ASHESI UNIVESITY COLLEGE

MOBILE HEALTH BASED INFORMATION SYSTEM

DANIEL KWABENA OWUSU ANKOMAH

2014

Applied Project

ASHESI UNIVESITY COLLEGE

MOBILE HEALTH BASED INFORMATION SYSTEM

By

DANIEL KWABENA OWUSU ANKOMAH

Dissertation submitted to the Department of Computer Science

Ashesi University College

In partial fulfillment of Science degree in Computer Science

April 2014

Applied Project

Declaration

I hereby declare that this dissertation is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:………………………………………………………………………….

Candidate's Name:……………………………………………………………………

Date:………………………….

I hereby declare that the preparation and presentation of the dissertation were supervised in accordance with the guidelines on supervision of dissertation laid down by Ashesi University College.

Supervisor's Signature:……………………………………………………………………..

Supervisor's Name:…………………………………………...…………

Date:………………………………….

Acknowledgement

I would like to thank God for giving me the strength, peace, and resolve to undergo this project. I also want to extend my deepest gratitude to my Supervisor, Mr. Aelaf Dafla, for all the guidance, support, and time he dedicated to ensuring that this project was a success, I could not have done it without him.

I would also like to thank my family (Barbara, Richard, Sophia, Samuel, Chris, Quendolyn, and Harriet) for sponsoring my education, encouraging me in the toughest times, and providing me with the guidance and care I needed for this project. I could not have gotten this far without them.

Furthermore, I would like to thank Dr. Amanquah for the additional guidance he provided on this project. Finally, I would like to thank my colleagues (Chloe Acheampong, Kelvin Wellington, Antoinette Doku, and Henry Olletey) for helping in testing and reviewing this application.

Abstract

Healthcare delivery continues to be a major focus for Ghana as it strives to reach the objectives outlined in the Millennium Development Goals [1]. A major roadblock to reaching these goals has been the average long distances between the rural areas and the healthcare provision centers in the country. In order to overcome this hurdle, the Ministry of Health decided to train healthcare officers who live within these rural areas to help improve the situation.

The health officers need an effective way to communicate or relay information back to the district office without having to travel to the District office regularly. They also occasionally need assistance in determining the best way to approach or cater to the emerging health needs of the communities. mHealth Knowledge is a Mobile Health based information system that provides functionality to allow for the sharing of information between community health officers and healthcare professionals at the district office.

Contents

Chapter 1: Introduction

1.1 Introduction

Mobile technologies have over the years increasingly become a part of our daily activities and have evolved to provide solutions to some of the challenges faced in industries and governments. These challenges include but are not limited to issues of data storage, synchronization, and communication. According to the "*International Telecommunication Union there are close to 5 billion mobile phone subscriptions in the world, with over 85% of the world's population now covered by a commercial wireless signal*" [2]. These figures hint at the immense potential of mobile technologies as a solution to the challenges being faced within the different industries because solutions devised by such mechanisms have the potential of reaching a wide audience.

The advent of such technologies inspired a new industry known as e-Health in 1999 [3]. "E-Health involves the use of information and communication devices for health services and information." As the e-Health sector evolved, a sub-segment of e-Health coined mHealth emerged as a viable candidate for solutions to health care provision in developing countries. M-Health involves the usage of mobile devices and technologies such as "computers, mobile phones, communications satellite, patient monitors, etc." [4] *A recent study by the WHO, World health Organization, found that "83% of the 112 participating Member states reported the presence of at least one mHealth initiative in their country"* [2]. This further solidifies its status as a potentially strong solution provider. The aim of this project is to develop a Mobile Health based information system to aid in the effective communication and sharing of data between Community Health Officers and the District health office.

1.2 Background

Ghana is not outdone when it comes to the field of mHealth as it has gradually sought to use mobile technologies to provide solutions to the problems it faces in the health sector. The health sector in an effort to get closer to the Millennium Development Goals (MDG), is gradually undergoing various mHealth projects to improve the health sector [3]. These projects include:

- Motech (mobile technology for maternal and child survival with mobile phones) by Grameen foundation
- SMS for Life (monitoring malaria commodities in health institutions with mobile phones) led by Novartis
- EWS (early warning system, monitoring commodities of malaria, family planning, and HIV with mobile phones)
- Sene Project (using both PDAs and mobile phone to report public health activities) led by Participatory Development Associates in conjunction with the ministry of health.
- Millennium development mhealth project (using mobile phones to support rural health)

These projects have been spurred on by the increasing emphasis placed on the delivery of health care on the community level [1]. This has over the years been provided by Community Health Officers (CHO) as a result of the long distances between the residents of the rural areas and health care providers with estimated average distances of 8km [5]. The CHOs provide communities with basic health service including health education to community members by living in the communities. They also provide a valuable health surveillance service to the Ghana Health Service by reporting back their observations to the district health office.

These officers keep health records on paper. This leaves the health records vulnerable to data loss due to weather damage, or loss of records during transit from the communities to the district health office. Furthermore, this also means that assessment of the health situation within the districts can only be conducted when the Health Officers move from the communities to the District Office. Also, keeping records on paper creates aggregation which makes it difficult to perform data analysis in reasonable time. It is apparent that a better means of record management and communication has to be established to cater for the inefficiencies in the current system.

Also, during the course of their operations CHOs sometimes need to communicate with specialists about ways to undergo certain tasks as well as receive training materials to help them improve their abilities as health officials. This involves the administration of care to patients or the provision of health advice to community members. These specialists do not live within the communities and hence can only be reached via phone or directly at the district health office. Furthermore, due to the similarities between communities it is possible to have similar questions thereby increasing the possibility of CHOs asking repeat questions to the specialists at the community health office.

1.3 Objectives

The objective of this project is to create an information system with a tablet PC/ phone application tailored to provide the following functionality to cater to issues of data transfer, communication and synchronization prevalent in the current Health provision system at the District and community levels. The large number of different management units within the health sector generate large amounts of data held in separate places making it difficult for collaboration for a better health care system. This application will thereby provide a means of sharing information and providing real-time assistance to CHOs as they undergo their duties within the districts. These functionalities include:

- Deliver training material to community health officers to improve their skill
- Enable community health officers to interact with specialists at the district level. This involves posting questions to the specialists for answers.
- Transferring routine data collected by CHO to Sub-district health center and district health office without the use of internet.

1.4 Outline of Dissertation

The remainder of this report will provide more details about the underlining details of the system. Chapter 2 will be a section on the design considerations and decisions taken to develop this application. This will be followed by Chapter 3 which deals with the implementation of the system. This section will discuss some of the measures undertaken to develop the entire system. Chapter 4 which follows next will discuss the testing phase of this project and outline the results of undergoing those tests. Chapter 5 will conclude this document by providing information about the challenges faced in developing this system as well as suggest areas for growth for future work.

