Know It's Off: Requirements document

Benjamin Martin Tristan Jong Kyle Peterson
Oregon State University
School of Electrical Engineering and Computer Science
Know It's Off Senior Capstone Project
Last edited: October 25 2019

Abstract

The requirements for the *Know It's Off* system are simple, yet robust objectives. As a full 6-member team, we have been tasked with continuing the work from last year's *Know It's Off* senior capstone team which previously consisted of only 3 ECE members. This year, we hope to build on their solution by improving, tweaking, and replacing components of their existing product as outlined below. The objective of the project, at its core, is quite simple: to provide the user with information when they are away from home on the status of one or more appliances in their house. In this way, the user can prevent potential future hazards from cropping up (such as an oven that was left on).

I. Introduction

A. System purpose

THE purpose of this system is to provide the user with information when they are away from home on the status of one or more "dumb" appliances in their house which are not connected to the internet. To achieve this purpose, the Know It's Off project will develop devices called *Scouts* that can be attached to chosen appliances and will provide information on various metrics of the attached appliance, such as its on/off status.

B. System scope

This project will continue the work from last year's *Know It's Off* senior capstone team. Last year's team concluded with a large, visually unappealing device that could fairly accurately report if the attached device was on or off to a blank webpage that said the word "on" or "off." We will be building on their solution by either improving, tweaking, or replacing existing components of their solution. We have also been tasked with improving the existing hardware in several ways, including reducing the device's size and improving its battery life. The device now will consist of several *Scouts* that provide the metric information on attached devices, and there will be one *Queen Bee* per residence.

C. System overview

This system will be as self-sustaining as possible, that is, it will run on its own with minimal user interaction. The system will shut itself off when necessary or during off-hours, the system will update itself as necessary, and will be flexible with changes deployed on the user end (such as adding more devices/sensors). Initial deployment of the system will also be as hassle-free as possible and will require minimal user input. This will be achieved by requiring the user to pair scouts to a wi-fi network by pairing it with a mobile device such as a phone. After this, they will make an account on either the mobile device application or web browser application and then register the device to their account. From then on, they will be able to check the information the scouts are reporting from either the mobile device application or web browser application.

II. System Requirements

A. Hardware

Hardware requirements are not a concern for the computer science side of the project, so we will omit the requirements here. For hardware requirements, please refer to the ECE team's requirements document for *Know It's Off*.

B. Security

We will be providing security to the system by ensuring a secure channel of communication between the sensor and receiver is established and maintained. There will be three components of securing the system.

1) Message confidentiality: Ensuring the message is not able to be viewed by an adversary.

2) Message integrity: Ensuring the message is not modified or tampered in any way by an adversary.

3) Origin authenticity: Ensuring and verifying the message came from the correct source.

We consider these three attributes to be the most vital for ensuring a secure system. We will achieve these goals with the following protocols and tools:

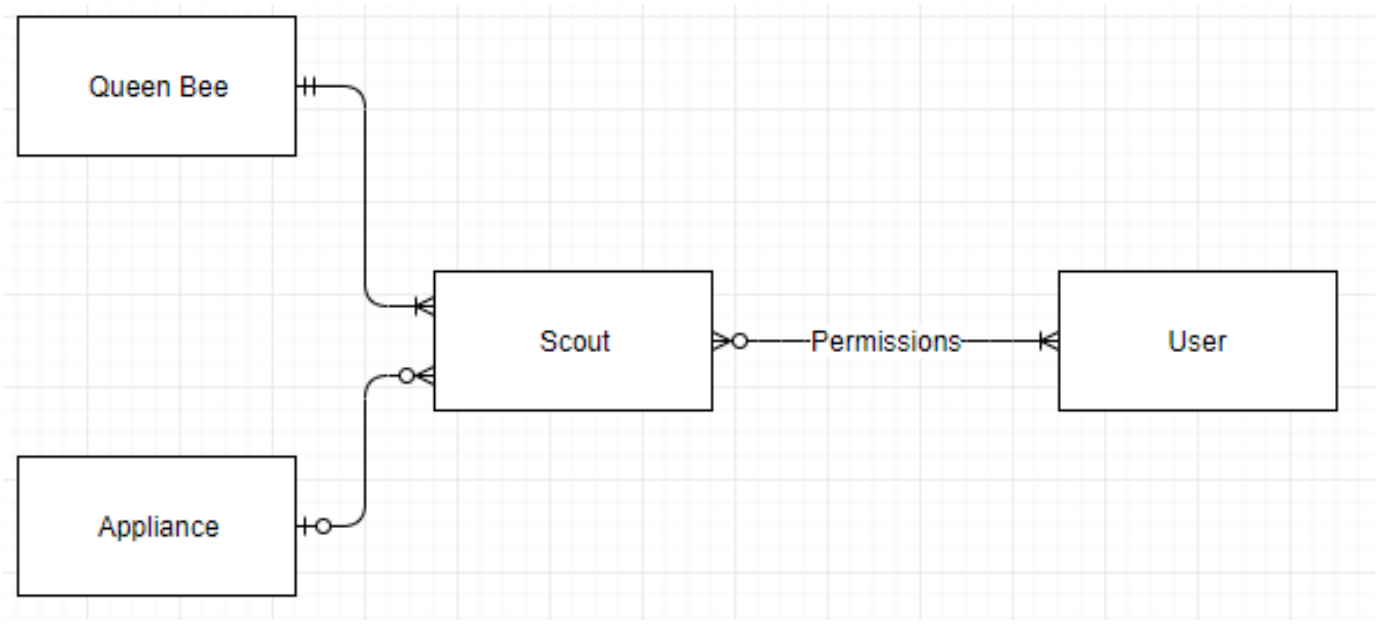

Fig. 1. The Entity-Relationship (ER) diagram of the database.

4) HTTPS with SSL/TLS encryption: HTTPS protocol provides a layer of security on which HTTP lacks. HTTP is vulnerable to eavesdropping and man-in-the-middle attacks. HTTPS mitigates these risks with SSL certificates and encryption.

5) User Authentication: User authentication will be done on the ENGR server, for both a mobile and web application development platform. Scouts will be registered under users, and signing in will give users access to only the devices and the information they provide registered under that user.

C. Documentation

All functions will have comments to appropriately describe (at the bare minimum) the inputs and outputs of the function, or what it does. Functions that exceed a full screen will need to be shortened or broken into multiple parts.

D. Interface

Users will be able to view appliance data through both a mobile device web application and a website.

1) Website: The website will be developed using Flask to provide an intuitive, easy-to-navigate interface. Flask is a web framework to develop websites using the Python language. The website will first require a user to sign in. If they do not currently own an account, there will be an option to register for one. After a user has signed in, the browser page will display all scouts registered under that user and an overview of the information that scout is providing. This overview page will display each scout, what appliance that scout is attached to, and the on/off status of the attached appliance. Users can click on any scout on this page to see its battery life as well as a history of the scout's information reports. From each scout's individual page, its information can be changed including what appliance it is attached to. The scout history page will display what the scout reported at every time interval that it sent information to the database. The website will also have a page to allow for users to change their credentials, and a button for adding more scouts. To add a scout via the website, a user account must first be created, then the "Add a scout" button must be clicked. This will bring the user to a page requesting the serial number on the scout. After correctly inputting the serial number, the user can configure the information for the attached appliance of the scout. The scout must already have been connected to the internet via pairing it with a mobile device such as a smartphone.

2) Mobile Web Application: We will be developing a mobile web application so that the information can be seen and viewed on from the phone as well. The solution to supporting mobile web application may be as simple as using Bootstrap.

E. Database Management

All information will be stored on a MySQL database hosted on the ENGR server. Information on Scouts and Queen Bees, and user permissions and appliance data will all be stored here. More information about the entity-relationships and schema of the database can be seen below. The database will also keep a log history of all events where the status of an appliance was updated, so any event in the past can be viewed.

Entity: User

Attribute: User ID	Attribute: User First name	Attribute: User Last name	Attribute: SMS Phone number
1	"Ben"	"Martin"	503-555-9999
2	"Don"	"Heer"	541-555-1111
3	"Tristan"	"Jong"	null

Entity: Permissions

Attribute: Permissions ID	Attribute: Name
1	View
2	Edit
3	Owner

// Note that many-to-many relationships require a special table.

Relationship: User / Scout (Many-to-many)

User ID	Scout ID	Attribute: Permissions ID
1	1	1
1	2	2
1	4	3
2	2	2
2	3	2
2	5	2
3	1	1
3	4	1

Fig. 2. Database Schema: User, permissions, and User/Scout entity tables.

F. Functionality

The system will support Google Home smart home technology, as well as allow the user to scale to add more appliances to the network. Adding new appliances to the network should be simple and hassle-free.

III. Verification

We will test the project in several ways: through web automation tools, with Python Selenium, and additionally through testing the web API with Postman. Postman is a web testing framework to test standard RESTful API calls to the web using standard HTTP verbs like GET, POST, DELETE. We can write unit tests in both frameworks to test and debug the web interface as much as possible. After adding support to mobile, we will also need an Android emulator to test our product further here: https://developer.android.com/studio/run/managing-avds. We will be spending time coding and debugging in all these testing environments, which is a time-consuming process. We have set aside ten days to build the initial test suite to test the barebones server and fix where necessary, and an additional six days for testing. After building the complete product, we have also set aside more additional time for building new tests on the Android virtual machine and tweaking where necessary.

