blueHat©

Blue Sheep

Mobile Data Collection Application

CSU San Marcos
Fall 2017 CS 441
Software Engineering

Final Project Report

BlueHat Technologies

Paul Thomas Rowe
Raul Perez
Mike Yoon
Takuro Iwane

ABSTRACT

The project is intended to give our users a peek into the incredible amount of information they unknowingly produce through their phone's various sensors. Our mobile app collects data from each sensor individually, periodically sending it up to a central server. There we use various algorithms to make the data collected comprehensible through the use of charts, giving users an easy way to see how their phone interprets their habits . The web server is capable of storing and parsing through large amounts of user information to get a more extensive understanding for the data being collected. The web server is hosed through Amazon AWS EC2. The mobile application will be simultaneously deployed as an app for the Android and iOS through the use of Xamarin Forms, which allows for the development of both platforms simultaneously, providing faster development and a universal interface.

TABLE OF CONTENTS

PROJECT DESCRIPTION

Modern mobile devices of today are capable of collecting a mind boggling amount of data through their sensors and software. A problem that is particularly present amongst users is that a majority will readily grant any application, program, or piece of software full access and permission to their personal devices, including (but certainly not limited to!) phones, computers, and tablets. By doing so, seemingly benign applications are capable of keeping track of a user's location, movements, activities, and preferences to name just a few categories. "Whether you are going for a run, watching TV or even just sitting in traffic, virtually every activity creates a digital trace" (The Economist). We argue that most users of such mobile devices are entirely unaware that even just through regular use, may allow their phones to collect huge amounts of unique data about their day to day lives.

In a recent article The Economist stated that "The world's most valuable resource is no longer oil, but data" citing that many companies, such as Alphabet (Google's parent company), Amazon, Apple, Facebook and Microsoft, use the data collected from their users to do incredible things. We want our app to shed some light how a single user's data can tell us so much about them and what makes that data just so valuable.

USER REQUIREMENTS

Users will only need a relatively recent Android or iOS smartphone and n internet connection to make full use of our application's features. The internet connection does not even need to be constant or overly reliable, the application will continue to collect sensor data even when offline. Internet is only needed to upload the log files. Upon connecting to an internet source, users will have access to anonymous data collected by themselves other users granting them more features and actions. The user will need to create an account in order for the application to distinguish their data apart from from the anonymous users.

Users will need to have the launch the application at least once for it is capable of running in the background or foreground in order for the data to be collected from the various sensors at all times. If the application is not running, no data will be collected. If users wish to access their data, they will need to launch the app and log in before data will be parsed and diagrams formed.

This application is oriented towards smartphone users who want to know more about what their phone knows about them, or those who don't care about their privacy. Our product will benefit both use cases immensely by giving them concrete evidence of the data being collected, educating them on how their smart devices are being used, and how particular applications are capable of taking pieces of your personal identity without express permission.

USE CASE DIAGRAMS

Diagram #1: Normal Use Case, Full Access to Application

In the first diagram, the user has an internet connection on their android or iOS device. User is able to log into their account, create a new one or retrieve a forgotten password. Once logged in, the user will be able view data collected from the Blue Sheep application on their mobile device through all the currently available methods.

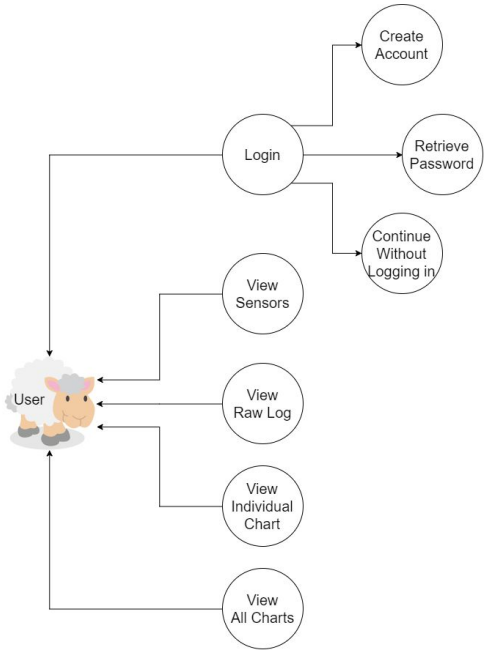


Diagram #2: Offline Scenario

In the second diagram, the user does not have an internet connection on their android or iOS device. Therefore because the device is not connected to the internet, it is not possible for the blue Sheep application to retrieve data authenticate the user. Thus the user may continue without logging in. In this use case the user is still able to view the sensors collecting data and the raw log being created, but are unable to view any form of charts due to a lack of connection to the central server.

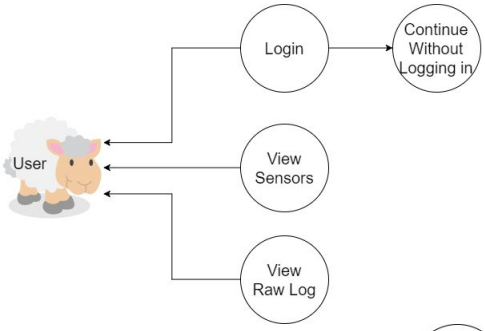
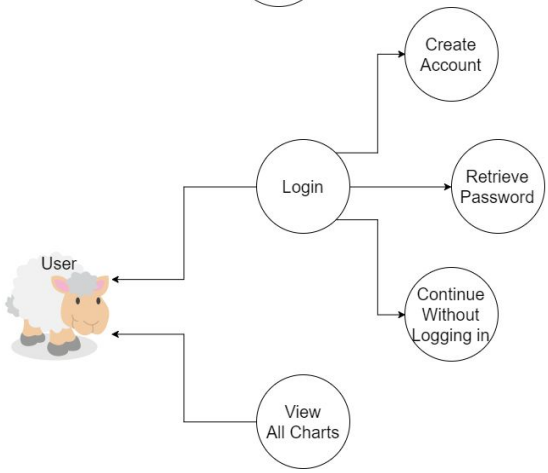


