PIT Tracker

Dr. Chen: Senior Project I

Nathan Newcomer: Computer Engineering Technology

Caleb Cordis: Computer Science

Jeremy Bugay: Computer Science

Ian McGee: Computer Science

Instructor Comments/Evaluation

Table of Contents

Abstract

This paper is designed to be read by the software developers of PIT Tracker. It will lay out the development, side designs, and requirements of the device. The PIT Tracker will send a GPS signal, which will be converted into a form legible by the Web Client. Signal will then get sent to the web client, which will then update the map of the store where the machine is located. The developers can use the information provided in this document to gain insight on their best process to handle the project. This document should provide adequate information for the formation of project source code. The details list the interactions between modules of the code, hardware that will be implemented, server environment, as well as what programming language will be used.

Description of the Document

Purpose and Use:

The purpose of this document is for the developing team's overall guidance in understanding the foundation of the software project. This will be written in detail such that the developers will know the functionality and language and how the hardware will work in tandem. The purpose of this document is to levy everyone's abilities so that the project can remain on track without any deviations.

Ties to the specification document:

This document is an extension of the materials inside the specification document. It adds in more detailed components for the software developers such that they will know the agreed upon language the program will be written in, how it interfaces with the hardware, and what type of software development style will be implemented.

Intended Audience:

The primary audience of this document is the software developers due to the nature of the material discussed herein. This document is designed exclusively for the development team, however, the client team may find some useful information to be extracted if they have within their department someone who is familiar with the software development side. This can be beneficial to the clients for a better understanding of the final project and their requests. This document however, is to help the developers understand the scope and nature of PIT Tracker, including it's functionality, design, process flow, modularity, hardware and system design and requirements.

Project Block Diagram w/ Description

The PIT Tracker project will consist of three parts: PIT Tracker device that consists of the Arduino controller, frontend code that will be the website that consists of JavaScript, and backend code on a Linux or Windows server. The Arduino will interact with the backend server by sending the PITs location to the server over the network. Once the backend has received the location, it will then process the locations coordinates and update the website with the coordinating location. Each PIT is labeled with a color-coded dot that distinguishes between each machine.

From there the employee is able to access the website from any workstation in the store.

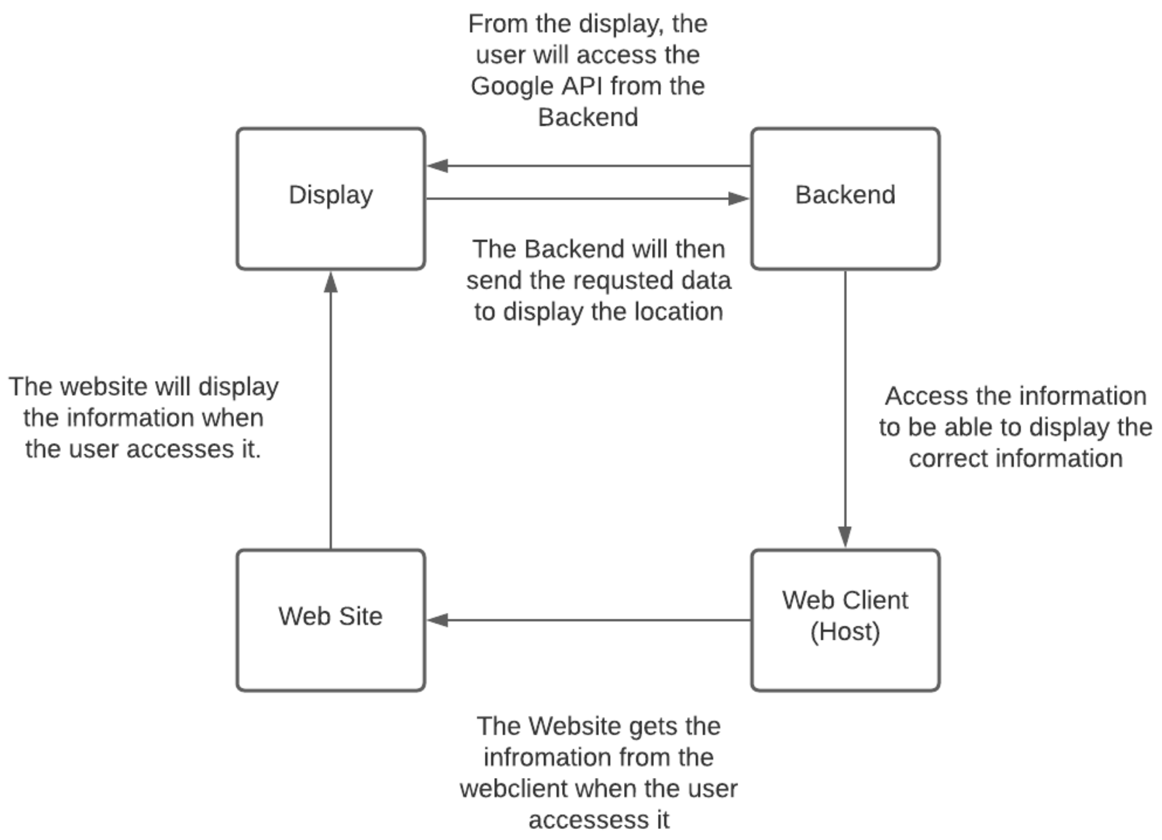
From the display, the user will access the Google API from the Backend
Display
Backend
The Backend will then send the requsted data to display the location
The website will display the information when the user accesses it.
Access the information to be able to display the correct information
Web Site
Web Client (Host)
The Website gets the infromation from the webclient when the user accessess it