Chapter 2: Design

This section will give an overview of the design decisions and considerations undertaken to develop this application. The overall system has two main facets namely, the Records and Knowledge branch however this project will be focused primarily on the knowledge branch as the Records branch has already been implemented in prior work undergone by Florence Jones, a former student [6]. The knowledge branch of this system will serve as a portal of resource materials and provide a platform to enable CHOs interact directly with the Health services. The interaction involves CHOs asking questions about the necessary procedures needed to cater to a health need or general health welfare from Health Professionals with specialties in the various fields of healthcare such as Childcare, Nutrition, Diagnosis and Treatment, Pharmaceuticals, etc.

In order to fulfill the requirements of high availability, portability, and information sharing, this application will be developed primarily using the android platform which will work together with a web application for communication between CHOs and the health specialists. The open source nature of android means that source code for the implementation of major parts of the system will be readily available for reference. Also the tools necessary for development are highly accessible and free within the open source community, hence reducing the need for any new purchases. Moreover, developing a web enabled application ensures that communication can take place without requiring both parties to be present at one location at the same time.

2.2 Functional and Non-Functional Requirement

As CHOs perform their duties within their communities they may encounter new situations which require the assistance of a specialist as stated previously. Initially they would have had to call the District office for assistance or in extreme cases commute to the District office for assistance. This application eliminates that need by providing them with a way to post a question and get a response from a specialists who will be positioned behind a computer via the internet. They will also have access to questions posted by other CHOs eliminating the need to request for assistance on issues with answers already provided.

Also, over time new ways of undergoing tasks may emerge. This may require all the CHOs travelling to the District office to receive training. Due to the long distances and bad roads leading to the communities it is cumbersome and inefficient to require all the CHOs to travel such long distances to receive training materials. The application will cater to this challenge by providing a means for CHOs to share training materials, mostly video content, via the Wi-Fi-Direct abilities of the tablet devices. Wi-Fi direct is a technology that allows for high speed data transfer by enabling devices to connect directly to each without joining a traditional hotspot network. This functionality is instrumental to the implementation of this project because poor internet connectivity within the communities limits the ability to download media content such as videos which require large data transfers due to their large sizes.

As stated earlier, there will be a web interface for the specialists to access and respond to questions posted by Community health officers. This interface will be aggregated according to different categories with each specialists receiving questions that pertain to their specialty.

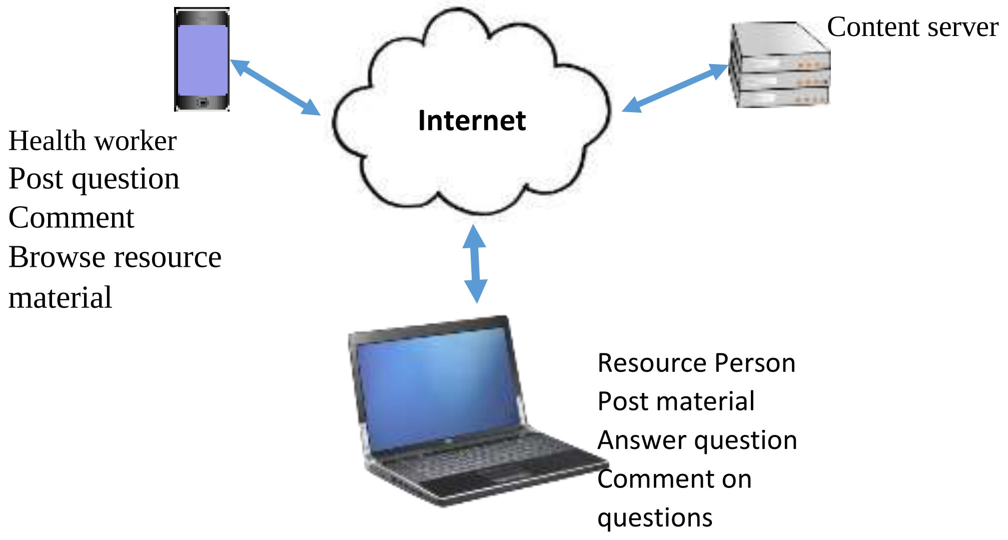


Figure 2.1: Model of system interaction

Due to the unstable nature of the networks in rural areas, the android application has to store materials and questions locally and synchronize with the District office whenever internet connectivity is sufficient.

2.2.1 Non Functional requirement

- Each question should be stamped with a geo location and time
- Users have to login before accessing the system.
- The mobile application should work over unreliable internet connectivity
- The application should be accessible offline

2.2.2 Functional Requirements

There are two perspectives in this project namely the Community health officer (CHO) and Resource Personnel perspectives.

Community Health Officers Perspective

- CHOs should be able to post questions to the system.
- They should be able to choose categories of questions to view
- They should also be able to list and view questions posted by themselves or other CHOs.
- CHOs should be able to see comments and answers to questions
- CHOs should be able to comment on questions and answers
- CHOs should be able to share training materials with other CHOs via Wi-Fi-Direct

Resource Personnel Perspective

- Resource personnel should be capable of selecting categories of questions to view
- They should also be able to list new questions
- They should be able to answer questions though the web portal
- Resource personnel can view, edit, or delete his/her answers
- Resource person can comment on question or answer
- Resource personnel should be able to update CHO tablets with new training material and resources.

2.3 Scenarios

Ama Asante is a 22 year old CHO recently posted to the Berekum community. She is an avid user of technology and enjoys using mobile chat applications like Viber and Whatsapp for communication with her friends. Ama has a strong presence on Facebook and has been using it to reach out to old friends since graduating from high school. After eight months in her previous community, she has grown fond of her role as a CHO and looks forward to executing her duties with precision each day. There is a chronic disease spreading within her community which is affecting children between the ages of 8 and 12. In order to treat this chronic disease, she has to get assistance from the district office about how to engage this disease. She visits the knowledge section of the mHealth application and looks to post a question to health specialists at the district office. Fortunately for her, someone has already asked questions on this subject and she immediately gets assistance for this issue. She is very pleased and browses the other topics to see if she can learn other new content.

Kwame Oduro is a 28 year old senior CHO who moves constantly from the districts to the communities to gather data for the district office from the CHOs residents in the communities. After obtaining recent content he travels to Berekum to gather data on the recent chronic disease from Ama, a CHO in Berekum. Upon arrival, Ama requests for new training materials from Kwame. He remembers receiving new training materials from the district office. Kwame connects his tablet to Ama's tablet and synchronizes the training materials to Ama. He leaves with the data gathered and Ama is able to view the new materials on her tablet.