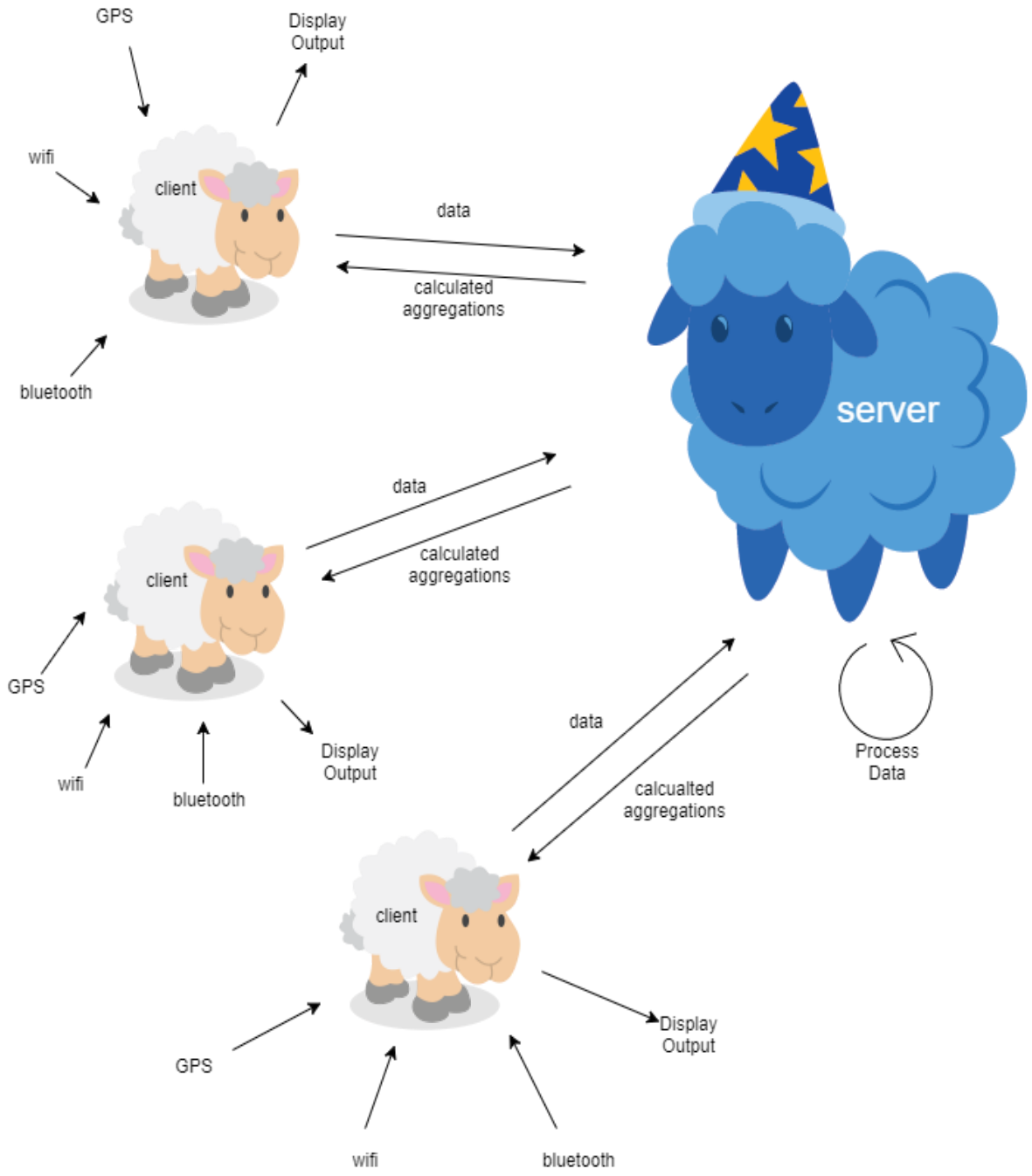
Diagram #3: Sensors Disabled Scenario

In the third diagram, the user does not activate their sensors. They are still able to log in normally, able to take advantage of all options in the login menu. However the user is unable to see the sensors working (because they're deactivated), raw logs (there is no local log to display), or individual graphs. The only form of data visualisation available is the master chart

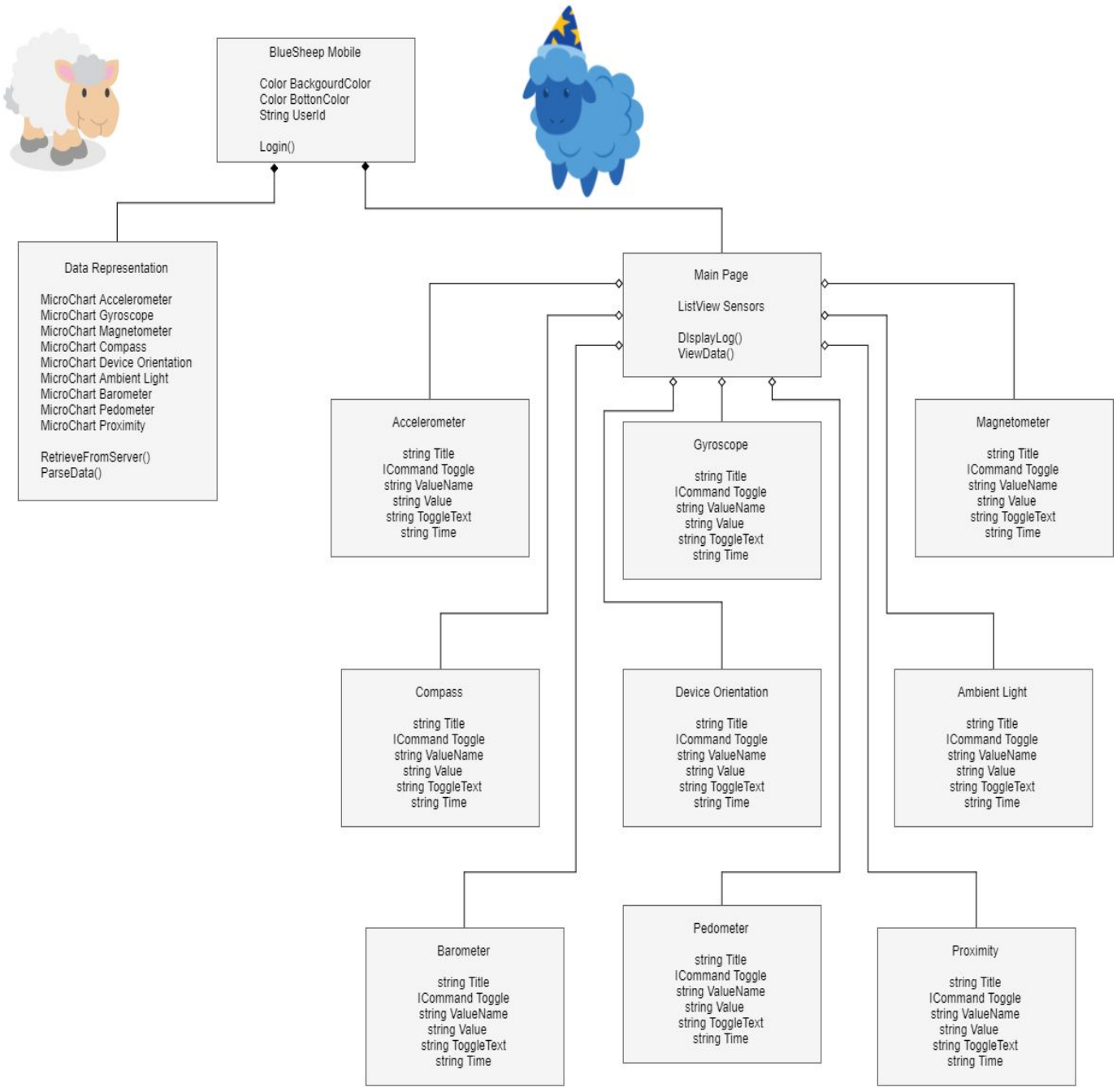


culmination of the all sensory activities of all of our users.