Figure 1: Block System Diagram

Design Details

System Modules and responsibilities:

- Architectural Diagram

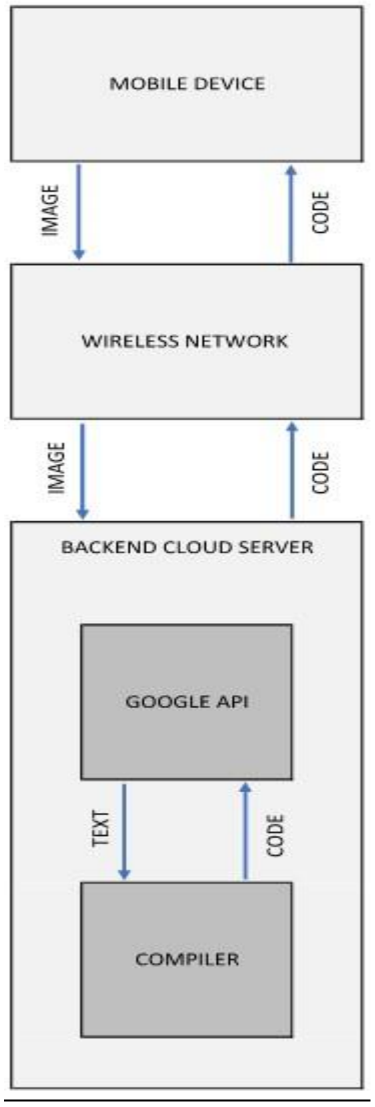

Figure 2: Architectural Diagram

Figure 2 Description

The PIT Tracker device will transmit data to the backend server. The server will then convert the coordinates that it receives from the PIT Tracker. The image data will be relayed over a wireless network and received by the backend server. The backend server contains the Google API for the store map, a program that is tasked with converting the raw data that is sent from the PIT Tracker and will be the website host. The compiled code will be returned to the mobile device over the wireless network.

System Models and Responsibilities

- System Overview (Where it is implemented)

PIT Tracker will provide retail employees working in a rapid pace climate with the ability to quickly access PIT Tracker. The Web Client will be available on different computers. It relies on the end user in the store to be able to access the web applications. The U.I will be simple and intuitive to understand and will show a legend so there is no confusion of what is what so the end user is not confused. The only thing the user will be required to do is logically read the map and know where all the PIT Tracker devices are located. The PIT Tracker we located. The PIT Tracker will automatically upload it's final location in real-time for the next employee who will need to access it. The Main components of this application will be a web host. The Arduino GPS will automatically upload it's coordinates to the web application and front end server. The back end server will do all the work and calculations to create an enjoyable user experience.

