Inventory Locator Assistant

Team: Alejandro Buentello, Caleb Gehris, Jacob Linch, Kurt Markham, Erin Power, and Chris Rice
Client and Advisor: Mr. Leland Harker

Project Vision

What: A location system allowing employees to find requested parts quickly.

Benefits: Employees will not spend as much time searching and students will spend less time waiting and more time on their projects.

What we expect the cabinets in the ETG to look like after we have added our finished product.

Requirements

Functional and Non-Functional Requirements

Functional

- Accept voice or text inputs
- Allow for adjustments of LEDs
- Allow for “last search”, “all off”, and “all on” commands
- Have test routines
- Use visual motion to direct user attention
- Ability to search for multiple parts

Non-Functional

- Allow for expansion
- Use database outside of product
- Efficient run-time
- Authentication when communicating between LEDs and applications
- Easy to understand documentation
- Easy to use
- Resistant to mild physical impact

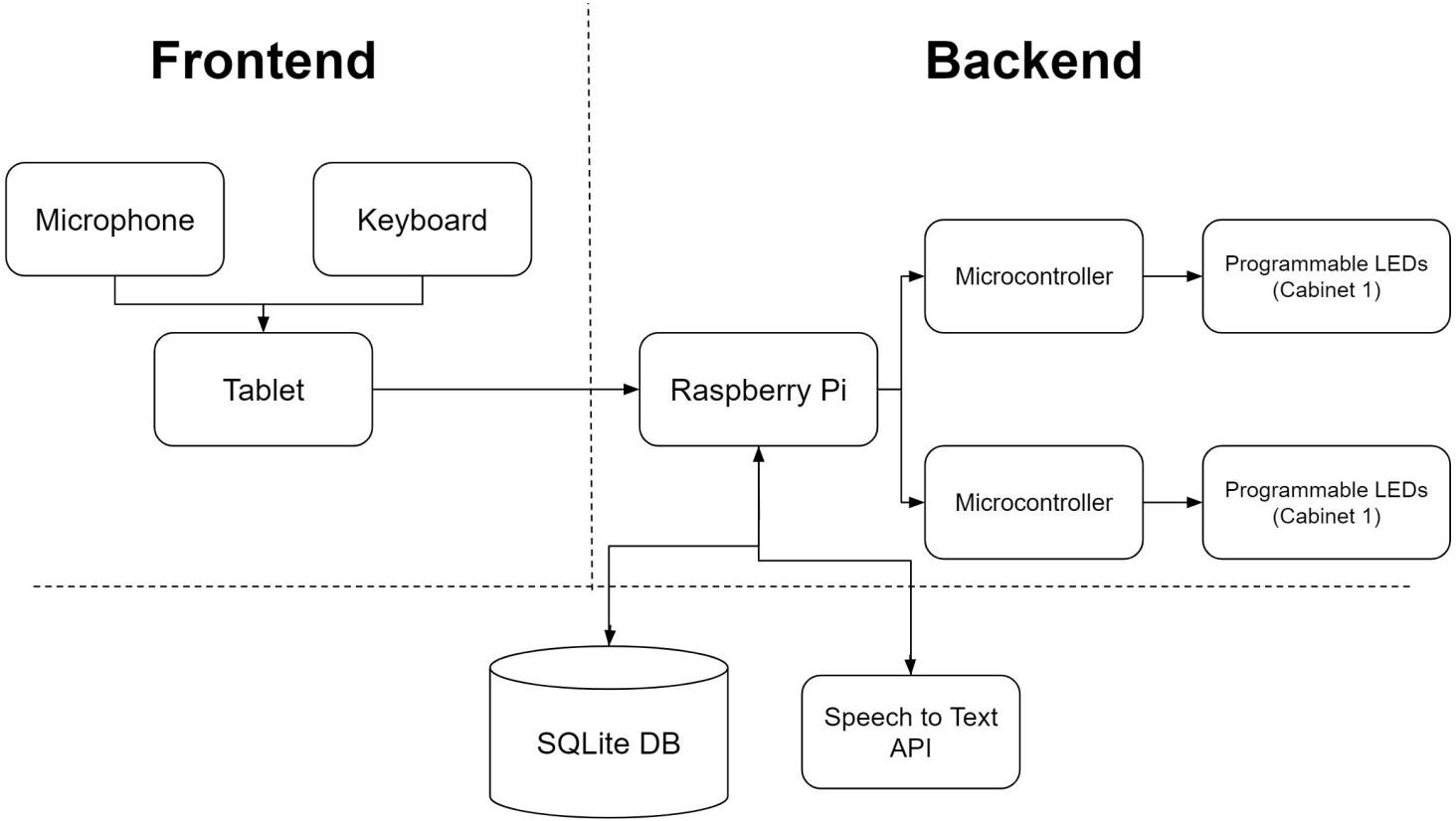
Technical Requirements and Constraints

Technical Requirements:

- LED strips shall fit neatly onto cabinets
- Connect internet devices to the iastate network
- Database must include all existing items in cabinets
- Modification to the database must be easy to perform
- Must be continuously connected to Wifi

Constraints

- Budget of 500 USD
- Operate continuously during ETG business hours
- Used only within the ETG by ETG employees
- Easy for new employee or freshman to use
- Connected to only one tablet



Block diagram of what will be needed to finish our product.

Project Plan

Tasks

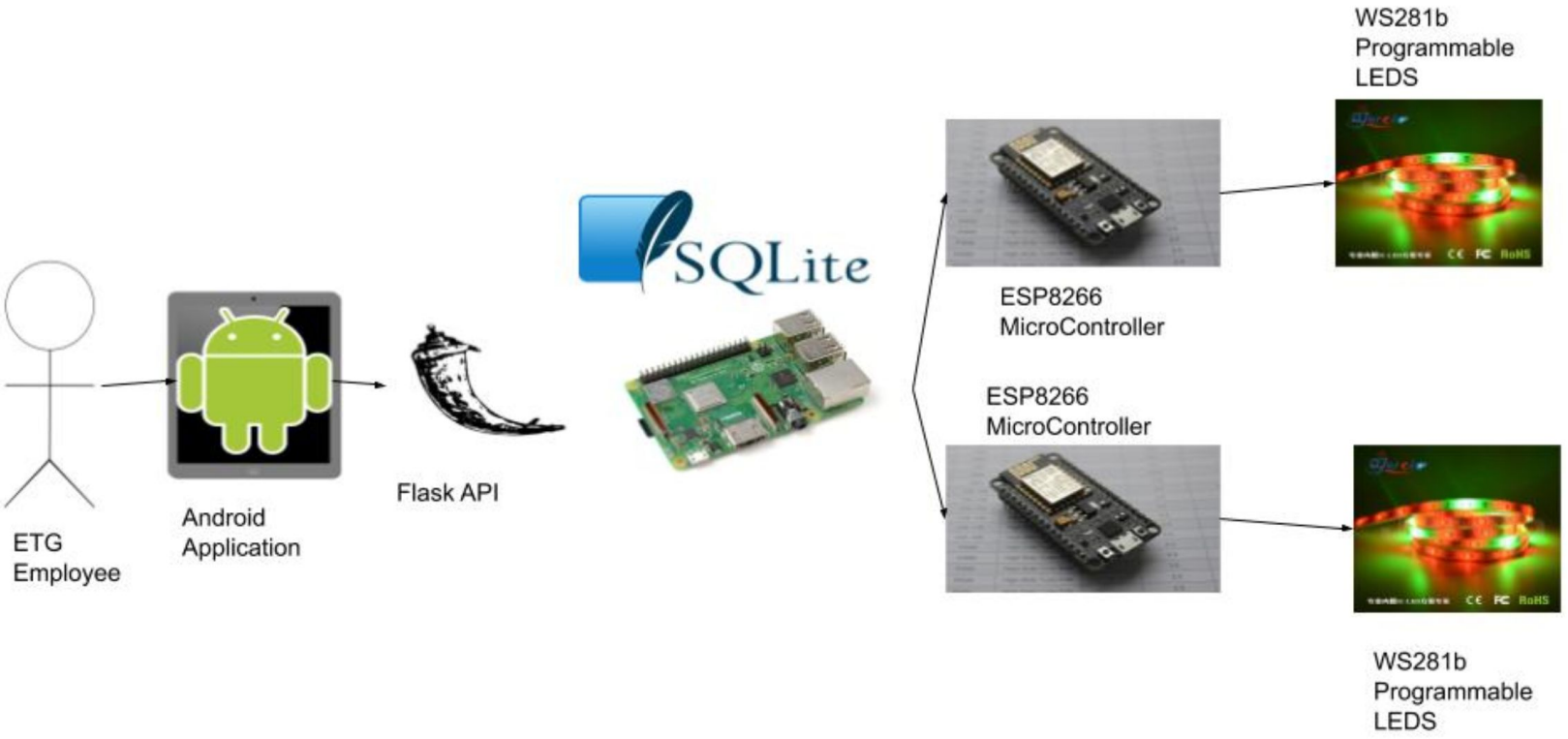
- Documentation
- Hardware and software setup
- Raspberry Pi
  - Communicate with LEDs via Wi-Fi
  - Receive input via Wi-Fi
  - Database (test with dummy data)
- LED Strips and ESP8266
  - Connect ESP8266 to Wi-Fi
  - Receive commands on ESP8266 via Wi-Fi
  - Address individual LEDs
  - Implement special LED functions
- Tablet Application
  - Basic Functionality
  - Construct UI
- Test components, intersystem operations with prototype
- Teach users after final implementation