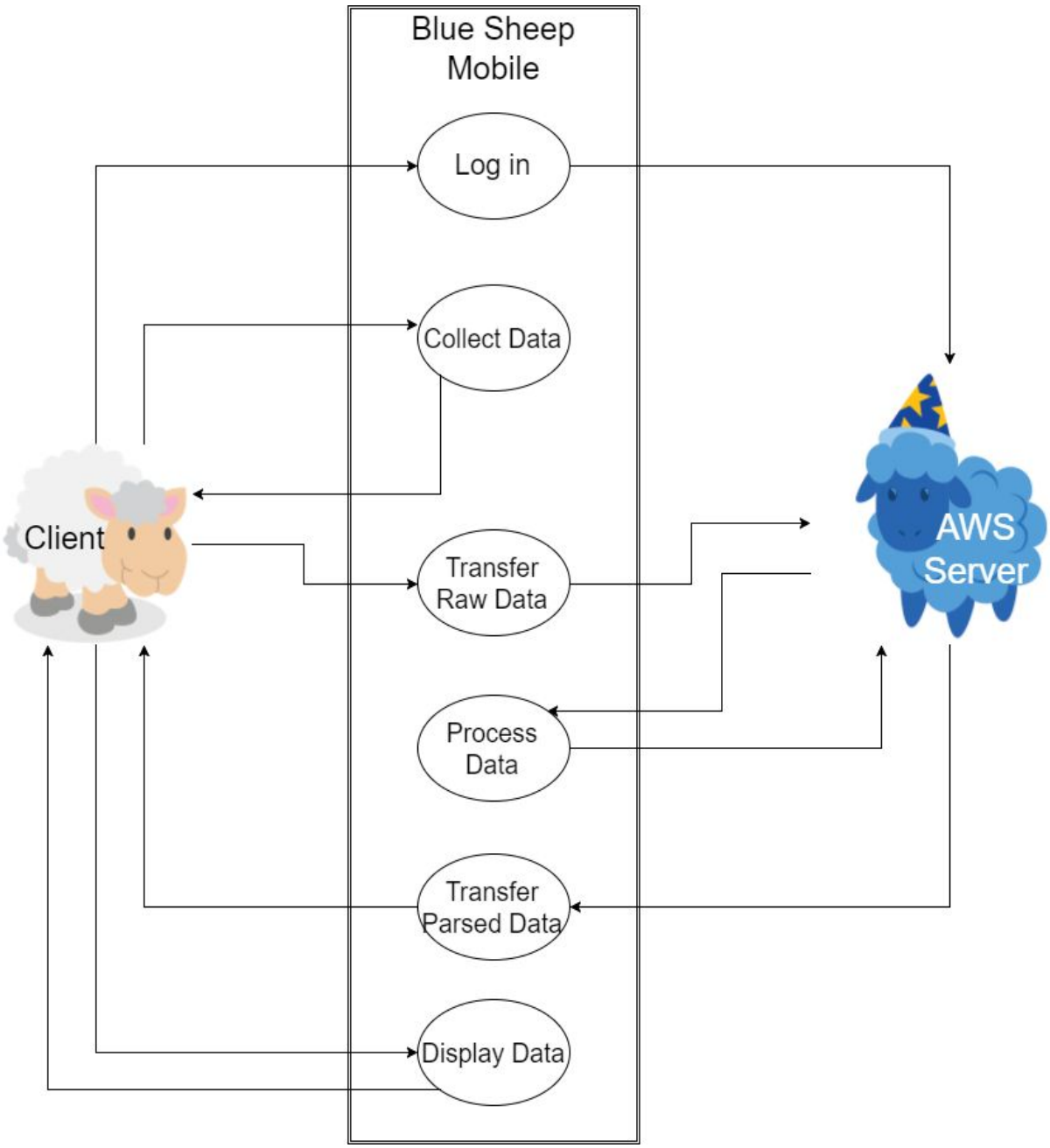
Overall Architecture Diagram of System Design



Class Diagram of Blue Sheep Mobile's Subsystem



Sequence Diagram of a Normal Use Case



IMPLEMENTATION MILESTONES			
Number	Type	Implementation	Accomplished?
1	Mobile	Xamarin Studio (C#)	X
2	Mobile	Detecting Specific Sensors	X
3	Web	Web Server	X
4	Mobile	Log file Creation	X
5	Web	File Parsing	X
6	Web	Visualizations	X
7	Mobile/Web	File Transfer Protocol (SFTP)	X
8	Web	Web page	X

Test Cases and Results

Test Case	**Result of test**
Mobile: User loses internet connection while the application is running	Application ceases to send logs to the server, instead the log continues to gather information until connection is established.
Mobile: Sensor is missing from user's device	Sensor Display page removes ability to toggle sensor's activity. Sensor does not send or receive individual log of non-existent sensor, yet still retrieves it when viewing all charts.
Mobile: User's personal Sensor Log file grows to massive proportions	Log size is now limited. Once max size is reached older entries are overwritten.
Server: User's Log is corrupted	Destroys contents corrupted log. Reverts log to new.
Server: DDoS attack	Firewalls and ports have been configured to withstand DDoS attacks and only accept requests from users of the mobile application.
Server: Malicious user gains access to server	Client account has limited access privileges. Client is only able to affect their personal directory

SCRUM PRODUCT BACKLOG

Sprint #	Priority	Type	Story	Accomplished?
1	1	Mobile/Web	Create GitHub to host project documents and versions	X
1	1	General	Create Project Report #1	X
1	1	Mobile	Create a working basis application for product	X
1	1	Web	Create web server to host data	X
2	1	Mobile/Web	Create method of data exchange between web server and mobile application	X
2	1	Mobile/Web	Cement roles and goals for each team member	X
2	1	Mobile/Web	Create method of user identification through accounts	X
2	1	General	Create Project Report #2	X
2	2	Mobile	Get Xamarin working for everyone in group	X
2	2	Web	Password hashing/security	X
3	1	Mobile	Read data received from server.	X
3	1	Mobile	Single type data sorting algorithms	X
3	1	General	Create Final Project Report	X
3	2	Mobile	Data combination algorithms	X
3	2	Web	Create a web page	X

Change History of Project Report

Date	Name	Change
10/11	Raul	Project Report Created and shared to group members, Added cover Page and Abstract, added change history table
10/13	Takuro	Added Team information, Roles and Obligations
10/13	Paul	Added Table of Contents and Project description
10/13	Mike	Added user Requirements
10/13	Raul	Basic scrum product backlog
10/14	Paul	Created Use Case Diagram
10/14	Mike	Created Sprint Meeting log
10/16	Paul	Appended Sprint Meeting log
10/17	Paul	Established Configuration of LAMP Stack in Project Progression
10/25	Mike	Updated the Sprint Meeting log
10/28	Mike	Updated the Use Case diagram
11/10	Paul	Created the Overall architecture diagram of System design
11/15	Raul	Created Implementation Milestones
11/15	Paul	Translated logs into .csv format
11/17	Mike	Created the class and Sequence diagrams
11/28	Raul	Create account and retrieve password pages in blusheep mobile
11/28	Paul	.Matplotlib Output
11/29	Raul	Update project description and use case diagrams to be more in line with our application
11/30	Raul & Paul	Switch over entire application to use SFTP
11/30	Paul	Resolve file directory management for scripts
12/3	Raul	Troubleshoot test cases and gather results
12/3	Paul	Git Repository URL
12/10	Mike	Updated the Spring Meeting log
12/10	Raul	Added progress pictures of how the app has come so far.
12/10	Paul	Invoked all scripts in a background daemon on server.
12/10	Raul	References
12/10	Takuro	Conclusion and Future Work

TEAM INFORMATION:

Name	*Contact Number*	*Email*	*Operating System*	*Program / Language*
Paul Thomas Rowe	(858)-344-1776	paul@rowe.com	Linux	Bash/Python
Raul Perez	(951)-816-9045	perez236@cougars.csusm.edu	Windows	Xamarin / Xaml/C#
Mike Yoon	(858)-880-8254	yoon005@cougars.csusm.edu	Windows	Xamarin/ Xaml/C#
Takuro Iwane	(858)-527-8870	takuro.iwane@gmail.com	Mac	Xamarin / Xaml/C#

TEAM STRUCTURE:

Name	Role	Obligations
Paul Thomas Rowe	Server Master	Oversee and manage all web server development. Assign roles and goals involving web related activities and applications.
Raul Perez	Mobile Master	Oversee and manage mobile development. Assign roles and goals involving mobile related activities and applications.
Mike Yoon	Web programmer, Communication master	Program for both web and mobile side of application. Manage web and mobile communication, standardize data transfers and allocation.
Takuro Iwane	Web programmer and QA	Program for both web and mobile side of application. Test for program quality and specifications.

