Technology Innovation   
& Architecture  
Summer, 2017

**Group Project Name**: Humanoid Accessibility

*Callie Warren|Truman State University|Bachelors in Computer Science|2017*

*Chia-wei Chang|University of Texas A&M|Masters in Computer Science|2017*

*Jaydeep Ingle|University of New York, Binghampton|Masters in Computer Science|2017*

*Ryan Brady|University of Central Missouri|Bachelors in Computer Science|2016*

*Samantha Stanley|University of Central Missouri|Bachelors in Mathematics & Computer Science|2019*

*Utsav Dholakia|University of Texas at Dallas|Masters in Computer Science|2017*

Contents

[Project 3](#_Toc490467990)

[Objective 3](#_Toc490467991)

[Description/Background 3](#_Toc490467992)

[Benefits 3](#_Toc490467993)

[Architecture 3](#_Toc490467994)

[Code – Sanbot Application 4](#_Toc490467995)

[Location 4](#_Toc490467996)

[Language/Tools 4](#_Toc490467997)

[Design 4](#_Toc490467998)

[Code – Phone Application 4](#_Toc490467999)

[Location 4](#_Toc490468000)

[Language/Tools 4](#_Toc490468001)

[Design 4](#_Toc490468002)

[Code – Sanbot Server 4](#_Toc490468003)

[Location 4](#_Toc490468004)

[Language/Tools 5](#_Toc490468005)

[Design 5](#_Toc490468006)

[User Experience 5](#_Toc490468007)

[Demonstration 6](#_Toc490468008)

[Demonstration – Sanbot Application 6](#_Toc490468009)

[Requirements 6](#_Toc490468010)

[Set up the Demo 6](#_Toc490468011)

[Run the Demo 6](#_Toc490468012)

[Demonstration – Phone Application 6](#_Toc490468013)

[Requirements 6](#_Toc490468014)

[Set up the Demo 7](#_Toc490468015)

[Run the Demo 7](#_Toc490468016)

[Demonstration – Sanbot Server 7](#_Toc490468017)

[Requirements 7](#_Toc490468018)

[Set up the Demo 8](#_Toc490468019)

[Run the Demo 8](#_Toc490468020)

# Project

## Objective

Program a humanoid greeter that can identify a person and greet that person as he or she walks into a retail store.

## Description/Background

Sanbot is a QIHAN Technology product with a MSRP of $12,000. The robot is Android-based, which makes the robot easy to program and work with. Most of the Sanbot functions (Follow, playing Duck, Dance, Play Music, etc) are applications that you can access using the tablet on the robot’s chest. The following image shows a Sanbot.



Sanbot has Wi-Fi and Bluetooth technology, projectors, 3D camera, subwoofers, USB ports, and much more. Many industries use Sanbots, including Retail, Hospitality, Security, Healthcare, and Education. For more information regarding Sanbot, refer to <http://en.sanbot.com/index.html>.

In this design, the Sanbot is stationed at the entrance of the retail store. The Sanbot waits for customers to walk in, welcoming them vocally by name, as well as displaying a message unique to that customer through its built-in projector.

## Benefits

The function is universally understandable. Sanbot greets the customer through sound and display. Offering both types of greeting makes this Sanbot project effective for most customers, including those who are hearing or sight impaired. The project can be adapted to include multiple languages for additional versatility.

# Architecture

The code of the Humanoid Greeter project is divided into three separate areas, including the Sanbot Application, Phone application, and Sanbot Server.

## Code – Sanbot Application

### Location

The code is located in the following GitLabs repository: git@git.sprintlabs.io:interns2017/Sanbot/HumanoidGreeter/SanbotApplication.git

### Language/Tools

Android Studio, Java, Sanbot SDK

### Design

The app is constantly listening for data from the server. The moment Sanbot receives data for a user (when the user walks into the store). That data appears as a welcome message with that user’s name, displayed on the projector. Concurrently, through the Sanbot SDK, the greeter voices a similar message. The following code snippet is from the RobotService.java file.

SpeechManager speechm = (SpeechManager) getUnitManager(FuncConstant.*SPEECH\_MANAGER*);

ProjectorManager projm = (ProjectorManager)getUnitManager(FuncConstant.PROJECTOR\_MANAGER);

projm.switchProjector(true);

String welcomeMsg = "Welcome to Sprint "+user.getFirstName() + **" "** + user.getLastName();

speechm.startSpeak(welcomeMsg);

## Code – Phone Application

### Location

Repository: git@git.sprintlabs.io:interns2017/Sanbot/HumanoidGreeter/PhoneApplication.git

### Language/Tools

The code is written using the React Native language. It is a framework based on JavaScript to create universal mobile apps in JavaScript code. Along with that, we used the “react-native-beacon-manager” library for scanning beacons. We used Webstorm IDE from Jetbrains for development.

### Design

The mobile app has two pages. Users enter their information on a registration page. The user is registered using the API (“/register”) provided by the Sanbot server. If the server-based registration is successful, the user information is stored in a local database on the phone.

Then, the app transitions to a static page that displays the user's registration information. There, the app scans for beacons with given “uuid” and “identifier”. When the given beacon is in range, the app sends the user’s email and location to the API(“/find”) on the Sanbot server.

## Code – Sanbot Server

### Location

The code is located in the test\_server directory of the following GitLabs repository: git@git.sprintlabs.io:interns2017/Sanbot/SanbotAppsServer.git.