Risks

Some parts will need to be entered manually which has the possibility of human error.

We might overburden the Raspberry Pi with the size of the database.

System Design



Detailed Design

Platforms

Database - SQLite Hosted on Pi

API - Flask Hosted on Pi

MicroController -

- ESP8266WiFi
- WiFiUdp
- FastLED Arduino Library

Prototype Implementation

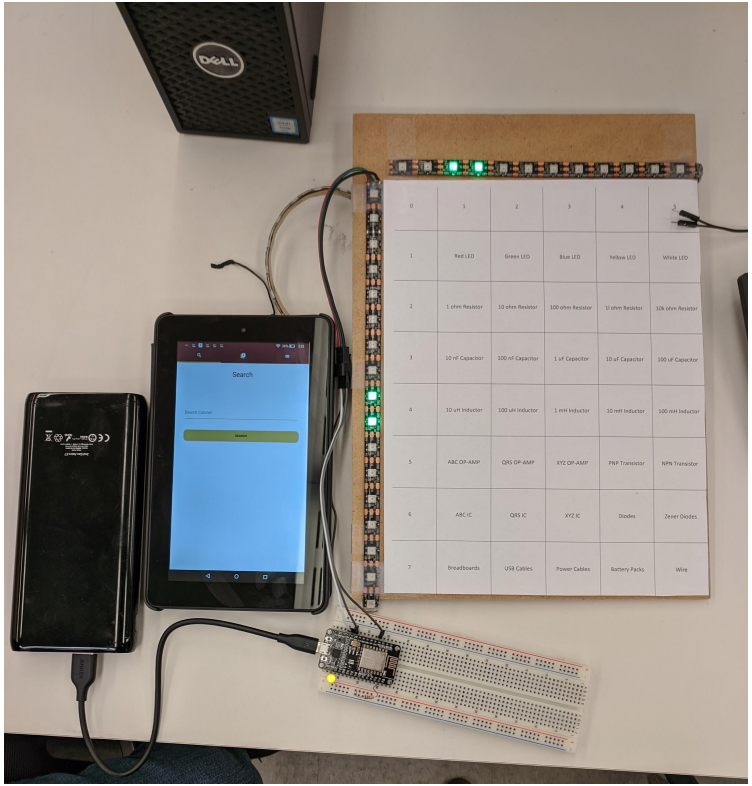
Description of Prototype

Small scale representation of end product

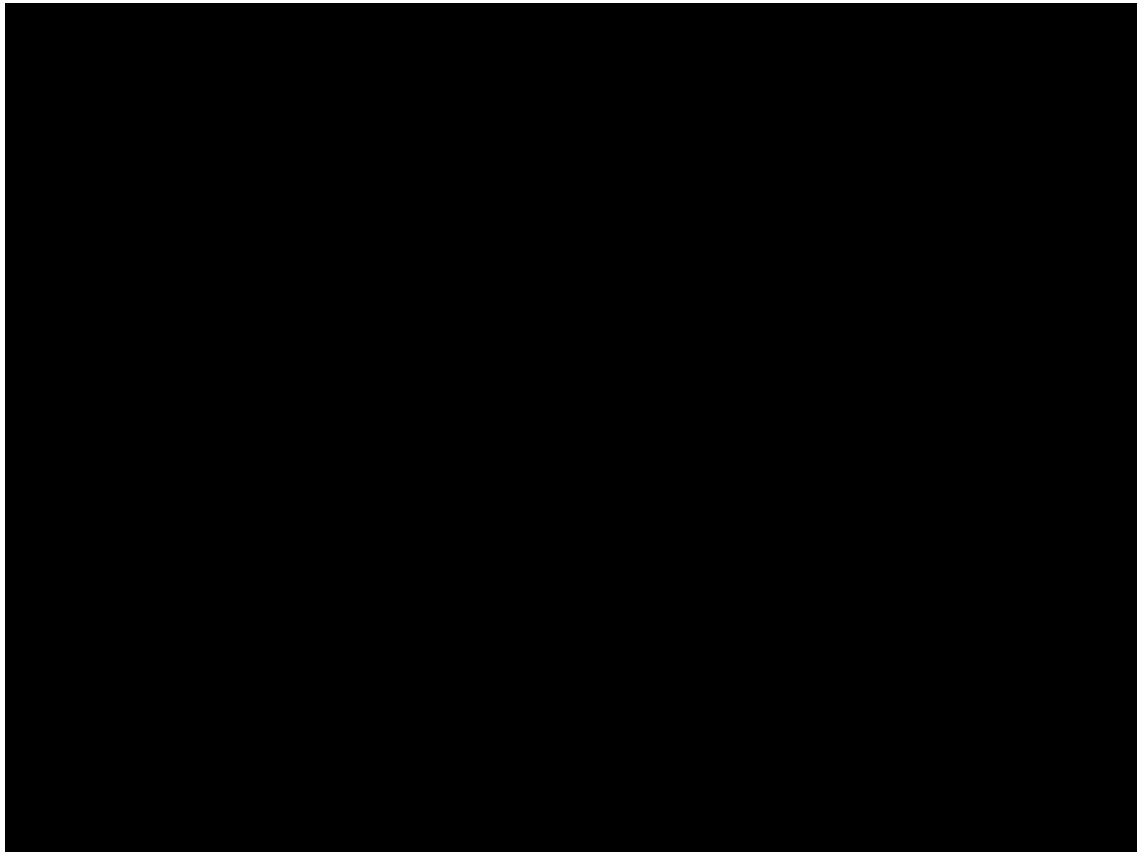
System setup very similar to end product