SCRUM MASTER ROLE

For our team structure, our project manager/scrum master is being switched on a weekly basis. The role for the Scrum master is to oversee the assignment and completion of the goals set by the team for that specific sprint. Scrum master is in charge of setting the time and location for any team meetings that are to occur during the time of their role as Scrum Master. The Scrum Master sets up the agenda for the meetings, and oversees that the sprint meeting log for that session is filled out.

RECORDER ROLE

Just like for our Scrum Master, our recorder is also being switched on a weekly basis. The role of the recorder is to keep a record of the what is done by the group member during their time as the Recorder, filling out the appropriate sections of the project report, in addition to all of the details that we discuss about in our meetings.

DEVELOPERS AND PROGRAMMERS

Every member of the team has some development or programming task for this application. There are a lot of parts to this project, so team members will get assigned development or programming tasks as needed to complete our goals.

Some of our team members have specialization and lead the development of a certain section of the project. Paul specializes in linux and is in charge of assessing and assigning goals to other members for everything related to web for this project. Raul specializes in mobile application and so he is taking the lead in developing the application, demonstrating for the rest of the team the instructions on how to start an application development. Mike is one of the programmers and also in charge of communications, and Takuro is another programmer, and is the main tester for our project.

QUALITY ASSURANCE

Every member of our team is an upperclassman in the Computer Science department, thus we all have experience and knowledge of what the application should do and how it should work, thus each member is equally responsible for making sure the application is up to acceptable standards. Paul has a focus on the web development portion of QA, ensuring that all things web are up to standards. Raul has a major focus on the mobile side of development, ensuring that all things mobile are up to the standard. Due to Xamarin's ability to create applications for both iOS and Android at the same time, each team member is capable of testing the application on their own devices, thus are also able to keep track of changes and quality of the application as we progress through it. Mike tried to develop his part of the project through Android Studio, using Java. The team decided to implement the project through Xamarin Studio.

SPRINT MEETING LOG

Date	Members	Log
9/11	Raul Paul Mike (Takuro was not member of team yet)	Agenda: ● Forming our team, and begin to discuss about what kind of project we want to develop Tasks: 1.Paul (Manager) - Work done: Gathering team members - Work to do: Think about ideas for the project 2. Raul - Work done: Created the basis for the mobile app - Work to do: Add more functionality to app 3. Mike (Recorder) - Work done: Meeting with other team members and thinking up of ideas - Work to do: Looking into mobile apps
9/20	All	**Agenda** ● Finalizing our project idea and starting on the project proposal **Tasks:** 1. Paul (Manager) - Work done: Making sure everyone arrived, and proposing the idea - Work to do: Working on the cover page, abstract, and signature 2. Raul - Work done: Introducing the basis of android mobile app - Work to do: Working on Tools/Language, Constraints and Challenges, Evaluation, and the logo 3. Mike (Recorder) - Work done: Giving ideas for our team name and project name - Work to do: Working on Proposal, What is it solving, and Who is our targeted users 4. Takuro - Work done: Giving personal inputs - Work to do: Working on Project Objectives, and Expected Outcome
9/28	All	**Agenda:** Start to work on the project **Tasks:** 1. Paul - Work done: Going over our proposal and picking Bluetooth as his focus - Work to do: Trying to get a working app using Android Studio 2. Raul (Manager) - Work done: Picking gyroscope as his focus

		- Work to do: Trying to get a working app using Xamarin 3. Mike - Work done: Picking ambient light sensor as his focus - Work to do: Trying to get a working app using Android Studio 4. Takuro (Recorder) - Work done: Picking screen motion as his focus - Work to do: Trying to get a working app using Android Studio
10/6	All	Agenda: See the progress of our android apps, begin working on project report #1 Tasks: 1. Paul - Work done: A somewhat working LAMP stack. - Work to do: Getting an HTML page connected to a SQL Database 2. Raul (Manager) - Work done: Used Xamarin to work on his app - Work to do: Continue Xamarin and work on report 3. Mike - Work done: Trying to develop an app through Android Studio - Work to do: Continue to develop and work on report 4. Takuro (Recorder) - Work done: A somewhat working app - Work to do: Work on report
10/13	All	Agenda: ● Talk about goals of Application ● Assign roles and Objectives ● Work on Project report ● Choose Mobile development platform ● Choose Application name and mascot Work Done Since Last Meeting: ● NA Work Needing to be done before next Meeting: ● Create GitHub for Mobile and Webs Problems/Obstacles ● Arranging meeting times. ● Choosing mobile development platform ● Time and role management delegation ● Communication
10/17	All	Agenda: ● Work on Project Report Work Done Since Last Meeting: ● Created basic mobile application

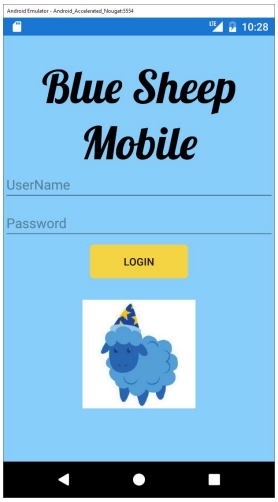
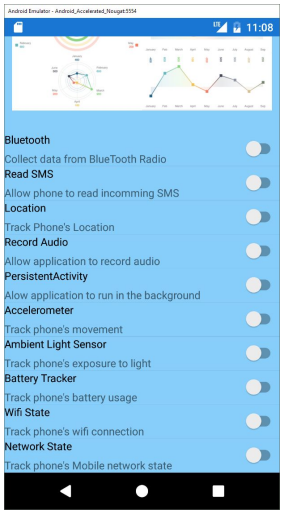
		• Created basic web server • Created Logo for Application Work Needing to be done before next Meeting: • Create method of data exchange between web server and mobile application • Cement roles and goals for each team member • Create method of user identification through accounts • Get Xamarin working for everyone in group • Finish Project Report 1 Problems/Obstacles • Arranging meeting times. • Role delegation
10/20	All	Agenda: • Intro to Xamarin • Get Xamarin working for everyone • Assign mobile permission focus for each team member Work Done Since Last Meeting: • N/A Work Needing to be done before next Meeting: • Downloading Visual Studio/Xamarin • Watching a tutorial to understanding it better • Looking into each focused mobile permission Problems/Obstacles • N/A
10/25	All	Agenda: • Understanding Xamarin better • Using Xamarin to begin our coding Work Done Since Last Meeting: • Everyone downloaded Xamarin • Selected a focus to work on Work Needing to be done before next Meeting: • Knowing the bits and pieces of Xamarin • Have a basic foundation of coding Problems/Obstacles • For many of us it's the first time to use Xamarin • Knowing how to use it • Which sensor to work on
11/3	All	Agenda: • Looking over our coding and see what is working and what's not • Start to take a look at visualizations and to implement them • Begin to work on class diagrams • Compare each other's work Work Done Since Last Meeting: • Started to code our project using Xamarin

		Work Needing to be done before next Meeting: ● Continue to work on the code ● Thinking about visuals and what can we use in our project ● Outline of diagrams Problems/Obstacles ● Still learning how to work around Xamarin ● What kind of programs or languages will we need for visuals
11/15	All	Agenda: ● Putting everyone's code and progress together ● Testing it on an Android device ● Begin working on Project Report 2 Work Done Since Last Meeting: ● The fundamental coding of our project ● Class and sequence diagrams ● Visualizations to display what the program is doing Work Needing to be done before next Meeting: ● Testing of our project ● Project Report 2 ● Finalizing the visuals ● Setting up AWS Problems/Obstacles ● Putting all the codes together
11/18	All	Agenda: ● Finalize testing show case for Mon/Wed ● Having our individual contributions to present ● Work on the Project Report 2 Work Done Since Last Meeting: ● Several test runs of our project ● Created the outline for our Project Report 2 ● Visualizations for our project Work Needing to be done before next Meeting: ● Project Report done by Mon ● A presentation of our project by Mon ● Sending information from the Xamarin client to AWS Problems/Obstacles ● How to send the data from the app to the servers
11/22	All	Agenda: ● Continue on working on the application ● Work on the AWS instance ● Research on how to get a visual working Work Done Since Last Meeting: ● Project Report 2 finished ● Did the presentation on Monday Work Needing to be done before next Meeting:

		• Getting the AWS to work • Deciding what to do for visuals • Continue to work on the application Problems/Obstacles • Xamarin is not the most compatible tool for AWS • Possible visual idea is using Python and Bash
11/25	All	Agenda: • Find out who will do what for the remaining objectives • Still figuring out the AWS • Learning Python and Bash Work Done Since Last Meeting: • Big progress on the project • Finally figured out the AWS instance • Individual work Work Needing to be done before next Meeting: • Get closer to getting a working AWS • An understanding of Python Problems/Obstacles • For all of us, we are new to Python and Bash • Getting the AWS to work is probably our biggest struggle
11/26	All	Agenda: • Continue working on the project individually • Using Python (matplotlib and numpy) and Bash for visualization • Get as much done as possible Work Done Since Last Meeting: • Figured out who will do what • General understand of Python • Started the visualizations (application) for our project Work Needing to be done before next Meeting: • Do our own individual work • Finish the visualization Problems/Obstacles • Python and Bash was difficult to figure out • A lot of work is still left to be done
12/2	All	Agenda: • Continue to work on our own individual work • Deciding where to display our visualizations • Make sure that our application is 100% complete (Testing) Work Done Since Last Meeting: • Progress on our own individual work (Almost complete) • Finished creating our visualizations for the application Work Needing to be done before next Meeting: • One possible idea for displaying our visualizations is by using a website • Look into how to create a website (HTML)

		• Finalizing our own individual work Problems/Obstacles • Using HTML
12/3	All	Agenda: • Begin working on the website • Adding the visuals to the website • Also begin to work on the Final Report Work Done Since Last Meeting: • Completed our individual work • Created a website Work Needing to be done before next Meeting: • Finish creating the website • Designing the website • Get as much of the Final Report finished as possible Problems/Obstacles • N/A
12/10	All	Agenda: • Finalizing our project, our last meeting • Finish creating the web page • Finish the Final Project Report Work Done Since Last Meeting: • Completed our individual work • Majority of the web page is finished and ready • Started on the Final Project Report Work Needing to be done before the final day: • Finish the web page • Finish the Final Project Report Problems/Obstacles • N/A

PROJECT PROGRESSION IMAGES

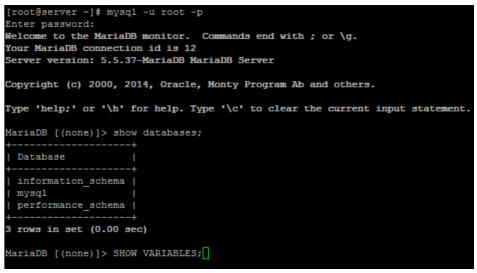
Image	Description	Image	Description
Blue Sheep Mobile Sprint 1 Minimum Viable Product			
	This is the home and login screen for our application. Here the user is required to input their username and password in order to login and use the application. We have the name of our application at the top, with the logo underneath the login button.		List of mobile permissions that we can access. Allow user to choose what is on or off to give them control over their privacy. All on by default, and must be manually turned off.
	Made minor changes to the visual and structural; layout of the application. Sensors are now able to be toggled on and off, displaying the raw data on the screen in real time. Now they are off by default and must be turned on to begin collecting data. Demonstrating the data being collected using the sensors available on a Pixel XL.		Representation of what the charts will look like after receiving processed data from the server. Charts are created using MicroCharts.Forms Note: *This is a mock-up.*

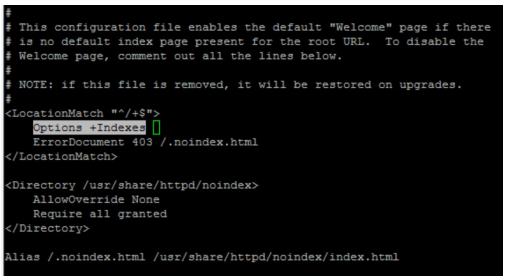
Central AWS Server Sprint 1
Initialization of Service

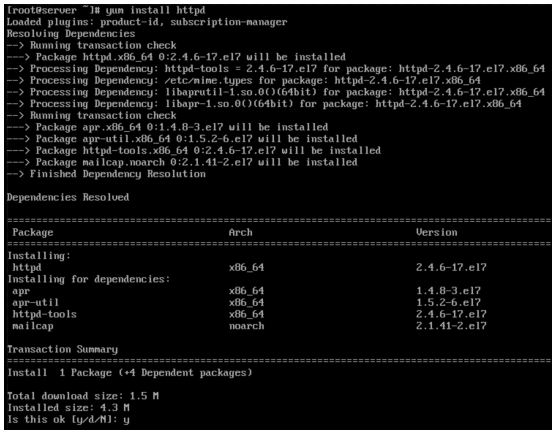
These are screenshots detailing the installation of our LAMP Stack Webserver.

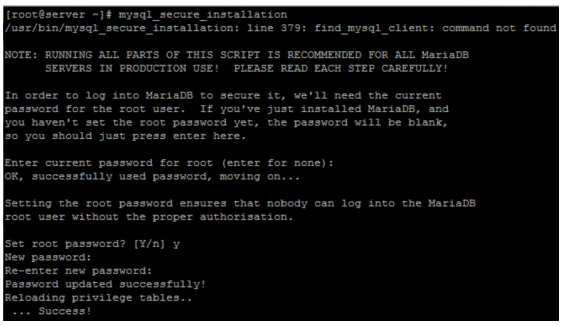
(Linux + Apache + MySQL + PHP)

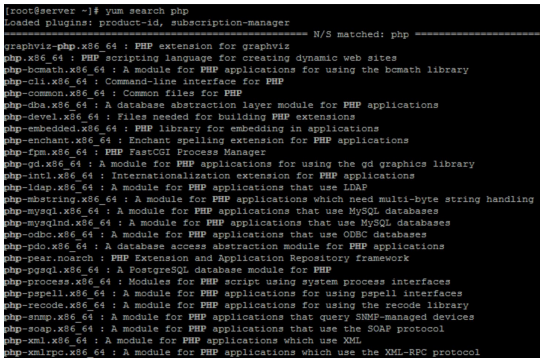
We still have yet to learn how to use it, but we've got it working!