Ama Biney is a 32 year old senior resource personnel stationed at the District Health Office. She loves to read and is passionate about delivering effective content to CHOs. She is conversant with computers and browses the internet constantly for new material to improve the health processes within the Healthcare Sector. After downloading new video content on the treatment of diaper rash in infants she is eager to share this new content with the CHOs who have come over for their monthly debriefing. With limited time she has to share the content with a few CHOs and hope that they will further share this content between themselves via the Devices they have been provided. She connects one of the tablets brought by Kwame, one of the CHOs, and launches the program developed for synchronizing new content to the tablets. She navigates to the folder on the tablet where all the media files are stored and initiates the synchronizing process. The process finishes in 40 minutes, just in time for CHOs to catch the bus back to their respective communities. She is glad because knows that given the long ride back to the respective communities Kwame will be able to share this media content with the other CHOs.

2.4 Use cases

Outlined in this section are use case diagrams which model the actions that take place within this system.

Community Health Officers

Community Health Officers have varying abilities. They will first be able to launch the application. After launching the application they will have the option to post questions, view questions or view resource materials. They will also have the ability to filter questions by choosing to see only questions they have posted or view all questions. They will also have the ability to show only answered questions or filter the questions by category. This should allow them to dive straight into a more streamlined result set geared towards their specific interests at the time. Also as stated earlier, they might want to share resource material they have with other CHOs. These functions are modeled in figure 2.2 below.

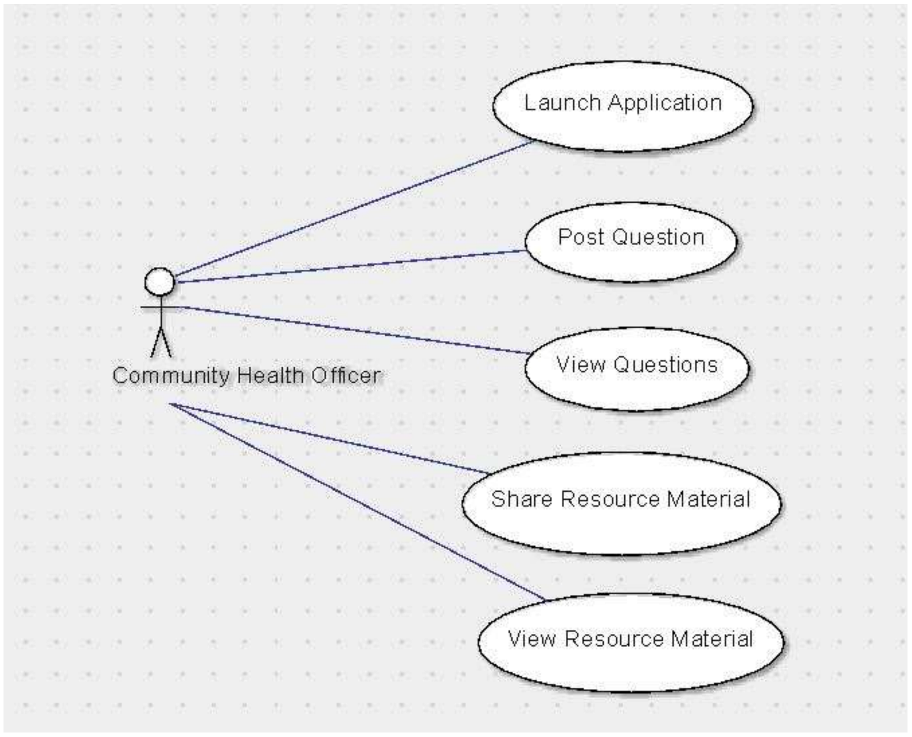


Figure 2.2: Use case diagram of Community Health Officers

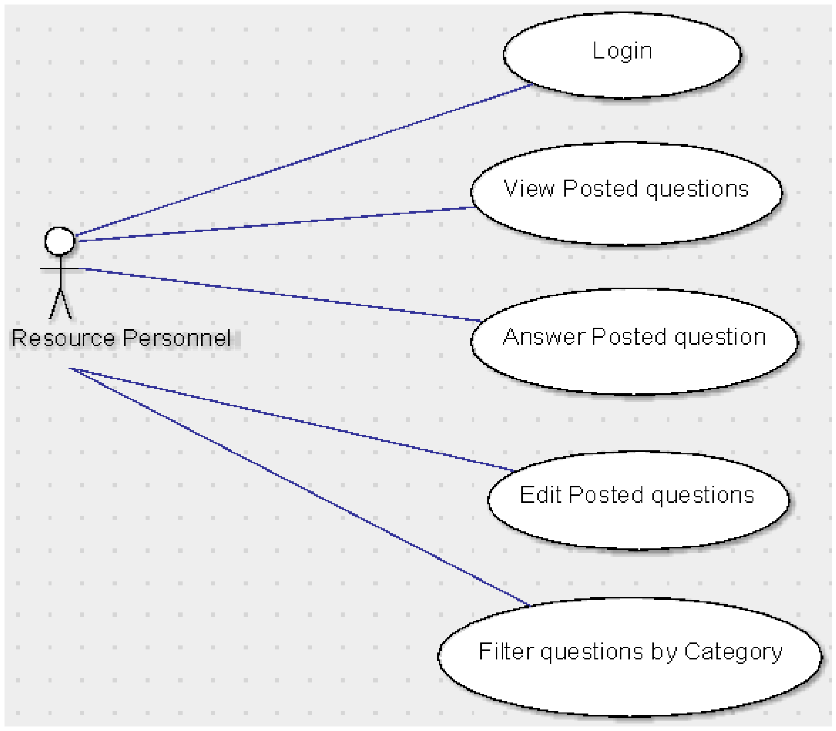


Figure 2.3: Use case diagram of Resource Personnel

2.4 System Architecture

Outlined in figure 2.4 below is a model of the overall system architecture. This model is displayed in a layered pattern which models the flow of data through the system. This architecture includes three layers namely the application user interface, application layer, and data access layer. On the application user interface layer, the Questions, Resources, and Synchronization components are hosted on the Android application accessible by CHOs from their tablet devices. The Answers component is a web page hosted on a web server. Beneath these components are the QuestionsFragment, ResourceFragment, and Synchronize Activity which act as the backend of the user interface. These components interact with the data classes to provide the user with the needed functionality at every instance.

