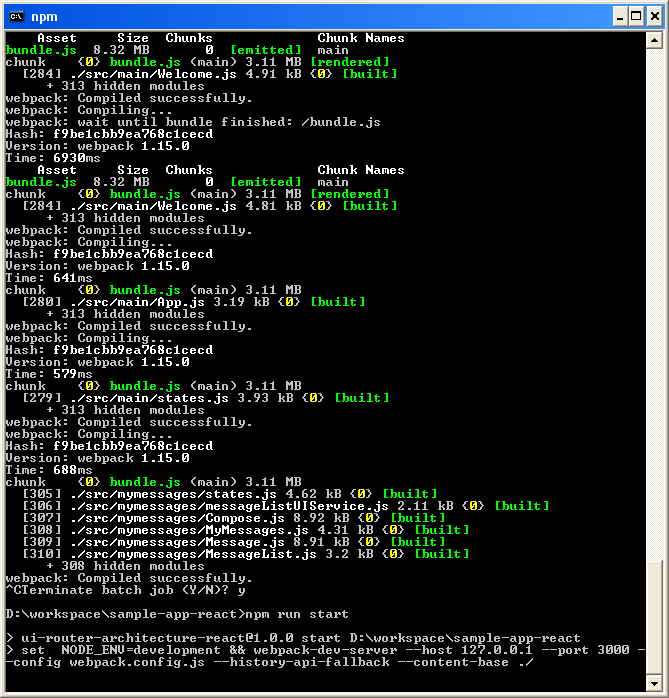
Jan Fontanosa was in charge of the Android application. The application allows the user to register an account with the project, login to this account, monitors the HVAC system, and notifies the user of the HVAC condition. Used Android Studio to create application and to test the application, the IDE’s emulator and an Android mobile device was used. The Android application can be designed in anyway provided that the main functionality is there. The main Java and XML code is included below.

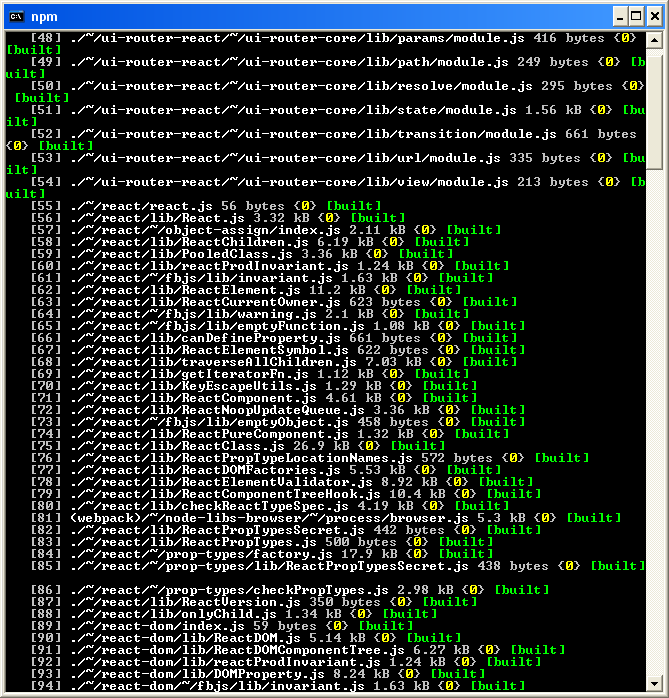
# 3.5 Website

The website is lead developed by Vyacheslav Perepelytsya. The website interface is under development using HTML, CSS, HTTP RESTful protocol, Simba JDBC drivers, ReactJS and NodeJS 3 tier system to interface with the Amazon AWS IoT and DynamoDB database created before. We have used Tomcat server for local testing of the website and Spring Tools Suite to develop the code. We originally used WordPress but it was not compatible with DynamoDB. The website will show a status report generated from the data in the database.