- Module Cohesion

  Our project will express module cohesion with functions from many classes. The GPS class will be able to upload the coordinates from the PIT tracker. The Receive Class will then convert these coordinates and then upload them and store them so that the Map Class will display this information for the User Class. These functions will work independently of the others but will n relay the return values to their respective places. The Web Client class will work as the backend server and communicate and store all of the data needed for the backend server.

On the front end side, The User Class, the Update Class, and Labeling Class along with the Map and Legend Class will use the data stored from the backend server for the user.

- <u>Module Coupling</u>

  When using a gps chip, a data logger is needed to be coupled together with the chip, this is how the gps's data is logged to be sent. The backend server will be coupled with the web client. This will include all our sorted data, and the map on the webpage. Each of these will have their own specific purpose, although it will be working together towards the same goal of displaying the location of the PIT. Our map will be run using a standard Google API, which will be updated every time it receives a signal. The receive and Send classes are two parts working to accomplish the same task and will work codependently of each other.

<u>Design Analysis</u>

- <u>Data Flow</u>

  The dataflow of PIT Tracker is handled by the send and receive classes that pass along the converted NMEA strings. The user activates the application in the workspace, the signal will already be sending after installation, so activation is not required. Once the server has received the strings. The data will be processed by the Web Client and Map classes, and the location of the PITS will be updated. The user then can access details about the PIT's location.