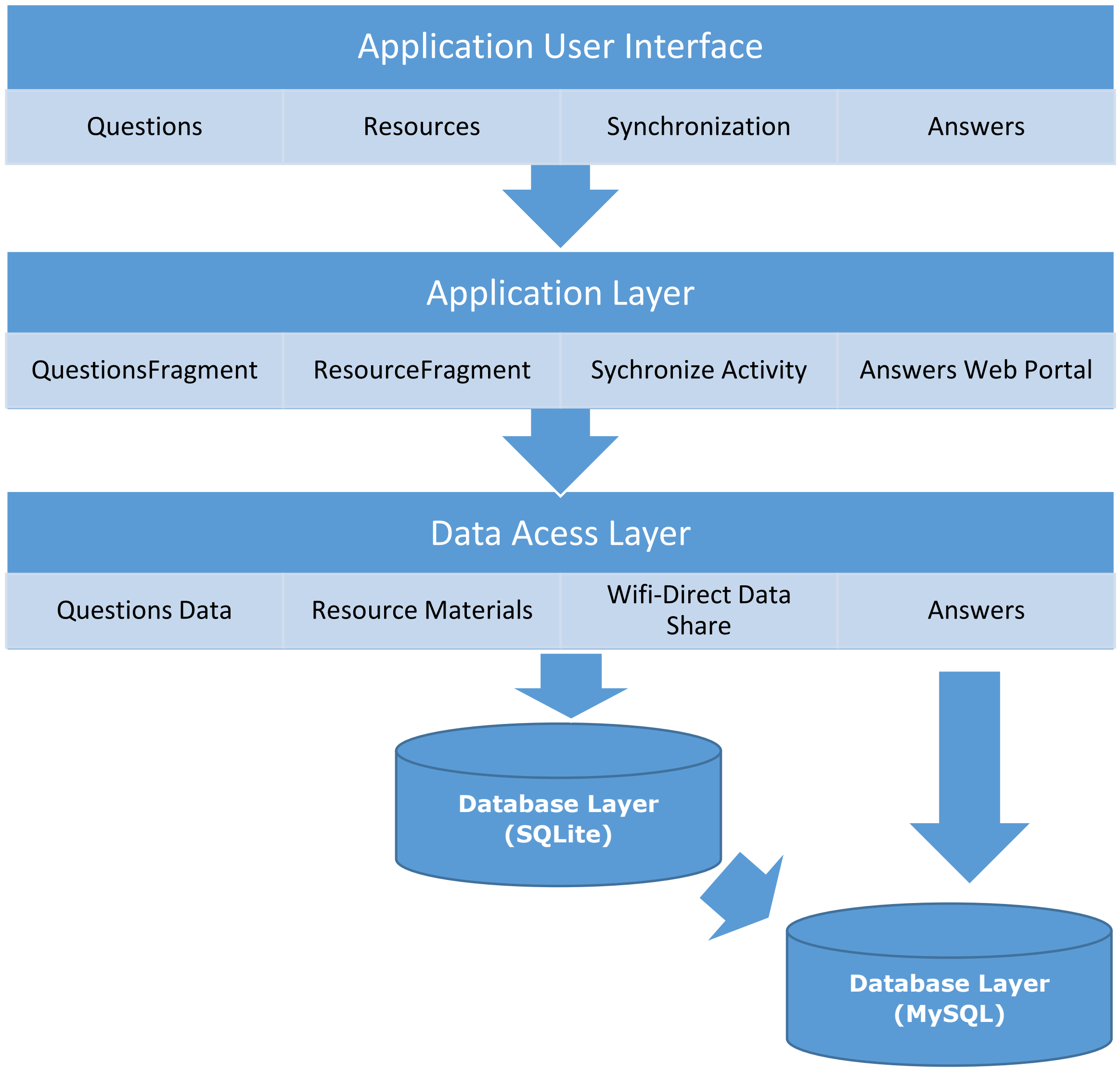


Figure 2.4: System Architecture

2.5 Database Architecture

In order to allow for resources to be available during periods of poor internet connectivity, a local copy of resources has to be kept to facilitate the CHO's operational duties. This was accomplished by deploying a SQLite database for the storage of local resources on the tablets. This local copy would then be synchronized with a MySQL database in a central location upon access to internet connectivity. The database is modeled below in figure 2.5. This will ensure that questions are not lost during periods of poor internet connectivity. Furthermore, this will enable resource personal to gain access and answer questions posted by CHOs after they have synchronized with the central database.

