Electronic Health Record System

Jack Laborde, Sagar Patel

CSCE 513/CMPS 499  
University of Louisiana  
Lafayette, LA, United States

***Abstract*—The following paper presents a project implementation that we have chosen that involves the creation of an Electronic Health Record system which has the ability to record data live for viewing by a Patient and their Doctor. The data is streamed via Bluetooth to the Patient’s mobile device, where it is processed for viewing and prepared for transmission to an Amazon EC2 instance, which contains an RDS database that stores this data. The data can then be retrieved by both doctors and patients to view their data on the web or on their devices, which is then visualized to observe certain patterns or behaviors, and potentially mark concerns.**

***Keywords***—***EHR = Electronic Health Record; EKG – Electrocardiogram; IoT – Internet of Things; MAC – Media Access Control; WLAN – Wireless Local Area Network; TX – Transmit; RX – Receiving; BLE – Bluetooth Low Energy***

1. INTRODUCTION

This project is based upon the creation of a multi-stage EHR system that can receive data from specified Bluetooth devices, which will be described in Section III (DEVICES), utilizing a point-to-point connection that will stream character data to a patient’s mobile device. The data is then processed within the mobile application (See Section VI, THE MOBILE APPLICATION) for viewing by the patient.

The patient is able to see their current data and data averages from the device they’re reading from. In this project, an EKG system is used as the data testbed, and the patient can view the current pulse, the last 500 recorded pulses, and the upper and lower threshold pulses from their heartbeat. The threshold pulses are set remotely on the doctor’s end, utilizing their mobile application or the website to update, view, and edit data.

During this stage, a timer value pulled from the same database server sets the rate (in seconds) at which the data is to be transferred to the database. For example, a doctor sets 20 seconds on their end and updates the patient data, which means the device will read that change, and a pulse every 20 seconds is sent to the database server and added to the patient’s file for visualization later.

The doctor is then able to view this data on their own mobile application or website, depending upon their preference. They will be able to visualize data, set the thresholds, and potentially send warnings to the user should thresholds be exceeded too often.

This multi-stage project was a fairly large undertaking, but overall we enjoyed success in creating the system, even if we did run into a multitude of errors and issues (See Section X ERRORS AND ISSUES). We believe this project has a potential commercial future as well, especially if we unify the project to take in data from almost any specified device (See Section XI, FUTURE).

1. BACKGROUND

This project is utilizing different communication protocols to transfer data from one end to the next. Many similar projects have been achieved before, including devices like the Fitbit, which also transfers data to a phone (although not necessarily to a backend server, as the Fitbit was meant for personal use). Fitbit use the Bluetooth 4.0 wireless standard, which we utilize as well for our project.

However, unlike the Fitbit and many other personal fitness tracking devices, we wanted our system to be used between a patient and a doctor, so a doctor can remotely monitor treatments and modify them as necessary, and a patient has comfort knowing that they are being taken care of, and if an emergency happens, detection systems can send out alerts to emergency contacts or even 911. Using the patient’s cellular or Wi-Fi data, we can send their data securely to a backend server which can be visualized by both the patient and the doctor.

So first, we start with Bluetooth, a short range communication technology developed by Ericsson in 1994, and expounded upon in 1998 by a coalition of different corporations (called the Special Interest Group – which involved Sony-Ericsson, IBM, Intel, Nokia, and Toshiba) that sought to make it a standard. Bluetooth eventually developed Enhanced Data Rate for file transfer and Advanced Audio Distribution Profile to play audio wirelessly. Over time, Bluetooth became much stronger, and began to permeate the commercial market, with wireless headphones, keyboards, audio systems, and other devices utilizing this new technology. In 2010, we get Bluetooth 4.0, which is now the common standard for many devices, although Bluetooth 4.1 and 4.2 have been released. 4.0 introduced new modes of Bluetooth, including High Speed and Low Energy. Low Energy is favorite of many because of its safe properties and extended range, and the ability to transmit data without utilizing too much power. This short range technology is perfect for data transfer between close proximity devices, in our case being the EKG and the user’s mobile device.