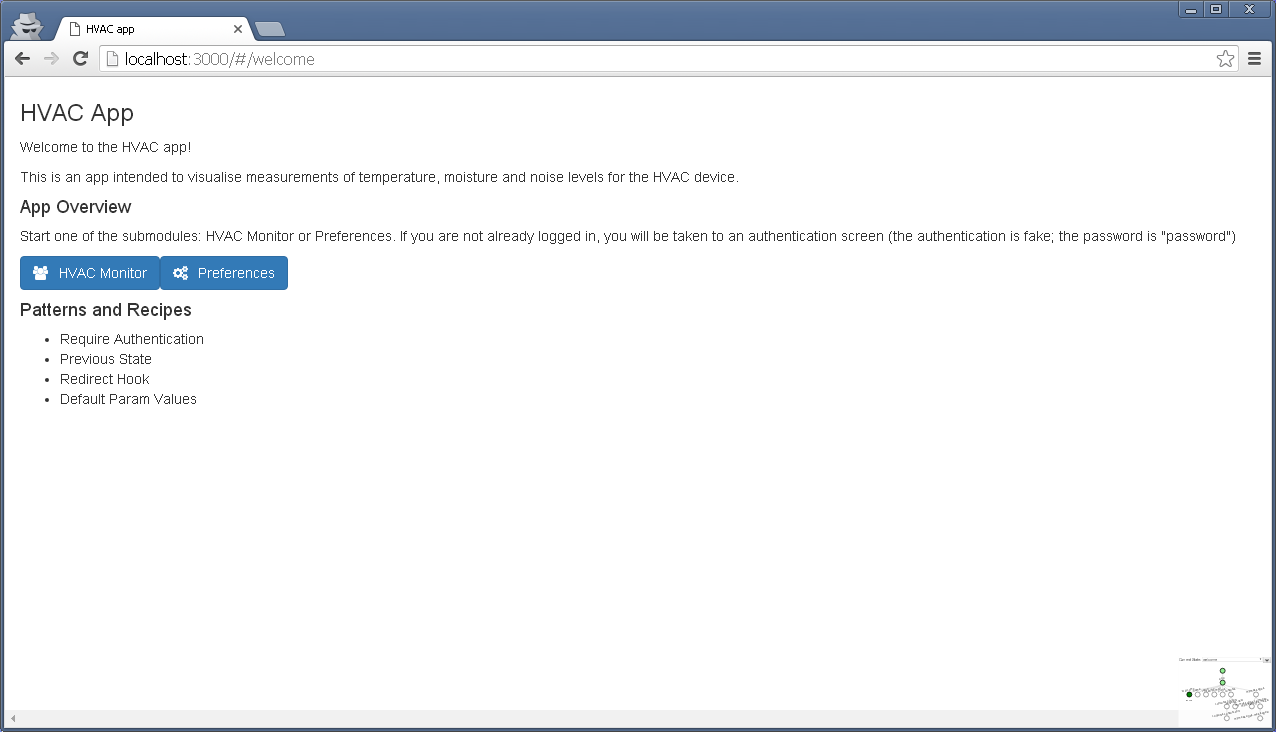
nodeJS + reactJS overview:



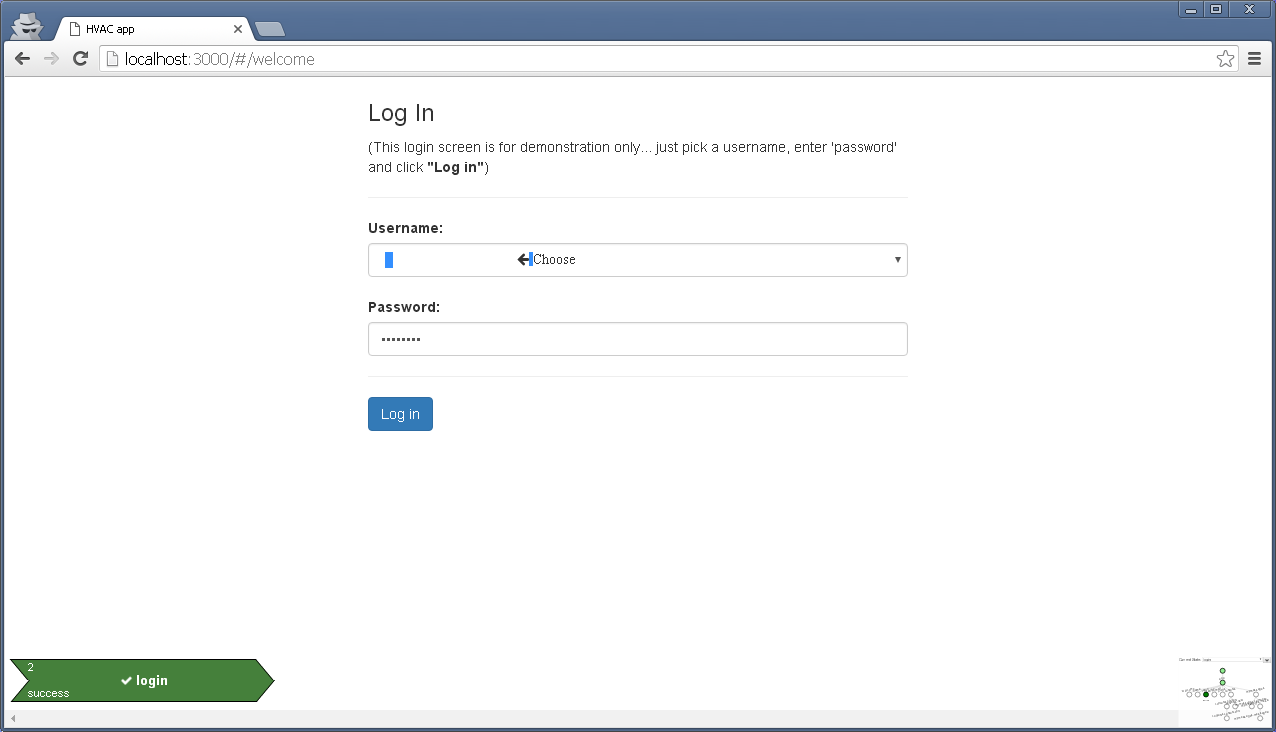




Welcome page:

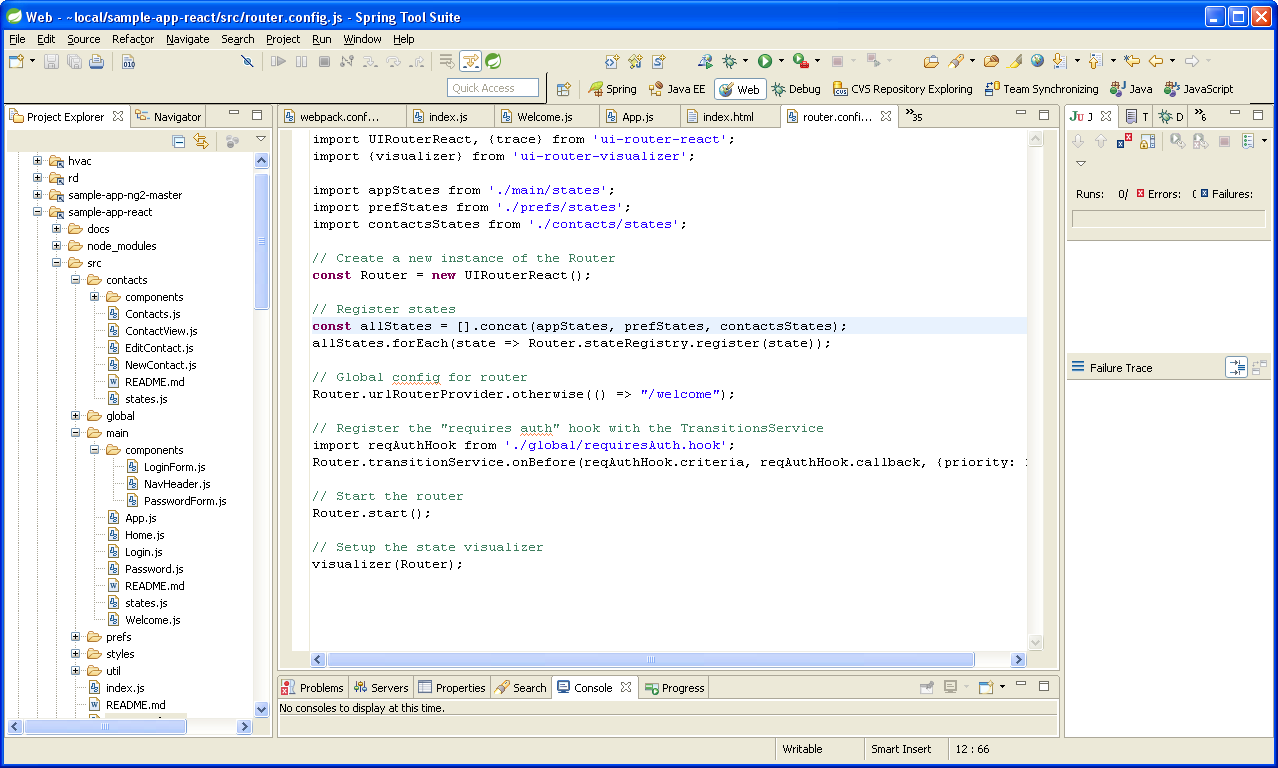


Login page:

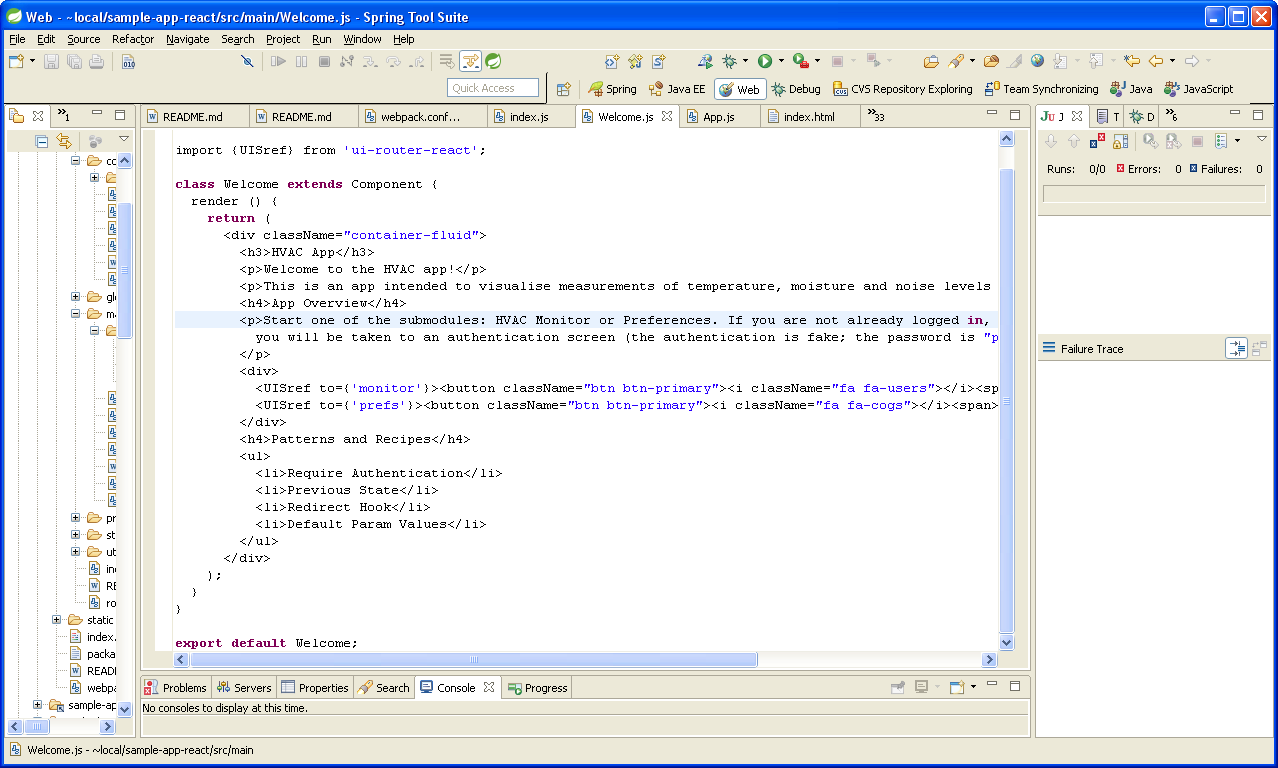


ReactJS code:

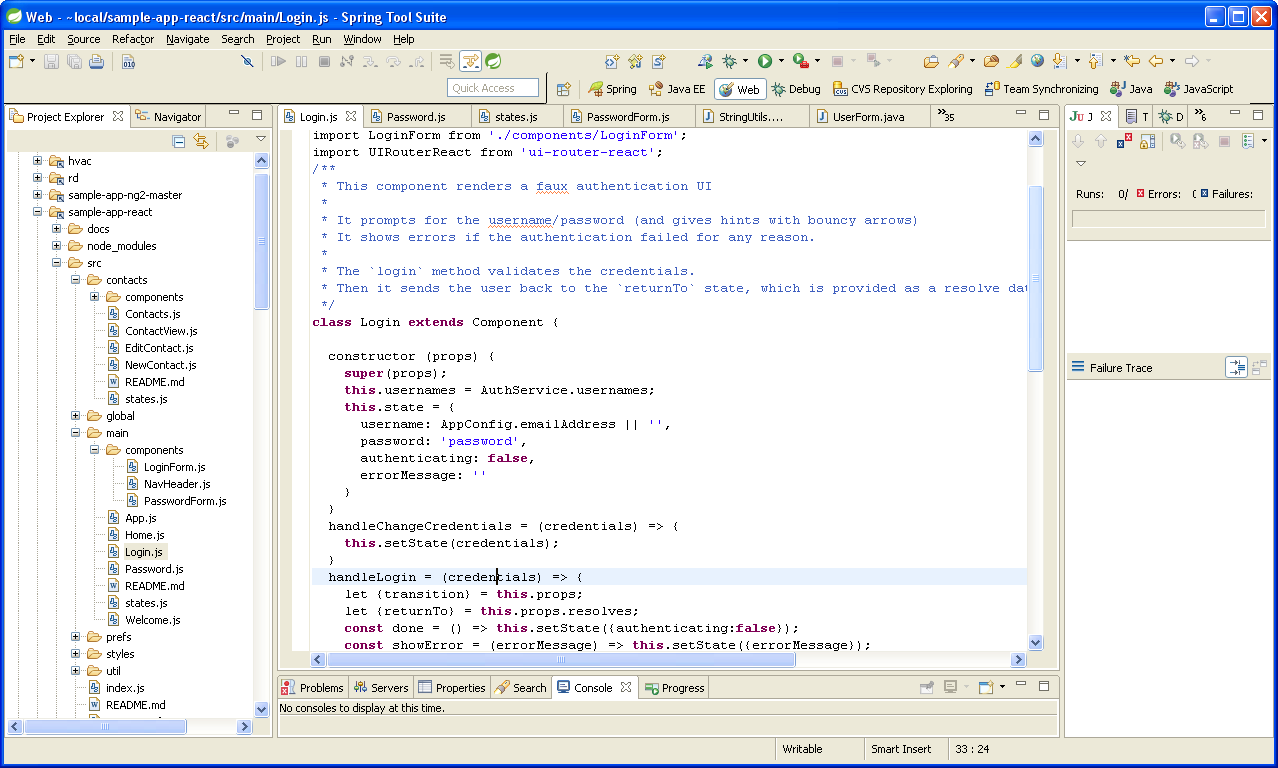
Router (main module for transitions between