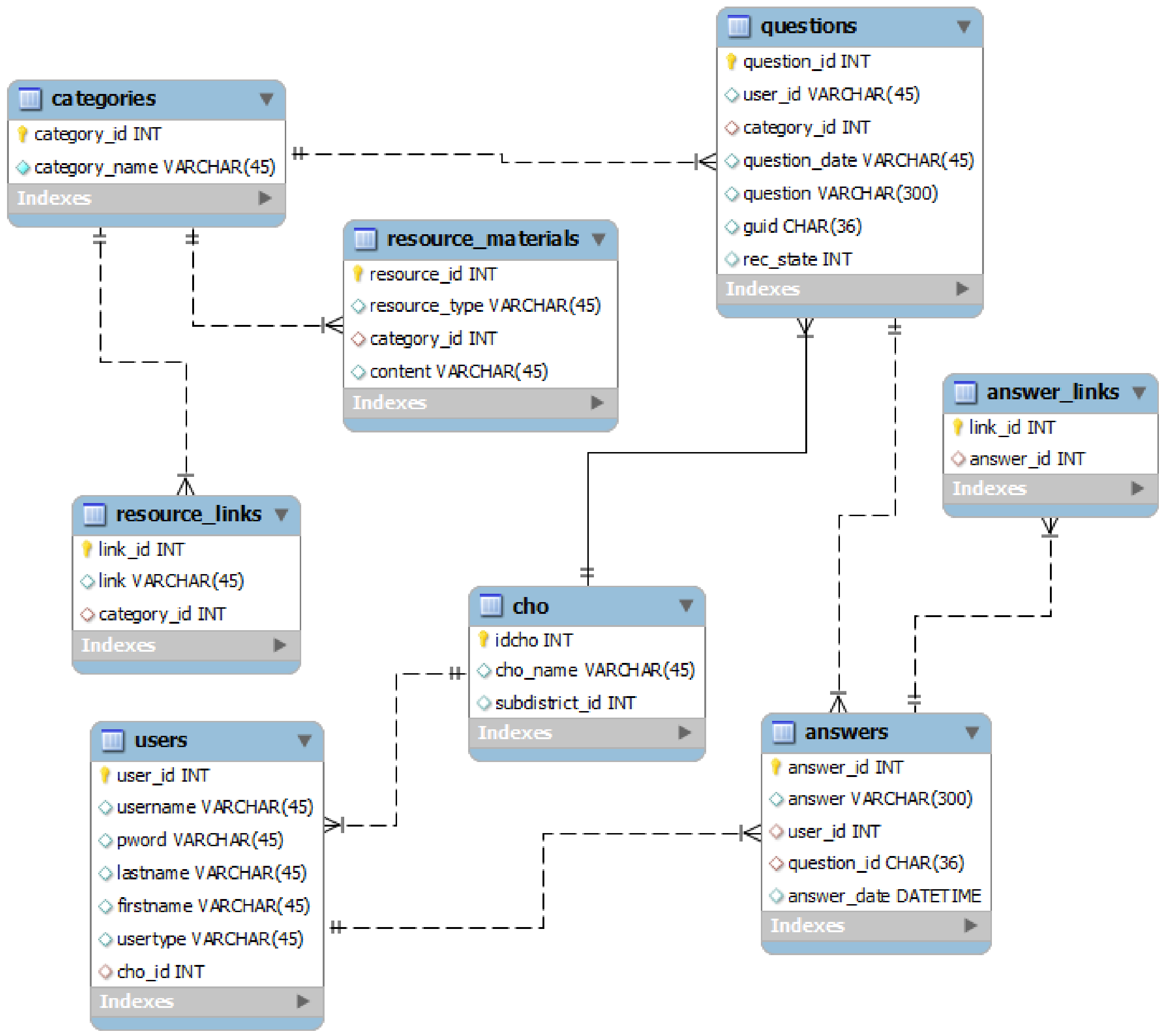


Figure 2.5: Database Design

Unique IDs

The nature of the question activity where CHOs have access to questions posted by other CHOs created a major hurdle because using an auto incremented ID column could lead to multiple questions having the same IDs. This is because locally they are unique i.e. device A can have a question with an ID of 1 locally and likewise device B can also have a different question of ID 1. This becomes a major problem because when they are synchronized with the remote database they cause a conflict. To solve this problem, the application uses Java's Universally Unique ID (UUID) class to generate globally unique IDs which ensures that there are no duplicate IDs across all devices.

2.6 Mockups

Prior to implementing this application, the system was modeled using a wire framing tool to get a better understanding of the flow of the system before delving deeper into the full implementation of the system. The model was developed using Indigo Studios after consultation with Mr. Aelaf Dafla, the project supervisor.

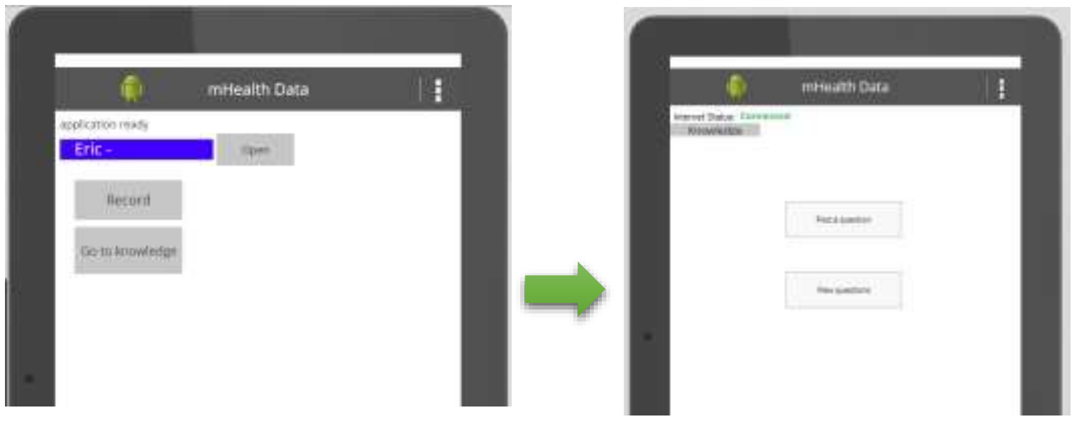


Figure 2.6: Mockup of the home screen of the knowledge activity

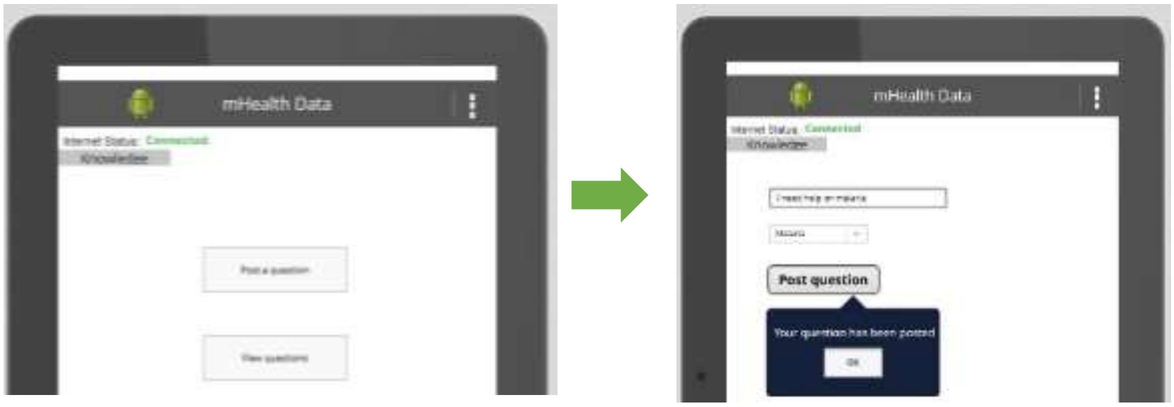


Figure 2.7. Mockup of posting questions

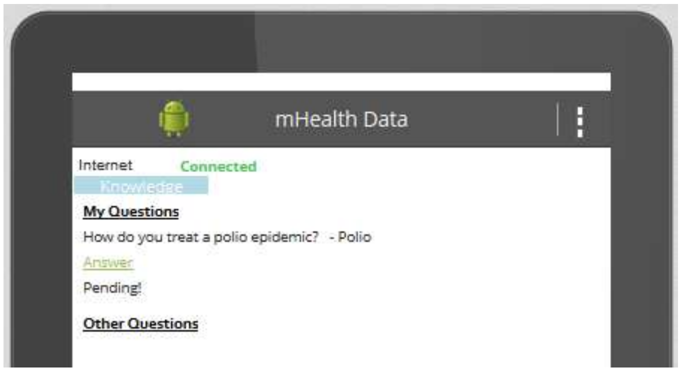


Figure 2.8: Mockup of the process of viewing questions

Chapter 3: Implementation overview

The final implementation of the system, varies slightly from the interface developed with the wire-framing tool because the mockups developed were mainly used to model the flow of the application.