Wi-Fi and Cellular Networks are the next utilized modes of communication for this project. Both of these have a long history, which is far richer than the history of Bluetooth, and is beyond the scope of this paper. What is notable, however, is the methods these communication protocols use. Wi-Fi uses the 802.11b IEEE Standard, which sets specifications for MAC and Physical Layers for implementing WLAN device communications across different frequency bands, including 2.4, 3.6, 5, and 60 GHz bands. These standards were released in 1997, and has been upgraded ever since to make it stronger and faster. Most smart devices (and even older devices) possess Wireless Network Interface Controllers that allow the end user to access Wi-Fi networks. However, it is subject to interference from Bluetooth devices, which was taken into minor consideration, but was never a true concern.

Cellular networks may use different frequencies and protocols, but are similar in nature to Wi-Fi. Discussing Cellular in detail is redundant, and plays as a backup to where Wi-Fi is unavailable (or if the patient has Wi-Fi disabled).

1. DEVICES

The devices used in this project range from custom hardware to smart phones, each device having a different purpose. We built a custom hardware set to serve as our EKG platform, using an open-source I/O board and an open-source EKG shield. This data collection platform was extremely useful and efficient for our needs, but any Bluetooth capable monitoring device can serve in its place.

1. Hardware
2. Arduino UNO R3

This is the main I/O microcontroller board that serves as the operating platform of our EKG. It has 14 input/output pins, 6 analog inputs, a 16 MHz processor, and a C++/C interface and serves as our main board for the project. It is powered by either a computer connection or through the power jack by a 9-volt battery. It processes all the raw analog data that’s received from the Olimex EKG Shield.

1. Olimex EKG Shield

This shield serves as a direct connection shield to the Arduino UNO. The shield is placed on the main pin slots of the Arduino, and only utilizes a single Analog input for reading analog data. The data is retrieved through an audio jack, which contains connections to three electrodes that the patient will place on their wrists and either their forearm or their foot. The electrodes will listen to the audio produced by the heartbeat, and then calculate a final reading on the Olimex board.

1. BlueSmirf Mate Silver (RN-42)

The BlueSmirf Mate Silver is the Bluetooth interface card that is powered through the Arduino UNO. The interface only requires two pins, an RX and TX pin and can be placed on any free pins on the Olimex shield. For this project, we chose pins 10 and 11 to serve as TX and RX respectively, and the code to set said pins was denoted in the Arduino IDE. The BlueSmirf allows BLE communication for up to 100 m, which is perfect for devices that can’t be out the patient’s sight. (See an example situation in Section VI, THE MOBILE APPLICATION [PATIENT-SIDE]). The BlueSmirf is also secured through a PIN code, so that no malicious users can attempt to access a patient’s data. A secure transmission is absolutely crucial for the security of a patient’s data.

1. MCUFriend 3.5’’ TFT LCD

This is simply an LCD screen that’s another shield that is place on top of the Olimex Shield. It has the ability to produce 16-bit images and it is able to display the current readings of the EKG shield, which is highly useful for testing purposes. It displays the readings by drawing lines like a typical EKG reader, and prints out the current BPM for the patient. It is not completely accurate due to calculation errors, but is fairly on target for testing purposes.

The final configuration of the project is shown in Figure 1.

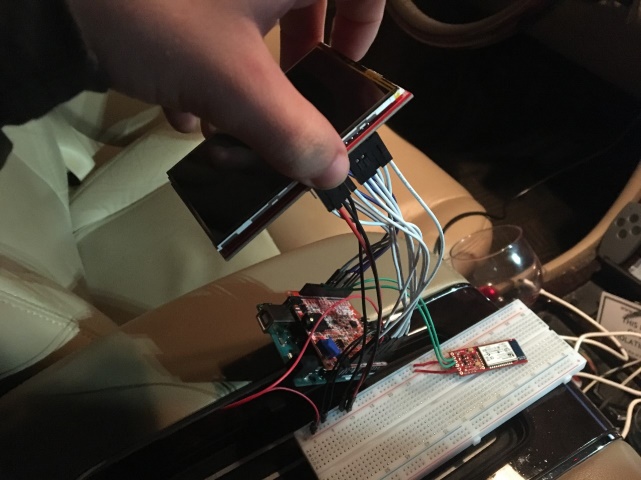


Figure 1: Project Hardware Configuration

1. Devices
2. Android Phone/Tablet

Any smartphone or device with Bluetooth capability are perfectly useful for the project. For testing purposes, we used a Samsung Galaxy Tablet with Bluetooth 4.0 capability. The Bluetooth device is successfully found on these devices (unlike iOS, which requires modification of the BlueSmirf settings to feature them on iOS Bluetooth listings). This device will hold the application software that will receive the Bluetooth character data, and transfer via Wi-Fi or Cellular Network to the backend server hosted by Amazon.