IV. Project Constraints

We have a hard nine month deadline for this project. We also have a fixed budget. The budget has been set to $300, as it was in the previous year. We will appropriate three quarters of the budget ($225) to the ECE team, and one quarter of the budget ($75) to the CS team.

V. Project Timeline

Entity: Scout

Attribute: Scout ID	Attribute: Appliance ID	Attribute: Name	Attribute: Battery power	Attribute: Queen Bee ID
1	1	"Scout1"	99%	1
2	2	null	100%	2
3	1	null	50%	1
4	3	"MyScout"	30%	3
5	4	null	5%	1
6	5	null	1%	1

Entity: Appliance

Attribute: Appliance ID	Attribute: Name	Attribute: Wattage Usage
1	"MyAppliance"	10Kw/Hr
2	"Oven"	15Kw/Hr
3	"GE Fridge"	20Kw/Hr
4	null	25Kw/Hr
5	"Coffee Pot"	130Kw/Hr

Entity: Queen Bee

Attribute: Queen Bee ID	Attribute: Name
1	null
2	"Eye of Mordor"
3	null

Fig. 3. Database Schema: Scout, appliance, and queen bee entity tables.

Entity: Event Log

Attribute: Event ID	Scout ID	Attribute: Event details
1	1	[Object]
2	2	[Object]
3	4	[Object]
4	2	[Object]
5	3	[Object]
6	5	[Object]
7	1	[Object]
8	4	[Object]

Fig. 4. Database Schema: Event log entity table.

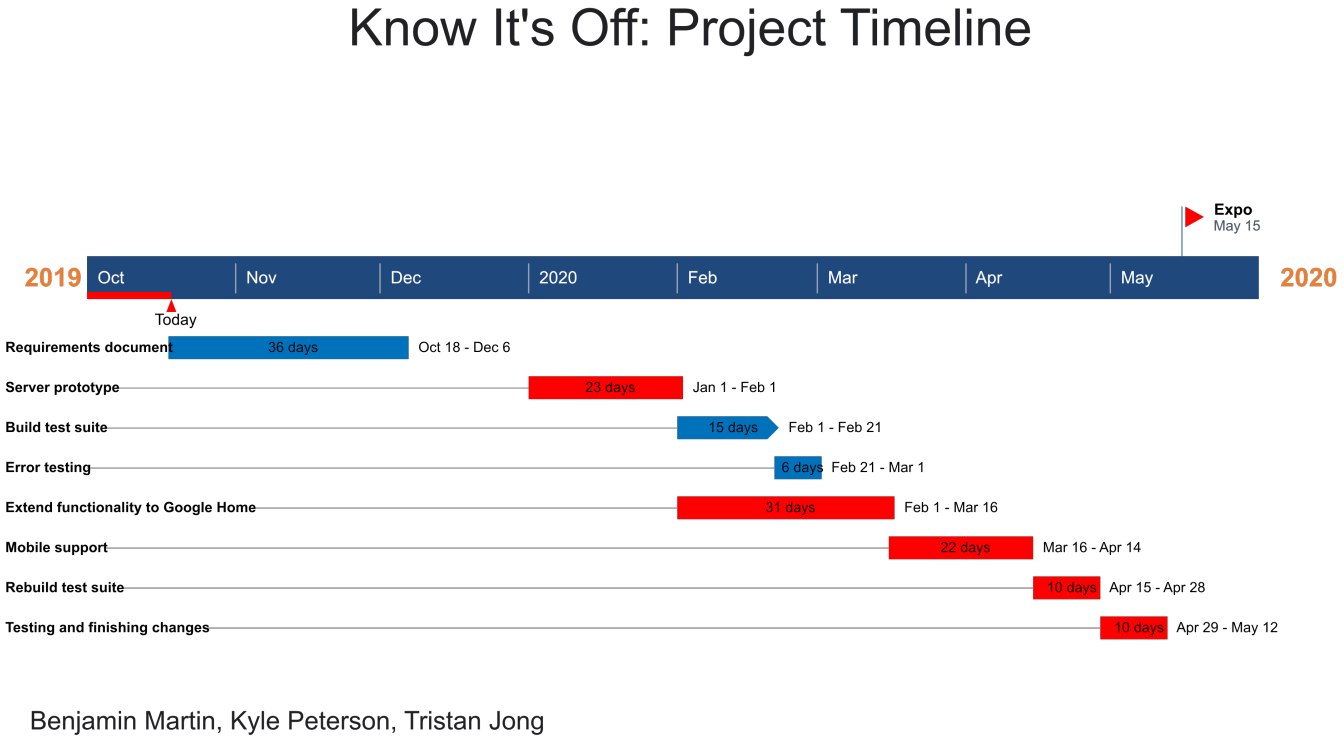

Fig. 5. A simple schedule of the Know It's Off project. Project testing will be reliant on the ECE team's completion of their respective portion of this project.