User Interface: Mobile Application.

When the resource personnel launches the mHealth Android application they will have the choice to navigate to either the Records section of the overall system or the Knowledge section of this system as shown below in figure 3.1.

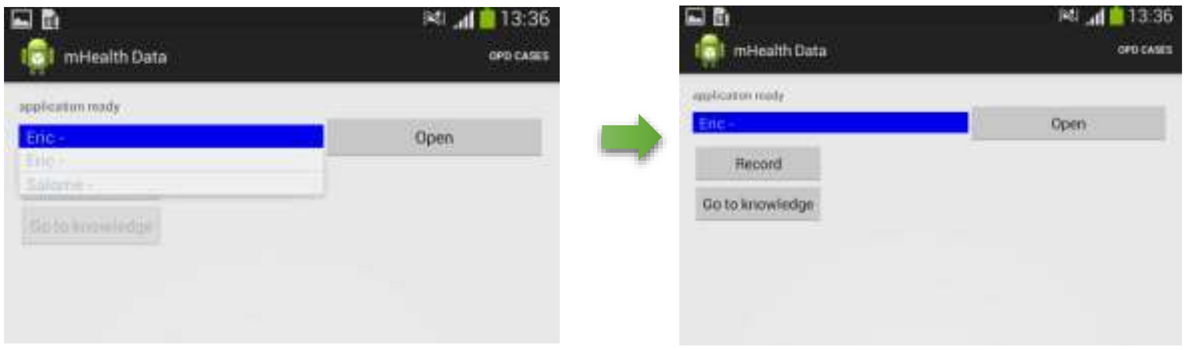


Figure 3.1: Main screen of the CHOs

After selecting the knowledge branch they are transitioned to another activity which allows them to either view and post questions or view available resource materials as shown in figure 3.2. This was implemented using a main activity, KnowledgeActivity, with two Fragments namely the QuestionsFragment and ResourceFragment both connected with a custom FragmentPagerAdapter to allow the user to easily swipe between the question and resource Fragments.

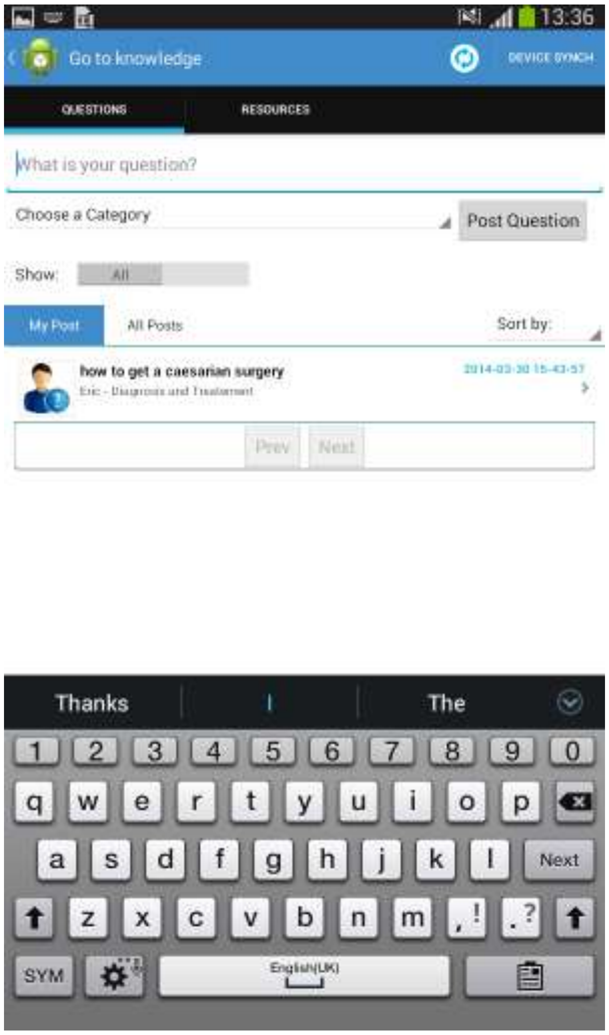


Figure 3.2: Knowledge Activity for posting questions

When the user navigates to the QuestionsFragment, a Listview with a custom adapter for the arrangement of data is used to display the list of posted question to the user as displayed in figure 3.2 above. The data for the questions is pulled via a Question and Answer class which are extensions of a SQLiteOpenHelper class. In order to post questions, CHOs will have to first type their question in the EditText field and then select the category that the question falls under from the dropdown menu and post the question. This functionality happens in the QuestionsFragment. When the user taps the Post Question button, an event listener triggers a function which extracts the input data and inserts it into the SQLite database via a Questions class. In order to differentiate between new questions, edited questions, and uploaded questions a record state column is included in the questions class which can be altered to indicate the status of a question. Until a question is uploaded to the District Office for answering, CHOs will have the ability to edit the questions appropriately.