1. ARCHITECTURE

The architecture of the system is fairly simple. We have the basic parts as described above: we have the sensor module (the audio port reading the data), the Bluetooth module to transmit the data, the mobile application to interpret, transmit, and receive data, and the web application to do the same. The sensor module will capture the user’s heart rate sound as it listens through the electrodes on their body, and transmits that signal to the Bluetooth module in a character format (the Bluetooth buffer automatically transmits as characters, even though it is numerical data). The Bluetooth module then modulates the data to make it readable to the mobile application. The mobile application (Android) will receive and process this data for visualization, storage, and further transmission to the web server. The web server, hosted on Amazon, receives the data and stores in the table for use and editing by medical personnel.

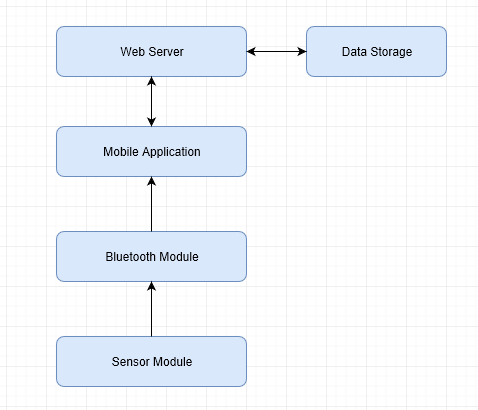


Figure 2: System Architecture

1. THE ARDUINO EKG

As described before, the EKG consists of an Arduino UNO R3, the Olimex EKG Shield, a BlueSmirf RN-42 Bluetooth card, and a 3.5’’ TFT LCD screen. The data that is read from the Olimex is received from three audio electrodes attached to the person’s body, which listens for the sound of the heartbeat through the patient’s body in real time, which is then sent to the analog pin for processing into a raw float value. This raw value is used for drawing the EKG lines as displayed on the screen as well as calculating the beats per minute. The analog input retrieves a voltage value, and uses a Divider Calibration Value (10.935) and an ADC Reference Voltage Calibration Value (5.03) to help fully calibrate the board. The calculation is as follows:

Variables:

1. Vl = Analog Read Value
2. VREF = ADC Reference Voltage
3. Dvl = Divider Calibration Value

* ((Vl\*VREF) / 1023) \* Dvl

This equation, if properly calibrated (the numbers provided for the VREF and Dvl variables were precomputed by the Olimex team, and may in certain instances require modification) can provide an accurate reading of beats per minute, which will then be displayed on the screen to the patient. The patient is able to observe the current beats per minute and their actual pulse monitor through a canvas drawing of the EKG readings.

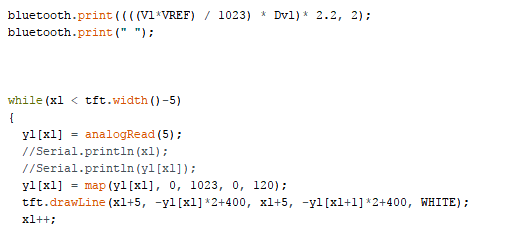


Figure 3: Bluetooth/Drawing Code from the Arduino IDE

This data is then forwarded to the Bluetooth RN-42 card, where it is now in a character format. Before it can transmit, it needs to detect if there is a paired device in a 100 meter area. If there is no paired device, a patient must use their mobile device to manually pair and enter the PIN code to begin transmission of the data. Once the PIN is accepted and the device paired, the Bluetooth will begin immediate transmission of the pulse data. The data is then received at the mobile application, where it is then processed for use in the application.

Figure 4: EKG Screen Data

1. THE MOBILE APPLICATION [PATIENT-SIDE]

The mobile application was developed in MIT’s App Inventor, which makes rapid prototyping and deployment much easier for the developers. It utilizes a drag and drop interface that integrates all the core features of Android devices, including the ability to make calls and send text notifications, which is useful for the emergency portion of this application.

1. *Initial Setup*

When a user enters the application for the first time, they will be presented the option to go to the Settings screen, as seen in Figure 5 below. Here is the initial setup of the application, where the patient must be configured to match with their website identification in order for their information to record correctly.