states/pages/views…)



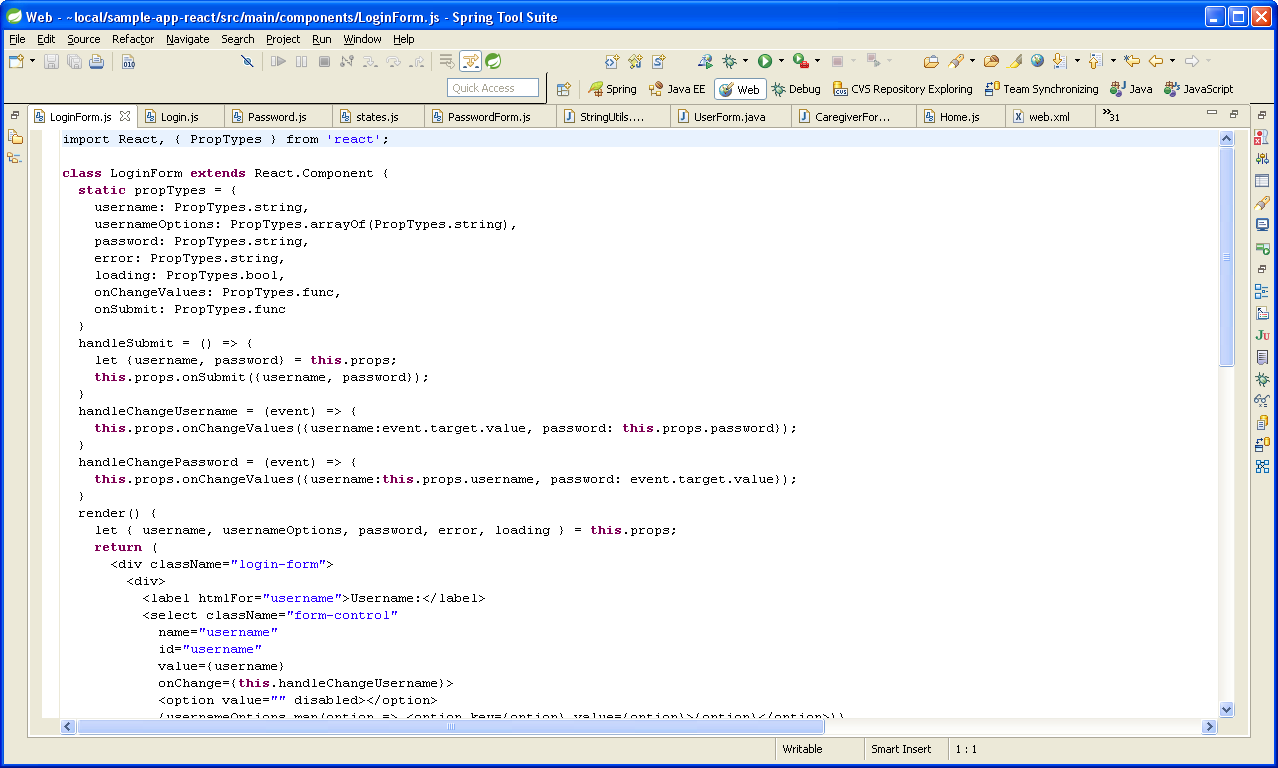
Welcome page:



Login page:

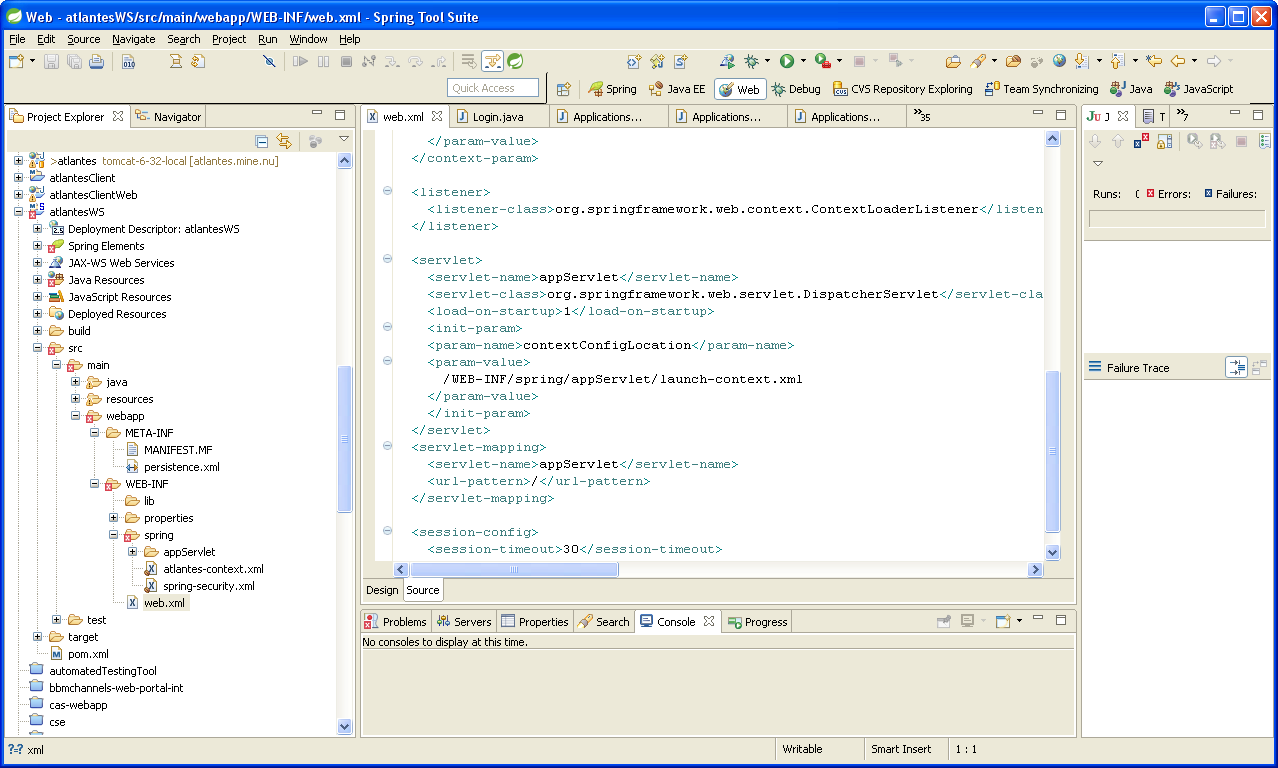


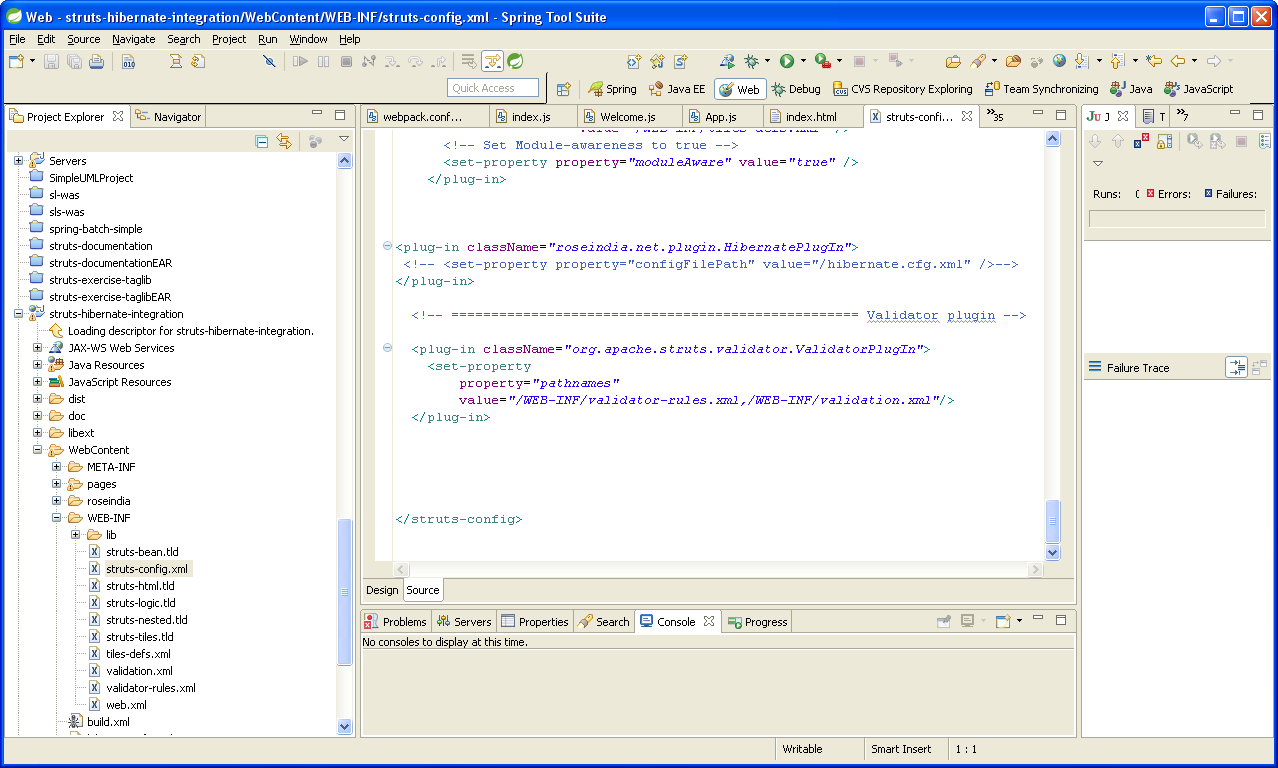
Login form (included in Login page as component)