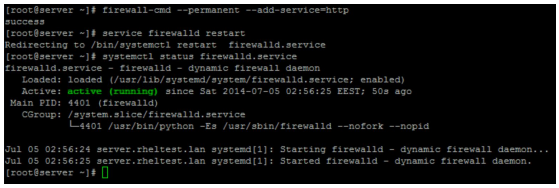


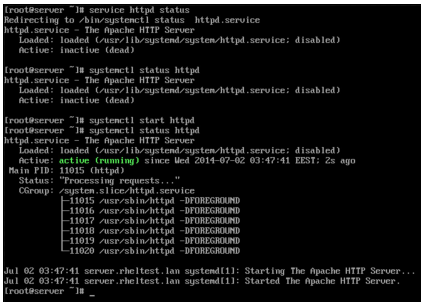












Parsing and Visualization Rehearsal

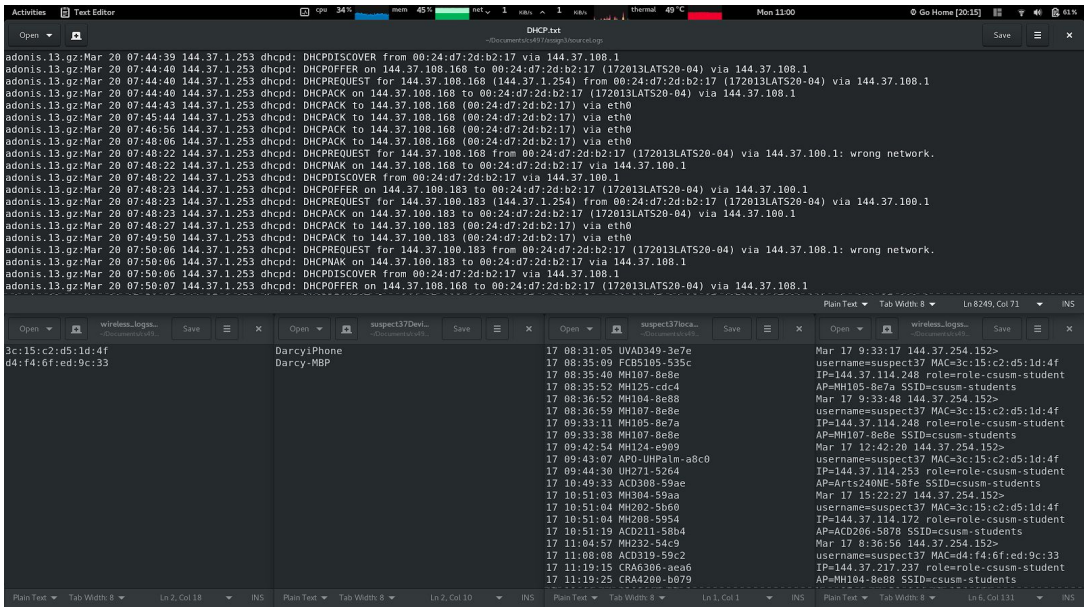
Using sample data in lieu of actual data.

Screenshot: Creating scripts to parse through log records:

ParseCommands.sh

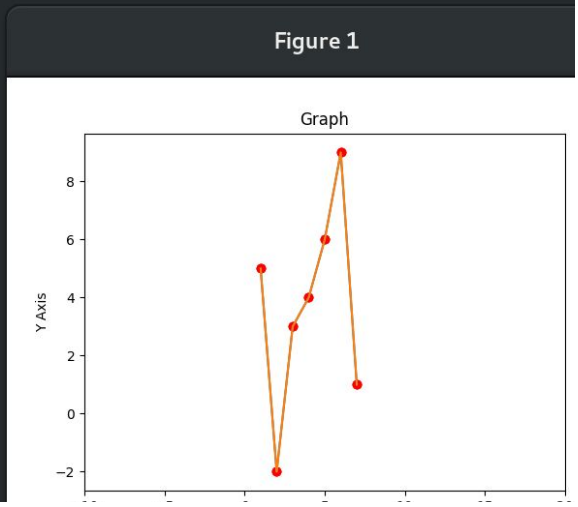
```
//Abbreviating the logs:
//DEVICE NAMES
cat ./sourceLogs/DHCP.txt | grep DHCPACK | sed  's/.*(\(.*\)).*/\1/' | sort | uniq > ./parsed/deviceNames.txt
//DHCP
cat ./sourceLogs/DHCP.txt | grep "DHCPACK" | cut -d ' ' -f2,3,8,10,11,13 | grep -v "eth0" | uniq -f5 | awk '!seen[$0]++' | sort -n > ./parsed/DHCPparsedOutput.txt
//WIRELESS
cat ./sourceLogs/wireless.txt | grep "AP-name" | cut -d ' ' -f2,3,4,9,10,11,17 | uniq -f5 | grep -v "DN:" | awk '!seen[$0]++' > ./parsed/wirelessParsedOutput.txt
//WIRELESS_LOGS
cat ./sourceLogs/wireless_logs.txt | grep "suspect" | cut -d\t -f1,2,3,6,10,11,12,13,15,16 | tr "\t" " " | uniq -f8 | grep -v "de-authenticated:" > ./parsed/wireless_logsParsedOutput.txt

//Identifying a suspect:
//WIRELESS_LOGS
cat ./parsed/wireless_logsParsedOutput.txt | grep "suspect37" > ./suspect37/wireless_logssuspect37.txt
//MAC ADDRESS
cat ./parsed/wireless_logsParsedOutput.txt | grep "suspect37" | cut -d ' ' -f6 | cut -f2 -d"=" | uniq | sort -u > ./suspect37/wireless_logssuspect37MAC.txt
//LOCATIONS
cat ./parsed/wirelessParsedOutput.txt | grep -f "./suspect37/wireless_logssuspect37MAC.txt" | sort -n | cut -d ' ' -f1,2,7 | sed 's/AP-name=//g' > ./suspect37/suspect37locations.txt
//DEVICE NAMES
cat ./parsed/DHCPparsedOutput.txt | grep -f "./suspect37/wireless_logssuspect37MAC.txt" | sort -n | cut -d ' ' -f5 | sed  's/.*(\(.*\)).*/\1/' | uniq > ./suspect37/suspect37DeviceNames.txt

//ScatterPlot:
cat wireless_logssuspect37MAC.txt | cut -d -f1,2,3
```

Screenshot: Examples of how a given log record had been parsed out:



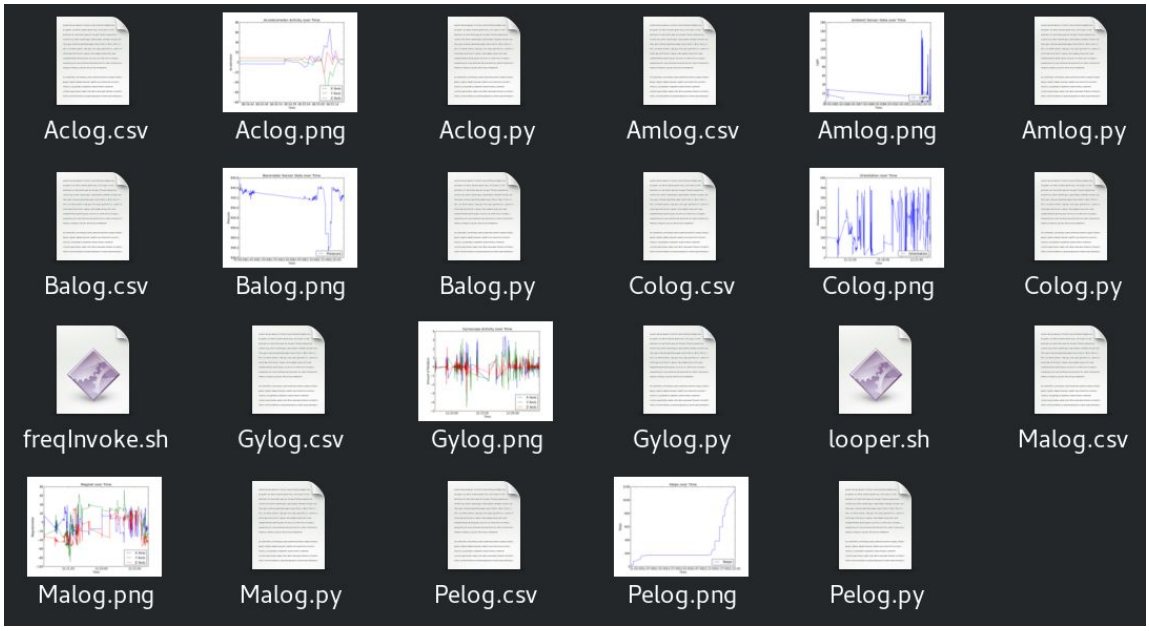
Screenshot: Beginning to visualize some very simple data:

```
#https://www.youtube.com/watch?v=oflixU61NDc
#data= pd.read_csv("cgpa.csv")
#print(data) ["cgpa"])
import matplotlib.pyplot as pt

data= pd.read_csv("cgpa.csv")
data=data.head(30)
pt.plot(data["rollno"],data["cgpa"],color="orange",label="line grap

pt.scatter(x,y,color="red")
pt.xlabel("X Axis")
pt.ylabel("Y Axis")
pt.title("Graph")
pt.xlim(-10,20)
PT.ylim(-20,20)

x=[1,2,3,4,5,6,7]
y=[5,-2,3,4,6,9,1]
pt.plot(x,y)
pt.scatter(x,y,color="red")

pt.show()
```