As of Version 1, the method of doing this is medical personnel were to enter their password on the application, then click the “Open Setting…” button, which would lead them to enable modification of the “Patient ID” setting. When this is saved and updated, it stores the new ID in a local database, which will identify which Patient to update in the backend and what information to pull from the server. This will allow the thresholds and data retrieval rate set by medical personnel to be retrieved and stored along with the current pulse being sent to the server.

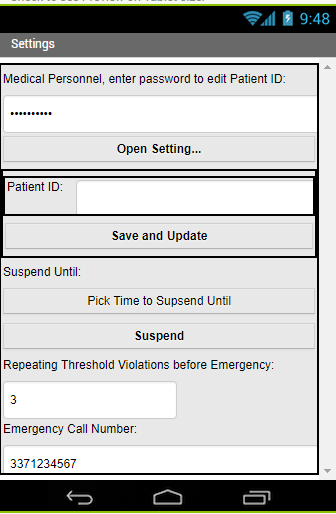


Figure 5: Settings

The patient will also have the ability to suspend the monitor until a certain time, in the case of events like a patient needing to shower, or do certain activities. The patient will have the ability to set an emergency call number if they wish, which will automatically call that number if the user exceeds the “Repeating Threshold Violations” value. This value simple means that if subsequent pulses exceed threshold values without returning to normal, then the emergency will trigger and send out a call to that number. The user may opt to send a text message as well, as well as opt to call 911 if the user’s emergency contact fails to answer.

1. *The Home Screen*

The Home Screen (See Figure 6) is where the patient can view all the data that is streaming in from the Bluetooth device. For a more personalized experience, “Hello, Patient” is replaced with the patient’s name, giving a more friendly atmosphere to the application.

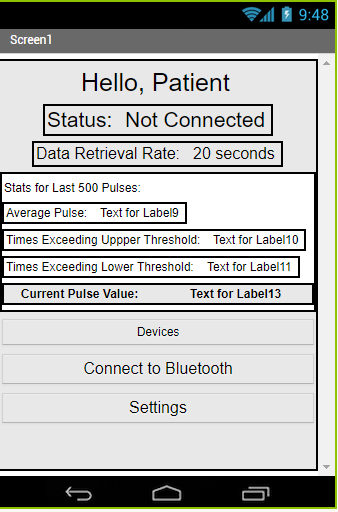


Figure 6: Home Screen

Next, there is a status label which shows the current connection to the Bluetooth device. If there is a Not Connected State, the user can simply select “Devices” and choose the RN-42 from the popup. Except in specific circumstances, they do not have to touch the “Connect to Bluetooth” button, as it is done automatically. If a connection is lost, the status will again say Not Connected but will appear in red, notifying the user that their device is not detected. Upon a successful connection, a green ‘Connected” text takes the status text.

The Data Retrieval rate that follows is simply the time per interval of data retrieval. In Figure 6’s case, where the rate is 20 seconds, every 20 seconds the application will send a connection string to the backend server, and will then take the current pulse value upon successful connection and store it to the Patient’s ID, and detect whether it has exceeded the threshold or not.

1. *500 Pulse Statistics*

While the data is fetched from the Bluetooth device, and before it gets sent to the backend server, it is calculated with the previous 500 or less pulses, attaining the average pulse and the amount of times one of the thresholds are exceeded. It lastly prints out the current pulse that was retrieved.

1. *Data Code*

The Data Retrieval code is run on a timer that first attempts to check the connection between the Bluetooth device and the user’s mobile device. The data is retrieved in byte format and automatically parsed to a variable, which is a string value. More than one data pulse is actually sent with each retrieval, so the parser has to split the text at spaces and add them to the global Pulse List. With that list in mind, the average and exceeds are calculated, along with detecting subsequent exceeds beyond thresholds. If the subsequent values go beyond the counter (in this case, 3), then the user is immediately taken to the Threshold Alert Screen (Figure 7).

If this does not happen, the labels are updated with the new values, and if the list exceeds 500, it is reset and emptied.