### Language/Tools

Python

### Design

The view.py file contains the API used in this project. Specifically, functions added to view.py include:

* /Reset: for inventory db restart, RFinit: to initialize inventory scenario database
* /inventory to handle inventory process
* /show, /get Missing, and /getTotal are used for the front-end display
* /register is used to register the user data
* /find receives the user beacon location and forwards that information to the end-point robot

The \_\_init\_\_.py file configures and creates a flask server with mongo engine. Log.out stores the log information. The model.py file provides the schema used to collect data in the database, shown below.

——————————————————————————————————————————

model.py

——————————————————————————————————————————

class User(db.Document):

firstName = db.StringField(max\_length=255, required=True)

lastName = db.StringField(max\_length=255, required=True)

email = db.StringField(max\_length=255, required=True)

physicalDisability = db.StringField(max\_length=255)

language = db.StringField(max\_length = 255)

phoneMacAddress = db.StringField(max\_length = 255)

phoneNumber = db.StringField(max\_length = 255)

class Store(db.Document):

physical\_address = db.StringField(max\_length=511, required=True)

client\_ip = db.StringField(max\_length=255, required=True)

latitudeMin = db.FloatField(required = True)

latitudeMax = db.FloatField(required = True)

longitudeMin = db.FloatField(required = True)

longitudeMax = db.FloatField(required = True)

class RFID\_Reader(db.Document):

tagId = db.StringField(max\_length=255, required=True)

product = db.StringField(max\_length=255, required=True)

class missing(db.Document):

tagId = db.StringField(max\_length=255, required=True)

product = db.StringField(max\_length=255, required=True)

# User Experience

The end user first registers by filling out a form on the Android phone app. After registering, the end user sees the phone display their information.

The end user then sees the following yellow pop-ups on the phone if the phone is in DEV mode. (For instructions on how to enable DEV mode, go to the "Demo - Phone application" section of this document.)

* “User registered successfully”
* “Beacon monitoring started successfully”

When the phone app identifies a beacon in range, the end user sees the following pop-up appear on the phone: “User location send".

When in range, the robot greets the end user by projecting the end user's first and last name on the tablet screen. At the same time, the robot speaks the end user’s name.

# Demonstration

The Humanoid Greeter project is a single demonstration comprised of the following parts.

* Demo - Sanbot Application
* Demo - Phone Application
* Demo - Sanbot Server

## Demonstration – Sanbot Application

### Requirements

Android Studio, Sanbot SDK, Sanbot

### Set up the Demo

1. Open Android studio.
2. Click **Import project** and select the location of the Sanbot application.

### Run the Demo

1. Plug your laptop into the Sanbot.
2. Install and run the app through Android Studio.

The app should appear on the Sanbot tablet to let you know the installation was successful. After that, Sanbot will be listening to welcome all incoming customers.

## Demonstration – Phone Application

### Requirements

1. Verify the internet connection. Internet access is needed to run the demonstration.
2. Download and install Node.js (version 8.2.1 or higher).
3. Download and install npm (latest version).
4. Install “react-native-cli” using the following command.

npm install -g react-native-cli.

1. Download and install package the dependencies “react-native-beacons-manager” and “react-native-config” using the following command, replacing the *package\_name* as appropriate.

npm install -g <*package\_name*>

If an error “Cannot find module <package\_name>” occurs, try (npm install --save <*package\_name*>) or use the same command with sudo.

1. Download and set up Android SDK for Android deployment.

### Set up the Demo

1. To make sure that the Sanbot server is reachable from the same WiFi as the phone. Change the serverAddress to the correct “<*IP\_address*>:<*port\_number*>” value in the ".env” file in the root directory with the address values for the Sanbot Server.

Note: The server and phone do not have to be on the same local network, although that would increase performance. Only the server needs to be reachable from the phone.

1. Verify the uuid of the beacon in “.env” file.
2. Navigate to the project directory and run the following command. (Run this once per machine.)

react-native link react-native-beacons-manager

1. Connect the android phone to machine and enable USB debugging on the device.
2. So that you can enable debugging on the app, keep same WiFi connected on both phone and machine.

### Run the Demo

1. To build and install app on the device, run the following command.

react-native run-android

If the command works but the app launch reports an error such as "Unable to download JS bundle. Did you forget to start the development server or connect your device", run this command.

npm run android-linux

1. To enable debugging of the app:
   1. After the app launches, shake the device and from the dialog box, select **Dev settings**> **Debugging: Debug server host & port for device**, supply <*your machine’s IP address*>:8081”.
   2. Verify that **Performance: JS Dev Mode** is enabled.

If you don’t want to enable debugging, disable **Performance: JS Dev Mode** to increase performance.

1. To enable remote debugging, shake the device again and select **Debug JS Remotely**. This action opens Google Chrome on your machine with a new tab. There, you can debug the app as you would any other JavaScript code.

## Demonstration – Sanbot Server

### Requirements

* Ubuntu 14.04
* Python 2.7.8
* Pip 1.5.4
* Flask 0.10.1
* Supervisor 3.0b2
* Socket (latest)

### Set up the Demo

1. Activate the mongo daemon by running the following command in directory where your /mongodb-osx-x86\_64-3.4.6/bin is located.

./mongod

### Run the Demo

1. Run the following command in the /test\_server directory.

python manager.py run server