Has basic search and add functionality

Has mock database to test inputs and outputs



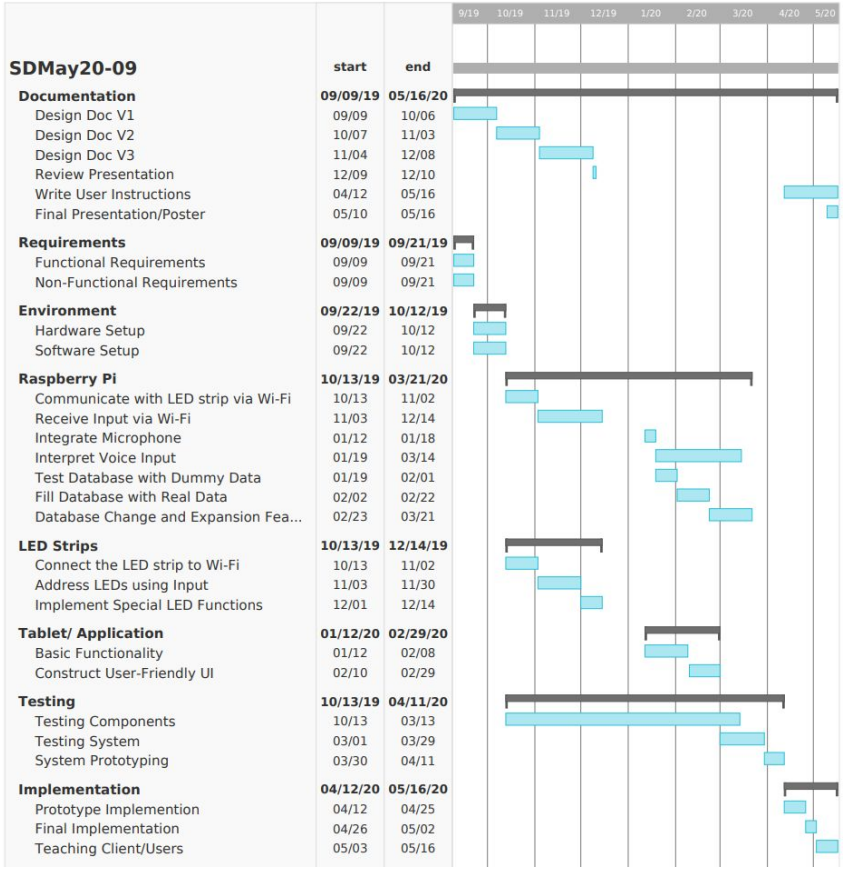
Setup of prototype



DEMO

Schedule

End of Fall and Plan for Spring



SDMay20-09	start	end
Documentation	**09/09/19**	**05/16/20**
Design Doc V1	09/09	10/06
Design Doc V2	10/07	11/03
Design Doc V3	11/04	12/08
Review Presentation	12/09	12/10
Write User Instructions	04/12	05/16
Final Presentation/Poster	05/10	05/16
Requirements	**09/09/19**	**09/21/19**
Functional Requirements	09/09	09/21
Non-Functional Requirements	09/09	09/21
Environment	**09/22/19**	**10/12/19**
Hardware Setup	09/22	10/12
Software Setup	09/22	10/12
Raspberry Pi	**10/13/19**	**03/21/20**
Communicate with LED strip via Wi-Fi	10/13	11/02
Receive Input via Wi-Fi	11/03	12/14
Integrate Microphone	01/12	01/18
Interpret Voice Input	01/19	03/14
Test Database with Dummy Data	01/19	02/01
Fill Database with Real Data	02/02	02/22
Database Change and Expansion Fea...	02/23	03/21
LED Strips	**10/13/19**	**12/14/19**
Connect the LED strip to Wi-Fi	10/13	11/02
Address LEDs using Input	11/03	11/30
Implement Special LED Functions	12/01	12/14
Tablet/ Application	**01/12/20**	**02/29/20**
Basic Functionality	01/12	02/08
Construct User-Friendly UI	02/10	02/29
Testing	**10/13/19**	**04/11/20**
Testing Components	10/13	03/13
Testing System	03/01	03/29
System Prototyping	03/30	04/11
Implementation	**04/12/20**	**05/16/20**
Prototype Implemention	04/12	04/25
Final Implementation	04/26	05/02
Teaching Client/Users	05/03	05/16

Test Plan

Hardware Test

- Manually check the 2 output pins during deployment
- Web server for each cabinet's microcontroller to facilitate testing
- Run a full sweep of both X & Y LED Strip arrays
  - Test any defective LED
  - Test data input by changing colors
- Be able to test all cabinets individually from backend

Software Test

- Unit Testing
- Regression Testing
- Manual Testing
- Continuous Integration

In conclusion ...