1. *Threshold Alert Screen*

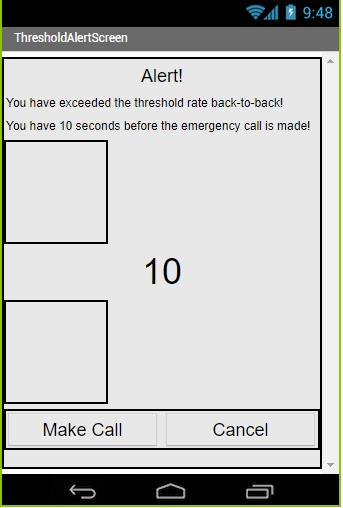


Figure 7: Threshold Alert Screen

The Threshold Alert Screen is automatically opened when an emergency is declared. The user is given 10 seconds to cancel a false positive or force the call to happen. Once the “Make Call” button is pressed, the user’s stored emergency number is called (and texted, depending upon the user’s settings).

The call uses text-to-speech technology to automatically make the call if the patient is incapacitated. If the emergency contact fails to answer, depending on the settings, 911 will be contacted with the same message. A text message may also be sent out, and if the emergency contact replies OKAY, it relies on that contact to get to the patient and assist if necessary and cancels a potential 911 call. If the patient presses Cancel, it will return them to the home screen.

1. THE MOBILE APPLICATION [DOCTOR-SIDE]

As of Version 1, the Doctor-Side mobile application is not deployed, but is in development. The function will be the same as the PHP website (See Section IX, PHP BACKEND), just on a mobile device to allow doctors and medical personnel to visualize and modify patient data on the go.

1. AMAZON EC2 INSTANCE

The backend server is run off of an Amazon EC2 instance. This is a free-tier server that allows us to create projects and databases quickly and easily. The EC2 instance currently contains a PHP project under Amazon’s Elastic Beanstalk platform, which allows for quick deployment and modification of a simple website (which will allow us to talk to the backend and retrieve data from the database). It also holds an instance of an Amazon RDS database, which is MySQL based and allows us to easily communicate and retrieve any patient’s information. The EC2 instance was not difficult to set up, but did require a few steps to get working.

1. *Security*

Security is a big deal for AWS servers, and Security Groups are blocks of IP addresses that are allowed or restricted access to certain parts of the server. For example, all addresses (0.0.0.0/0) are allowed to access the HTTP port 80 in order to view the website, while the database is limited to access only from within the server instance’s static IP address. Each security group is given an ID, a group name, and is assigned a VPC (Virtual Private Cloud).

1. *Virtual Private Cloud*

A VPC is a sub-cloud inside of the Amazon Web Service’s public cloud. It is an isolated network such that other servers cannot see these instances. VPCs intentionally make sure specific instances are not visible to the outside except through a single point, which is far more secure than a non-VPC server. A VPC is given a subnet mask that will identify all the instances within the said VPC. A VPC is given DHCP options, a gateway, and most importantly, subnets and routing tables.

1. *Subnets*

Subnets are sub-networks within a VPC. EC2 instances will belong to a subnet, and cannot be made without one. Instances will have an IP being the same as the subnet, but in the range of 2 to 254. Routing tables are located inside subnets, and network traffic inside a subnet is controlled by the routing table. It can destine traffic directly to a certain IP or make it go around a different path. There is only one routing table per subnet.

A VPC is not immediately accessible, because a reusable public IP is needed to be attached to allow access to the greater Internet. Another possibility for access is through a NAT gateway instance acting as the point for outside traffic, which allows for any subnet with this routing table to become private because they cannot be exposed publicly. Even an elastic (reusable) IP will not allow access, as a normal routing table would till be required.

1. *Dashboard*

All of this is easily accessible through a user-friendly dashboard that lets you view all of these. Our project’s server contains 3 running instances with 2 elastic IP’s, 3 volumes, 1 load balancer, and 7 security groups. Users are able to view all of these and modify them as they see fit. They can monitor the service health and even launch new instances. We have a specific instance called “networking-csce513” that is a micro instance (as there is not heavy traffic to this kind of system), which contains an Elastic Beanstalk project running on a Web Server Environment Tier. The project is PHP based, which is easily updateable as all a user needs to do is upload and deploy their PHP project.

1. PHP BACKEND

The PHP backend is an actual website that acts as the doctor and patient’s visualization system, with doctors being able to view and modify data as they see fit, along with being able to change conditions of the studies that are being performed. The site is a Web Tier Environment operating on a single instance (which means it’s not scalable), and is on a micro instance for testing. The site is easily updated by just uploading and deploying the PHP project, and then the server restarts, deploying the new version to the instance and checking the health of the environment.