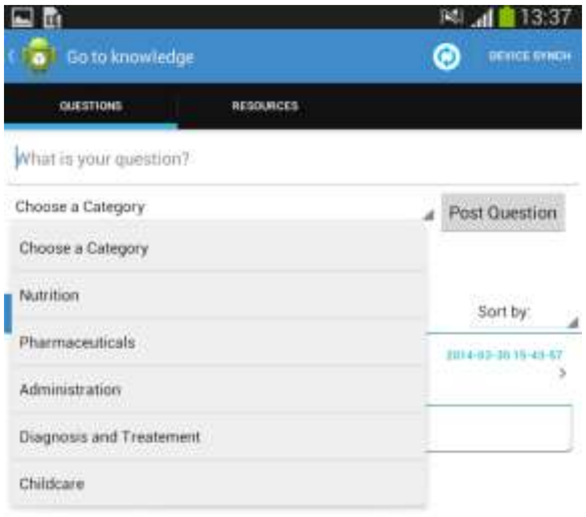
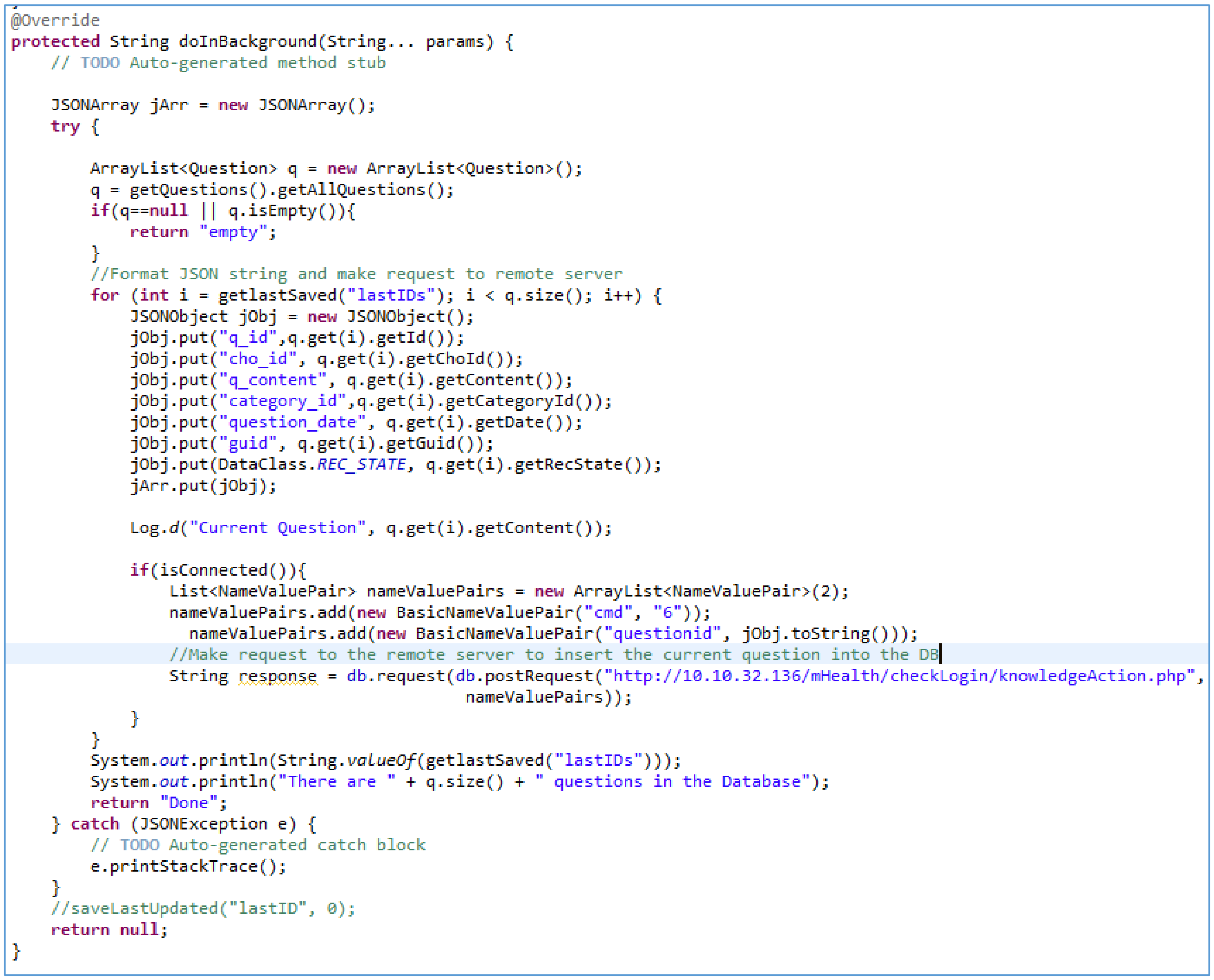


Figure 3.3: Dropdown menu for choosing a category when posting a question

3.1.1 Synchronizing questions with the central remote database

When questions are posted by CHOs on their tablet devices via the mHealth application, they are initially stored locally in the SQLite database on the devices pending internet connectivity for synchronization to the remote database as shown in figure 3.4. Upon available internet connectivity, the new questions are extracted from the database and formatted as JSON strings and sent to the web server via an HTTP POST request which inserts the questions into the remote database as shown in figure 3.5. Likewise, the answers posted by resource personnel are retrieved via an HTTP GET request when internet connectivity is available.

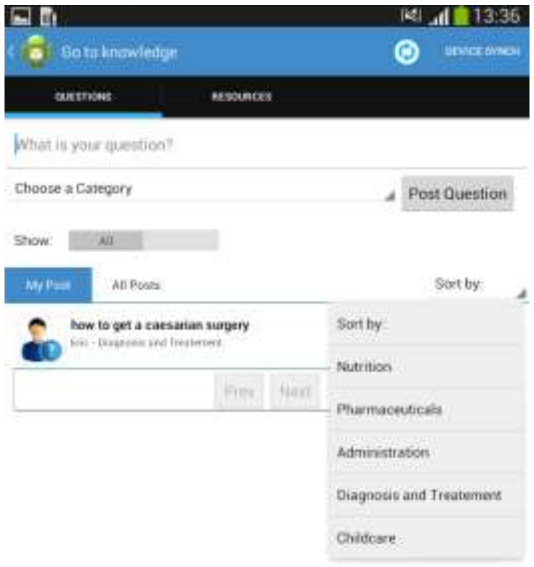


```
@Override
protected String doInBackground(String... params) {
    // TODO Auto-generated method stub

    JSONArray jArr = new JSONArray();
    try {

        ArrayList<Question> q = new ArrayList<Question>();
        q = getQuestions().getAllQuestions();
        if(q==null || q.isEmpty()){
            return "empty";
        }
        //Format JSON string and make request to remote server
        for (int i = getlastSaved("lastIDs"); i < q.size(); i++) {
            JSONObject jObj = new JSONObject();
            jObj.put("q_id",q.get(i).getId());
            jObj.put("cho_id", q.get(i).getChoId());
            jObj.put("q_content", q.get(i).getContent());
            jObj.put("category_id",q.get(i).getCategoryId());
            jObj.put("question_date", q.get(i).getDate());
            jObj.put("guid", q.get(i).getGuid());
            jObj.put(DataClass.REC_STATE, q.get(i).getRecState());
            jArr.put(jObj);

            Log.d("Current Question", q.get(i).getContent());

            if(isConnected()){
                List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(2);
                nameValuePairs.add(new BasicNameValuePair("cmd", "6"));
                  nameValuePairs.add(new BasicNameValuePair("questionid", jObj.toString()));
                //Make request to the remote server to insert the current question into the DB
                String response = db.request(db.postRequest("http://10.10.32.136/mHealth/checkLogin/knowledgeAction.php",
                                              nameValuePairs));
            }
        }
        System.out.println(String.valueOf(getlastSaved("lastIDs")));
        System.out.println("There are " + q.size() + " questions in the Database");
        return "Done";
    } catch (JSONException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    //saveLastUpdated("lastID", 0);
    return null;
}
```

Figure 3.4: Code Snippet for synchronization to the remote database

```
/**
 * Submit the question to the database
 */
private void postQuestion() {
    //Retrieve the posted question from the user
    Category selectedCat = cat.get(spinner.getSelectedItemPosition() - 1);
    db.addQuestion(1, question.getText().toString(), currentCHO.getId(),
                selectedCat.getID(), "", "", DataClass.REC_STATE_NEW);

    //Give the user feedback
    Toast.makeText(getActivity(),selectedCat.getID() + " Your question has submitted under "
                + selectedCat.getCategoryName(), Toast.LENGTH_SHORT).show();

    //Reset the spinner and question
    refreshData(onlyAnswered);
    spinner.setSelection(0);
    question.setText("");
    //new Synchronize().execute();

}
```

Figure 3.5: Code Snippet for saving a posted question locally



how to get a caesarian surgery
Eric - Diagnosis and Treatement

Figure 3.6: Dropdown menu to filter posted questions by category

The dropdown menu for sorting questions is implemented using a spinner which is attached to an event listener which calls a function to update the list of questions based on the currently selected category as shown in figure 3.6.