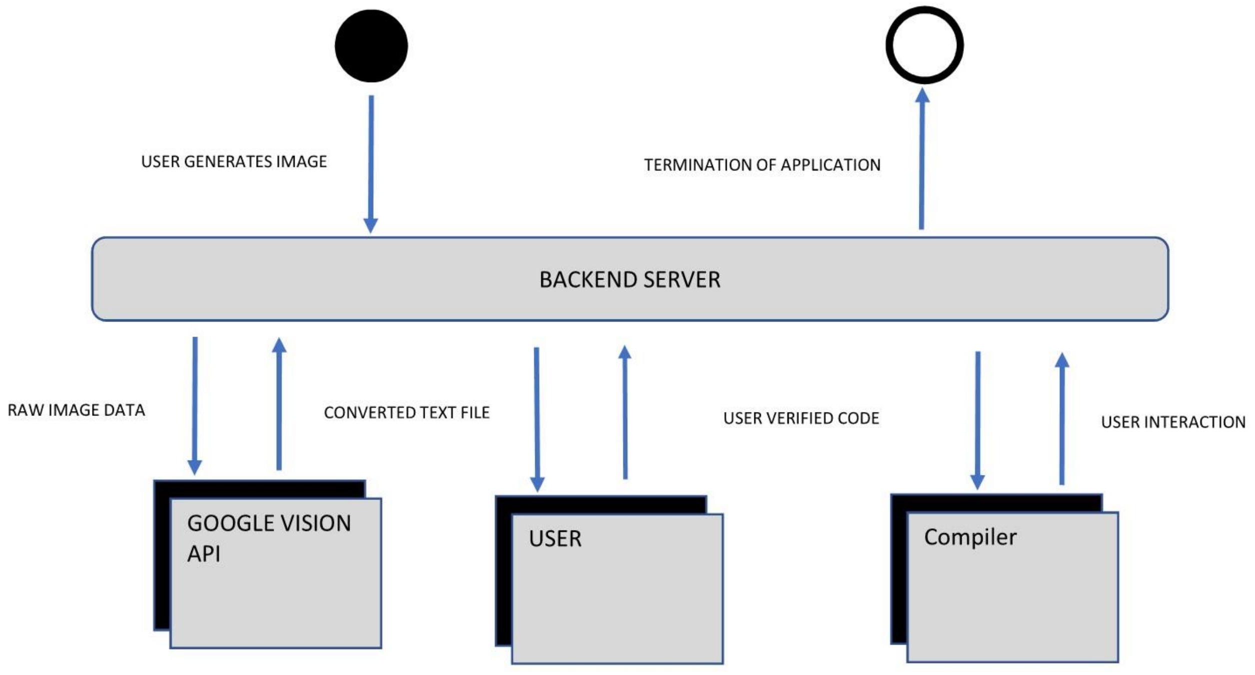

Figure 3: Data Flow Diagram

Design Organization (Object Oriented Design)

Detailed tabular description of Classes/Objects

Class name: GPS (Send)

Class description: The GPS (Send) class is a boundary that will send the coordinates of the tracker to the web client. This includes the whole PIT Tracker as the GPS unit needs to be connected to the Arduino to report the coordinates back to the web client.

Class data members: GPS Object, PIT Object , Coordinate

Class member functions: sendCoordinate()

Class name: Receive

Class description: The Receive class is a boundary that will receive the data

Class data members: Receive Object

Class member functions: showReceivedCoordinate(), convertReceivedCoordinate(), sendLocation()

Class name: Map

Class description: The Map class is the layout of the store. The map will have several different locations set to report to the web client. Each PIT will be color coded within a legend at the top of the map to distinguish between each PIT.

Class data members: Map Object, GPS Object

Class member functions: showMap(), closeMap(), addMessage()

Class name: Web Client

Class description: The Web Client class is responsible for hosting the website for PIT Tracker. The web client receives the information from PIT Tracker and then converts the coordinates to a location on the store map.

Class data members: Map Object, GPS Object, Web Host

Class member functions: connectionHost(), openConnection(), closeConnection()

Class name: Labeling

Class description: The Labeling Class will take care of labeling and distinguishing between each PIT. In order to do this, there are 6 different PITs. Those are the Reach, Stock Picker, Pacer, Small Sitdown Forklift, Large Sitdown Forklift, and Electric Pallet

Jack. Each PIT will be represented by a colored circle on the map. The map will display each PIT location by this colored circle. The labeling will be as follows: Reach will be a red dot, Stock Picker will be blue, Pacer will be Yellow, Small Sitdown Forklift will be green, Large Sitdown Forklift will be orange, and the Electric Pallet Jack will be purple. A legend will be displayed on the map informing the user on how to distinguish between the PITs.

Class data members: User Object

Class member functions: displayMap(), displayLegendOverlay()

Class name: User

Class description: The User class is the end user end of the operation. The end user will be able to access the website via URL address (in the future hyperlink on each workstation) to access the PIT Tracker map.

Class data members: User Object

Class member functions: displayMap(), displayLegendOverlay()

Class name: Update

Class description: The Update class is the controller class that handles taking the coordinates that the GPS class sends to the web host and updates the map. This way whenever an employee opens the web page on a workstation, the location will be updated with its latest location. Ideally, the update class should update every minute.

Class data members: PIT object

Class member functions: locationUpdate()

Functional descriptions

GPS (SEND) Class

sendCoordinate()

Input:

The sendCoordinate() function will receive data from the GPS Satellite to determine the coordinates of the PIT.

Output:

The function will send the coordinates of the PIT to the backend server.

Return Parameters:

The only values returned are any exceptions thrown during error checking.

Types:

The data types used within this function are based on the coordinates that has been received from the PIT Tracker. This will be an integer and then converted in the receive class.

Receive Class

showReceivedCoordinate()

Input:

The showReceivedCoordinate() function has no input.

Output:

The output of this function will show the coordinate on the backend server

Return Parameters:

The only values returned are any exceptions thrown during error checking.

Types:

The data types used within this function are based on the built-in Arduino GPS Tracker

convertReceivedCoordinate()

Input:

The convertReceivedCoordinate() function gets input from the showReceivedCoordinate() function

Output:

This function will convert the coordinates to a location in the store via store map.