The site itself is also connected to an RDS database, which contains all of the patient data, doctor data, and the monitor data. The site uses these connections to view and modify them through a series of PHP files designed just for the task.

The site is currently deployed to a default Amazon link at networking-csce513.us-east-2.elasticbeanstalk.com, and users will be immediately presented with a login page (Figure 8). As of the time of this paper, the website was still under development, and was a key source of errors (See Section X, ERRORS AND ISSUES).

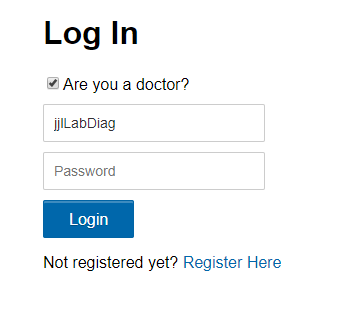


Figure 8: Log In Screen

The user is then presented with a screen similar to Figure 9, where they are able to View their patients, modify data, or log out. The doctors have the ability to modify the patient data, while patients simply just view their own straight from the page. In Version 2, which is under development, they will be able to visualize data in charts, and see their data in a user-friendly manner.



Figure 9: Home Screen

If a user does not exist on the system, they have the option to register and add their information to the database. They can choose between doctor registration or patient registration, but more security is needed to avoid non-medical personnel registering as doctor’s in the system.

In the view page, doctors can see all of the patients that are under their name. They have the ability to edit the data exposed to them, and even remove patients from the system.

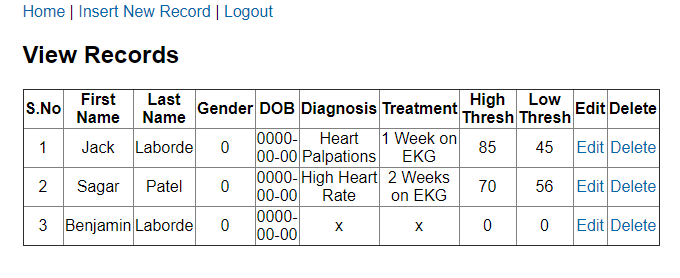


Figure 10: View Records

1. ERRORS AND ISSUES

The project did indeed have its fair share of issues, with most of them being on the web end. Fortunately, most errors have been fixed by the time of this paper.

1. *Arduino*

The Arduino project fortunately ran without a hitch, the only minor issue being data separation initially. This was solved by adding a space after a transmission, and the mobile application will manually split the data to add to a list.

1. *Mobile Application*

The mobile application only had one true error, which resulted from earlier failed testing that resulted in a bad list and making bad comparisons between an empty string and float values.

1. *Web Application*

The web end was where we had our most serious issues. We suffered from rushed work to get the record viewing to work, and we could not find out why the monitor data for specific patients was not showing up. Fortunately, though, these issues were resolved.

1. FUTURE

This project could have a substantial future if it keeps getting worked on. Features that could develop are geolocation for patients in crisis, routing to the nearest hospitals, more data collection and visualization, and much more. This project was overall successful, and could develop into a viable commercial product.

REFERENCES

[1] L. Guo-cheng and Y. Hong-yang, “Design and implementation of a Bluetooth 4.0-based heart rate monitor system on iOS platform,” *IEEE Xplore*, 13-Mar-2014. [Online]. Available: http://ezproxyprod.ucs.louisiana.edu:2055/document/6765297/. [Accessed: 03-Dec-2017].

[2] S. Rafiq, A. H. M. Z. Alam, and M. R. Islam, “Development of ECG home monitoring system,” *IEEE Xplore*, 19-Oct-2017. [Online]. Available: http://ezproxyprod.ucs.louisiana.edu:2055/document/8069119/. [Accessed: 03-Dec-2017].

[3] C. Stobing, “What is WiFi | How Does It Work | What Does WiFi Stand For,” *Gadget Review*, 30-Dec-2016. [Online]. Available: http://www.gadgetreview.com/what-is-wifi-what-does-wifi-stand-for-how-does-it-work. [Accessed: 04-Dec-2017].

[4] R. Dey, “AWS - VPC Networking for Beginners - DZone Cloud,” *DZone*, 17-Aug-2012. [Online]. Available: https://dzone.com/articles/aws-vpc-networking-beginners. [Accessed: 04-Dec-2017].