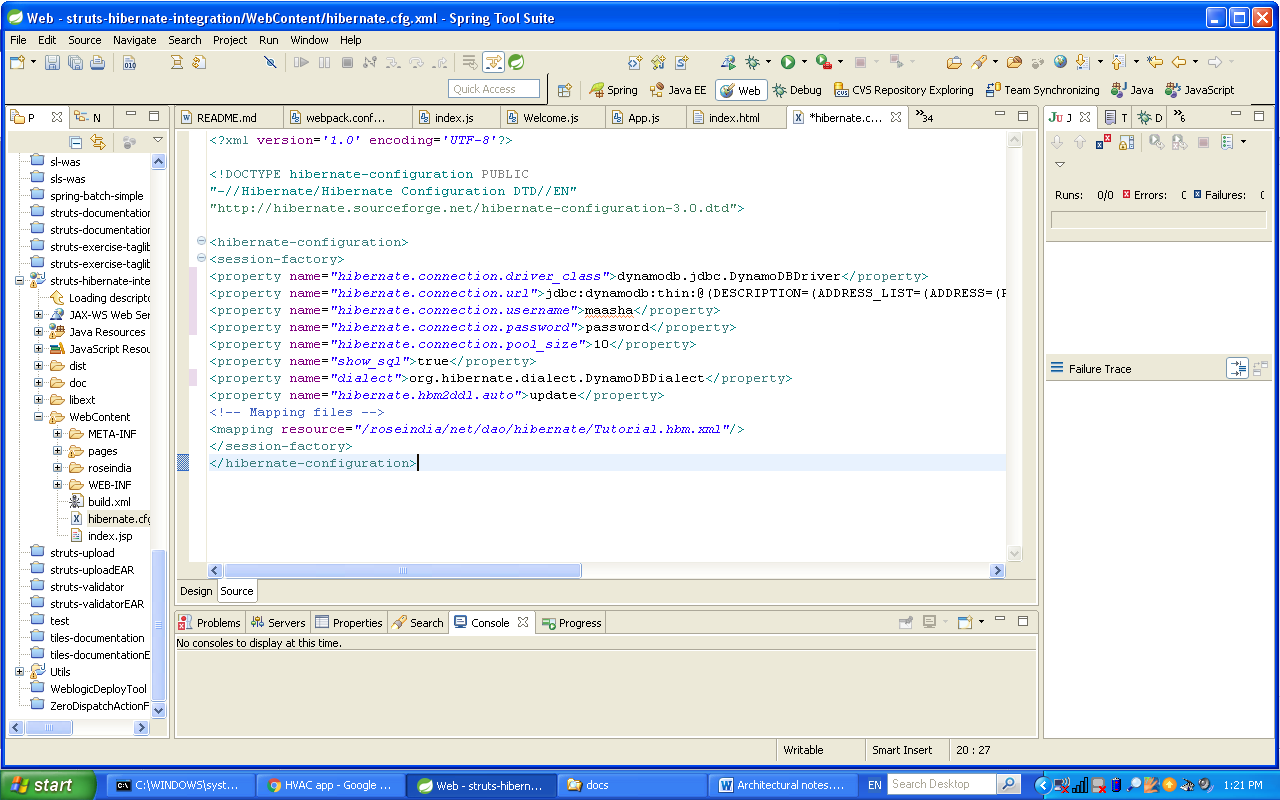
Descriptors:

web.xml





Hibernate.cfg.xml



# 4.1 Conclusion

# 4.2 Recommendation

# Bibliography

Al-Ali, A. R., Alikarar, M., Gupta, R., Rashid, M., Zualkernan, I.A. (2017). A smart home energy management system using IoT and big data analytics approach. IEEE Transactions on Consumer Electronics, 63(4), 426-434. Retrieved from <http://ieeexplore.ieee.org/document/8246800>

Amazon Web Services, Inc. (2012, August 10). Amazon DynamoDB Developer Guide. Retrieved from <http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/>

Amazon Web Services, Inc. (n.d). AWS IoT Developer Guide. Retrieved from <http://docs.aws.amazon.com/iot/latest/developerguide/>

CanaKit Corporation. (n.d). Raspberry PI Quick Start Guide. Retrieved from <https://www.canakit.com/quick-start/pi>

Google LLC & Open Handset Alliance. (n.d). Android API Guide. Retrieved from <https://developer.android.com/guide/index.html>

Ecovent Systems Inc. (n.d.). Ecovent. Retrieved from <https://www.ecoventsystems.com/smart>

Kuman Ltd. (2016, May 12). Documents and recources,Support & Tutorials. Retrieved from <http://www.kumantech.com/help/documents-and-recources_h0037.html>

Institute of Electrical and Electronics Engineers. (2015, August 28). IEEE Xplore Digital Library [Online]. Retrieved from <https://ieeexplore.ieee.org/search/advsearch.jsp>

Medri, Kristian & Tian, Austin. (2017). six0four/StudentSenseHat: Student Sense Hat. Retrieved from https://github.com/six0four/StudentSenseHat

SoundFounder.com. (2017, April 13). Lesson 19 Sound Sensor. Retrieved from <https://www.sunfounder.com/learn/lesson-19-sound-sensor-sensor-kit-v2-0-for-b.html>

Wordpress.org. (n.d). WordPress APIs << WordPress Codex. Retrieved from <https://codex.wordpress.org/WordPress_APIs>

# Appendix A: Glossary

**AWS** – Amazon Web Services. Cloud computing-based software.

**DynamoDB** – A NoSQL database service that can be used with AWS

**GitHub** – A website to share and develop code/projects using version control

**IoT** – Internet of Things. Products/systems that connect to and communicate using the Internet

**Raspberry PI 3** – a recent version of a portable computer/Broadcom development platform used for practical projects and programming education.