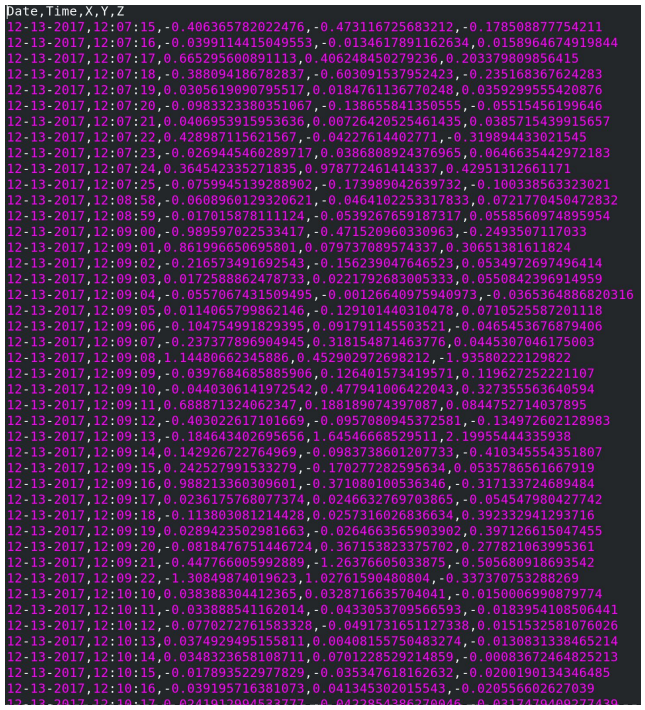


Parsing and Visualization Finalization

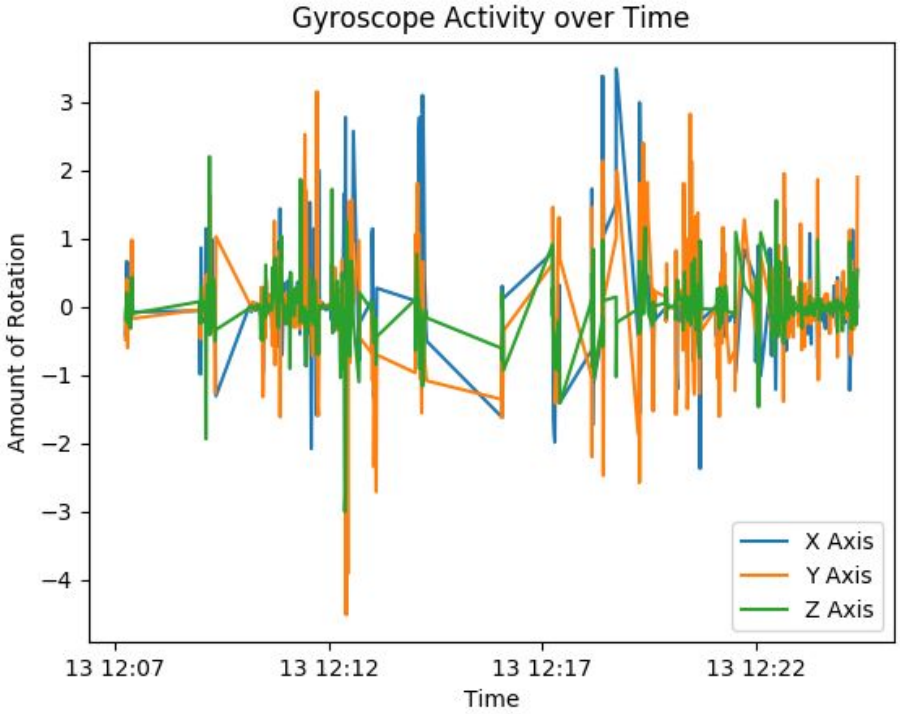
We're now using actual data!



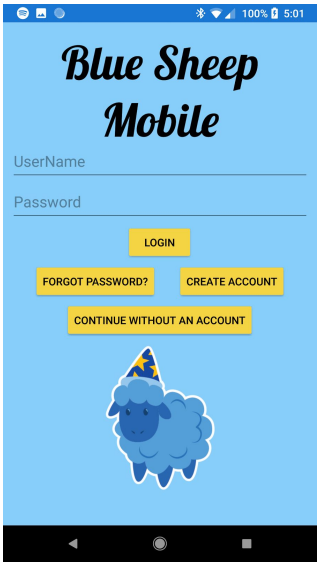
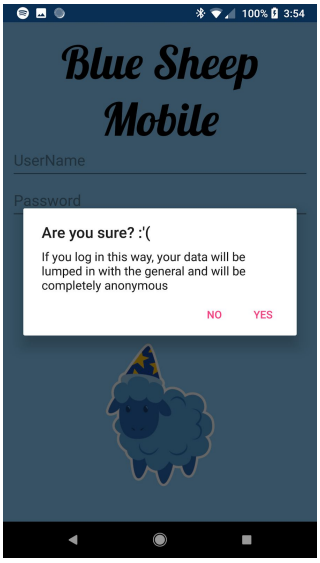
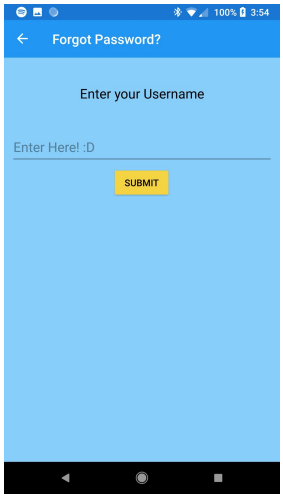
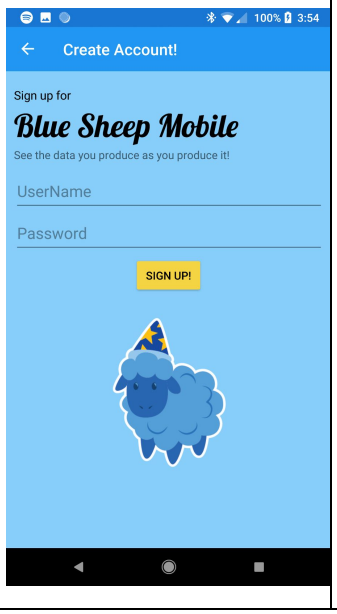
Sample of Data Processing:

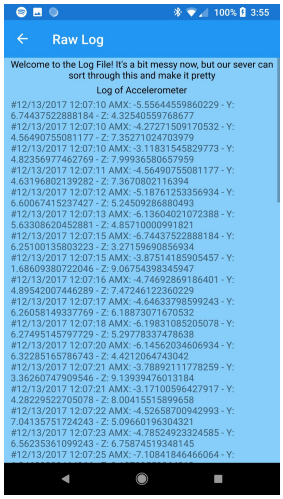
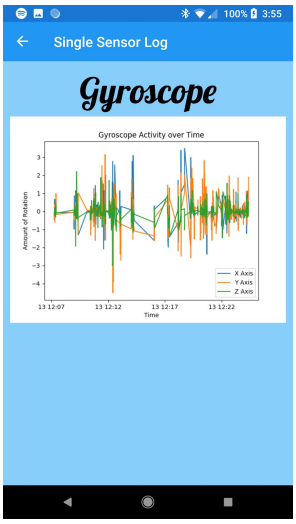
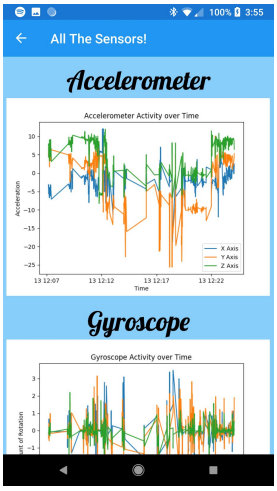
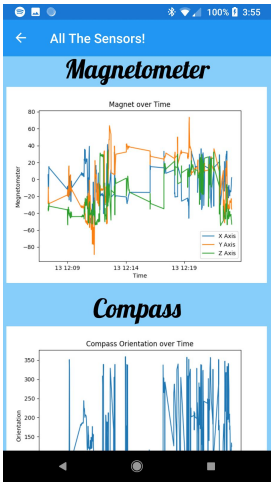
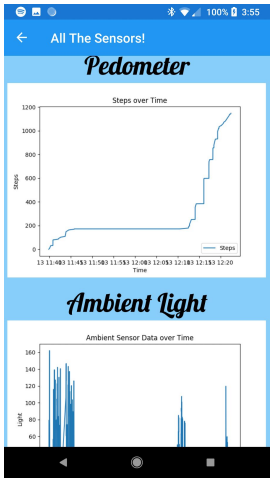


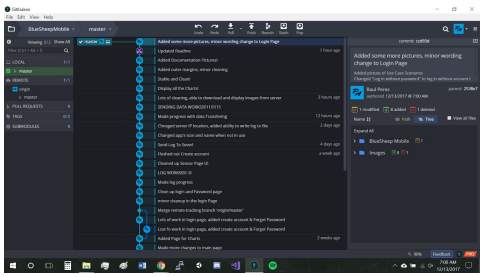
Final Visualization of a Set of Data:

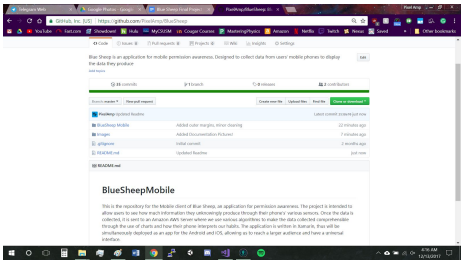


Mobile Sprint 2
Interface Redesign

	Log in screen is now much cleaner, minimalistic and lightweight. Users now have the option to create an account, retrieve a forgotten password, or continue without an account		Notification appears if the user wishes to login without username or password. By selecting this the user's data become anonymous, then proceeds to the main application with limited usage.
	User is able to retrieve a forgotten password. The server checks the devices ID, ensuring that the username matches the user and sends the user their password.		Users have the ability to make their own accounts, making their data personalized. This gives the user to track their data against the general populous.

	Small redesigned central page. Now data is much easier to read. Interface redesigned to have the useful button clearly available, Concealing less useful options behind a long press.		Performing a long press on a sensor brings up context actions to be performed using that sensor. The user is now able to see the raw log being created by that specific sensor, as well as see the individual graph for that specific sensor.
	RawLog Page demonstrates what log of the selected sensor looks like before being sent to the server for parsing.		Single Graph Page page fetches and displays the graph for that sensor alone.
			Master Graph page fetches and displays all the available charts currently on the server.





Blue Sheep Mobile Client Github was continuously updated as milestones and goals were reached. In the end there were over 39 commits made to repository. https://github.com/PixelAmp/BlueSheep

INDIVIDUAL CONTRIBUTIONS

Paul Thomas Rowe:

Paul has set up the project's web server, by setting up an AWS EC2 instance with an installed LAMP stack. In preparation of receiving actual log files, Paul has paved the way towards parsing through the massive amounts of log records we anticipate receiving by writing sample bash shell scripts. He has done this by using sample log records containing hundreds of thousands of lines as a test case. He has also begun visualizing very simple subsets of data, with the overall goal of creating comprehensive visualizations of the data we receive from all available sensors.

Raul Perez:

Raul has worked on developing mobile cross-platform application Blue Sheep Mobile using Xamarin.Forms. Raul configured the application to gather and assemble data collected from various sensors into individual log files, which are then used sftp to periodically sent to the sentral server. The mobile application is capable of receiving processed information from the server and displaying it on screen for the user to view.

Mike Yoon:

For this project, Mike contributed to the recording of sprint meeting logs, including keeping a record of what the group did and intended on having accomplished before the next meeting. Mike made contributions to Project Report #1, #2, and the Final report.

Takuro Iwane:

Takuro was in charge of creating a web page that would display the images that will be generated in the servers. Also did the designs and features for the web page.

CONCLUSION

In conclusion, we knew this project was going to be difficult from the very beginning. Our goals were ambitious and we were working with technology we had little to no prior experience with. However, our team has persevered through the difficulties by keeping an optimistic attitude. All semester long we forged through several different kinds of challenges, and as a team we had to figure out different ways to find a solution for every problem that we encountered. Some of the difficulties we encountered and have to overcome, include sending files to and from from our application to the AWS instance, by using programs and languages that we've never touched before, as well as creating a visualization that parses through and is able to display monumental amounts of information collected from our application. Now that we are at the end we are glad to say that we have been able to make a product we are proud of. We focused on achieving our goals and working toward our sprints, changing our outlook on the difficulties we faced to be learning opportunities, rather than stumbling blocks. In the end, our project managed to come even better than we expected having reached all of our goals. We really are thankful for this experience as we know that working on on this project will make us better software engineers.

REFERENCES

Xamarin
Software used to code the application
https://www.xamarin.com/

AWS (Amazon Web Services)
Used to store data from the application
https://aws.amazon.com/

Apache
Web server software
https://www.apache.org/

Draw.io
Flowchart and Diagram application
https://www.draw.io/

Rebex
Xamarin plugin allowing our application to perform SFTP
https://www.rebex.net/

Stack Overflow
Used in resolving python visualization issues.
https://stackoverflow.com/

Unix & Unity Stack Exchange
Used to learn how to use bash scripting tools in order to efficiently parse log data.
https://unix.stackexchange.com/

Red Hat
Provided valuable documentation on configuring our Red Hat- based AWS instance with an Apache web server.
https://redhat.com/

Aritchie's Plugin.Sensors
This was a Xamarin plugin giving us a way to natively read sensors at a fixed time interval, monumentally simplifying our data collection system.
https://github.com/aritchie/sensors

Adobe Photoshop CC & Illustrator CC
Used for web design
https://www.adobe.com/Photoshop

Bootstrap
Tool kit for developing HTML, CSS
And JS.
https://getboostrap.com/

Adobe Dreamweaver
Create responsive HTML website for
Both PC and mobile phone
https://adobe.com/Dreamweaver