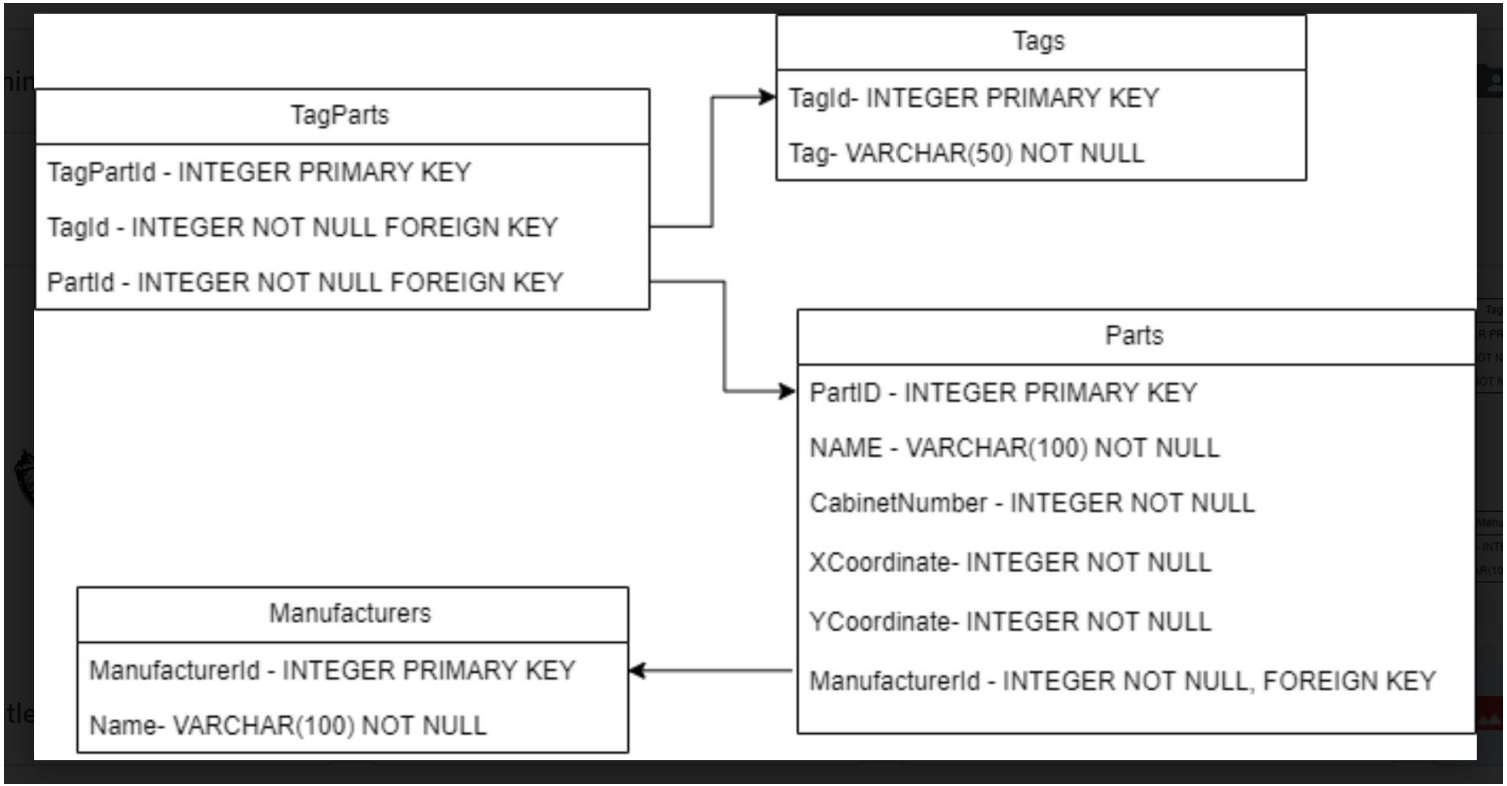
Where are we now?

Finished a small prototype with a search capability of 35 dummy items using an initial tablet application.

How does next semester look?

Next semester will be a focus on software, especially on the database and voice search.

DB Schema


Tags
TagId- INTEGER PRIMARY KEY
Tag- VARCHAR(50) NOT NULL
TagParts
TagPartId - INTEGER PRIMARY KEY
TagId - INTEGER NOT NULL FOREIGN KEY
PartId - INTEGER NOT NULL FOREIGN KEY
Parts
PartID - INTEGER PRIMARY KEY
NAME - VARCHAR(100) NOT NULL
CabinetNumber - INTEGER NOT NULL
XCoordinate- INTEGER NOT NULL
YCoordinate- INTEGER NOT NULL
ManufacturerId - INTEGER NOT NULL, FOREIGN KEY
Manufacturers
ManufacturerId - INTEGER PRIMARY KEY
Name- VARCHAR(100) NOT NULL

RESTful API

Base URL: ilapi.ece.iastate.edu:5000

URL	Method	Description	Request Payload	Response PayLoad
/parts	GET	Returns all Parts in the Database. Main use case is testing	N/A	[{ "CabinetNumber":<int> "ManufacturerId":<int> "Name":<string> "PartId":<string> "XCoordinate":<int> "YCoordinate":<int> }, {...}]

/	POST	Adds a Part to the database, If the manufacturer isn't added yet, this adds one	{
			"CabinetNumber": <int>
			"XCoordinate":<int>
			"YCoordinate":<int>
			"Name": <String>
			"Manufacturer": <String>
			}
				"success"

/find/name	POST	Find a part with the given name and illuminate the LEDs of its location	{
			"Name": <String>
			}
				"success"