Return Parameters:

The only values returned are any exceptions thrown during error checking.

Types:

The data types used within this function are based on the built-in Arduino GPS Tracker

sendLocation()

Input:

The sendLocation() function has no input.

Output:

This function will send the PITs location to the Map Class

Return Parameters:

The only values returned are any exceptions thrown during error checking.

Types:

The data types used within this function are based on the built-in Arduino GPS Tracker

Map Class

showMap()

Input:

The showMap() function has no input.

Output:

This function will output the map when the webpage is accessed

Return Parameters:

This function will return the success or failure of the MAP Displaying on the webpage

Types:

The data types used within this function are based on the built-in Arduino GPS Tracker

closeMap()

Input:

The closeMap() function has no input.

Output:

When the website is closed, the program should close down the map.

Return Parameters:

This function will return the success or failure of the map.

Types:

The data types used within this function are based on the built-in Arduino GPS Tracker

addMessage()

Input:

The addMessage() function has no input.

Output:

This function will display a message on the map where each PIT is located.

Return Parameters:

The only values returned are any exceptions thrown during error checking.

Types:

The data types used within this function are based on the built-in Arduino GPS Tracker

<u>Web Client Class</u>

connectionHost()

Input:

The connectionHost function has no input

Output:

The connectionHost() function will be the function that makes the connection from web client to the backend server.

Return Parameters:

This function will return success or failure of the recognition procedure, received by the server, and any other error checking.

Types:

The data types used within connectionHost() are string, integer and boolean data types.

openConnection()

Input:

The openConnection() function will receive a request when the user opens the
webpage

Output:

The openConnection() will output the webpage if the connection has
successfully connected.

Return Parameters:

openConnection returns the success or failure code of the function.

Types:

The openConnection() function will just be opening the appropriate socket to allow a connection to the client.

closeConnection()

Input:

The closeConnection() function will close the connection when the user closes out the webpage.

Output:

There is no output for the getFromRecognizer() function.

Return Parameters:

This function will return if the recognition and transmission was successful or not and any other error checking.

Labeling Class

labelPIT()

Input:

This function does not receive any input.

Output:

labelPIT() labels each PIT with a different color to distinguish between the machines

Return Parameters:

This function will return the success or failure of a message to be sent or any other error checking.

Types:

The data types used within this function will include string, integer and boolean.

User Class

displayMap(),

Input:

This function receives no input.

Output:

This function will display the store map.

Return Parameters:

displayMap() will return a message if an error occured.

displayLegendOverlay()

Input:

This function will not receive any input.

Output:

This function will display the PIT Legend over the map.

Return Parameters:

The function will return any error checking done within.

<u>**Update Class**</u>

locationUpdate()

Input:

No input is required for the locationUpdate() function.

Output:

This will call the receiveCoordinate() from the Receive class the GPS signal has been sent and converted to a location on the store map.

Return Parameters:

locationUpdate() returns the success or failure of the process of editing the code and any other error checking.

Types:

This function will use string and boolean data types within.

webpageUpdate()

Input:

webpageUpdate() will receive the finished map update from the locationUpdate function

Output:

There is no output for this function.

Return Parameters:

This function will return any error checking done within this function.

Types:

The data types used within this function will include integer, string and boolean.

<u>Real-time requirements:</u>

The web client should be able to perform in a relatively fast timeframe. The web client needs to display a store layout and the different PIT nodes, and should be able to do this fast enough to make the service useful for employees within the store. If the application does not display fast enough, this eliminates the time-saving benefit of displaying PIT locations to a user. Also, the rate of updating the GPS locations has to be frequent, if not continuous, in order to ensure accurate location reporting and usability.

Signals/Messages:

The signals and messages sent between the frontend and backend processes will include the following pieces of data:

Signal/Message Type	**Source / Destination**	**Data**
sendCoordinates()	GPS Device <-> Backend	GPS Sends coordinates.
receiveCoordinates()	GPS Device <-> Backend	Receive the coordinates and convert to a location in the store.
sendToUpdate()	Backend -> Web Client	Send coordinates form the backend server.
receiveFromUpdate()	Backend -> Web Client	Received the coordinates from the backend server.
sendToMap()	Web Client -> Backend	Coordinates from Update.
receiveFromMap()	Backend -> Web Client	Recognized coordinates.
recognizeLocation()	API <-> Backend	Location is recognized by the Google API and is reported back to the backend server.

<u>Narrative / PDL:</u>

1. The user will login to their workspace.
2. The user will access the webpage from the workstation.
3. The map will be displayed.
4. The map will show all of the PITs located in the workplace at the time.
5. The user can log out of the interface
6. The Pit is then located and used by the user
7. The GPS chip will automatically send the new location to the server for the next user.

<u>Decision: Programming language / reuse/ portability:</u>

The backend code will be written in Arduino which is very similar to the language that that is C++. Arduino uses many libraries that are similar to what would be in C++. By using Arduino, the development team will be familiar with C/C++ and will be able to adjust to programing through the Arduino IDE. Using the Arduino language and it's IDE, the development team will have a significant decrease in the barrier for entry for programming purposes.

The front end of the program will be written in java-script due to it's ease and structure of allowing the web application to connect to the arduino board. This will be easy for the user to interface with. - side note: will need to research more about JS for this portion of the front end of development.

Implementation Timeline

Below is a timeline of the PIT tracker’s development.

Activity	Jan	Feb	March	April	May	May+
Component Level Design	🟥	🟥				
Software Design		🟥	🟥			
Unit Testing	🟥	🟥	🟥	🟥	🟥	🟥
Subsystem Integration and Verification				🟫	🟫	
System Integration and Verification				🟫	🟫	
System Validation				🟩	🟩	
Operation and Maintenance Manuals				🟩	🟩	🟩
Changes, Upgrades and Retirement					🟩	🟩

Design Testing

Design and testing will be done regularly throughout the lifetime of the project. The team will work on front end section first by writing the code for the Arduino to upload the coordinates. The first logical step is to ensure that the Arduino is capable, in real time, being able to upload it's GPS coordinates to a server. The second phase will be for using the GPS coordinates to be stored in the backend server. The Final stage will be to upload those to the Web Application.

References

Appendix A: Technical Glossary

Arduino- The Arduino Uno is an open-source microcontroller based on the Microchip ATmega328p microcontroller.

Microcontroller- A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system.

GPS- A device used to communicate with satellites to report position on the globe.

NMEA- NMEA is a standard data format supported by all GPS manufacturers.

IDE- A software for building applications that combines common developer tools into a single graphical user interface

Pacer- This is a stand-up forklift. In lieu of sitting down, the operator stands and operates the PIT. This is helpful to get into narrow or tight areas. The forks do not extend on the pacer unit.

PIT machine- Powered industrial trucks such as forklifts or a lift truck

Reach Truck- PIT that is specialized for going down narrow aisles and gripping pallets on either side of the racking in an aisle. To achieve this, the forks extend in and then a retracted in to receive the pallet.

Stock Picker- Or man up is a PIT that is used to grab bulk items from the racks. The operator rides up while the machine is in motion to retrieve the item the customer requests.

Web Client- A webpage used to host the GPS application.

Appendix B: Report from the Writing Center

Cal U Vulcan Learning Commons Report

Client: Ian McGee

Staff or Resource: Sierra M

Date: December 3, 2020, 5:00pm - 6:00pm

What course was serviced by this visit?:

CSC490- Sen. Proj. I: Software Engineering

Did the student request that the instructor receive a visit report?:

Yes

Please provide any additional comments relevant to this session.:

The project seems to be quite efficient!!!

How did the process of this consulting session address the established goals?:

Review the team project/essay to make sure all the rubric requirements were met. The group contributed efficient information for the project. A few suggestions were given dealing with grammar and adding details.