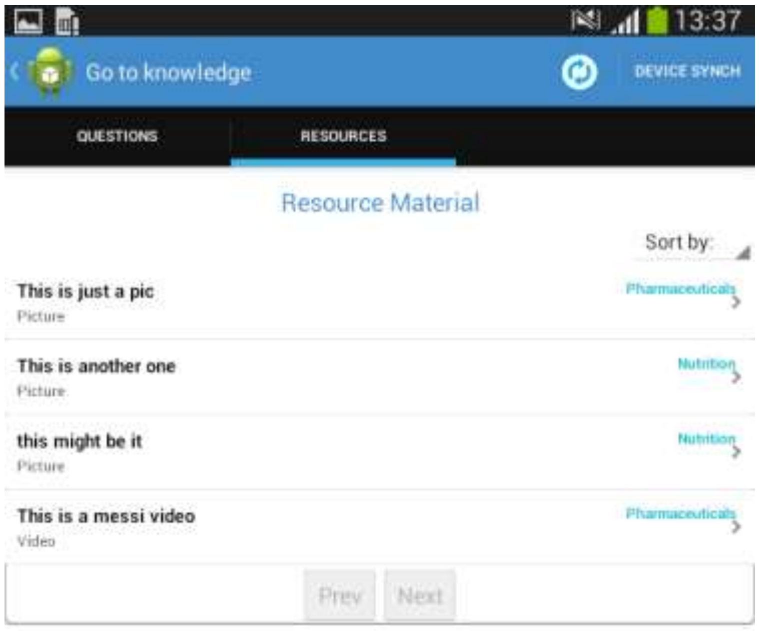
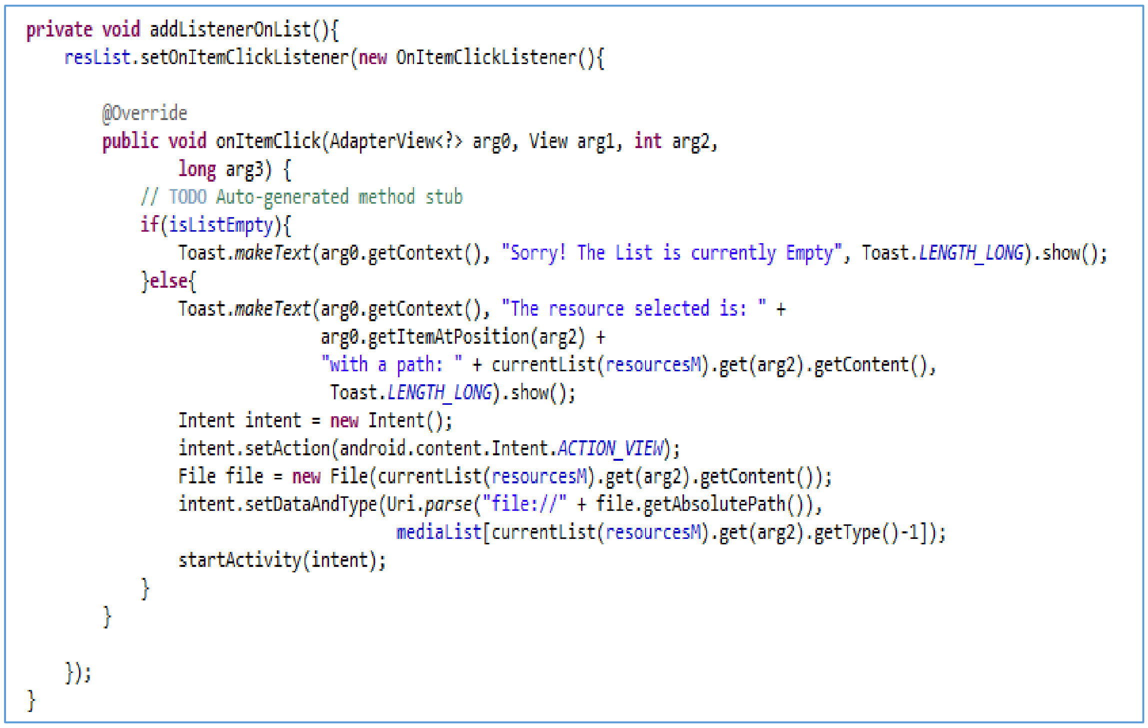


Figure 3.7: Resource Activity for viewing resource materials

When a user selects a resource from the list of resources as shown in figure 3.7, based on the type of resource an activity is started that would retrieve the path to the resource from the ResourceMaterials class and display the resource based on its type. The type of resource material includes HTML pages, videos, pictures, as well as tutorials. Figure 3.8 is a code snippet showing how the ResourceFragment initiates an activity to handle a request to view a resource material.



```
private void addListenerOnList(){
    resList.setOnItemClickListener(new OnItemClickListener(){

        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                long arg3) {
            // TODO Auto-generated method stub
            if(isListEmpty){
                Toast.makeText(arg0.getContext(), "Sorry! The List is currently Empty", Toast.LENGTH_LONG).show();
            }else{
                Toast.makeText(arg0.getContext(), "The resource selected is: " +
                               arg0.getItemAtPosition(arg2) +
                               "with a path: " + currentList(resourcesM).get(arg2).getContent(),
                                Toast.LENGTH_LONG).show();
                Intent intent = new Intent();
                intent.setAction(android.content.Intent.ACTION_VIEW);
                File file = new File(currentList(resourcesM).get(arg2).getContent());
                intent.setDataAndType(Uri.parse("file://" + file.getAbsolutePath()),
                                   mediaList[currentList(resourcesM).get(arg2).getType()-1]);
                startActivity(intent);
            }
        }

    });
}
```

Figure 3.8: Code snippet for accessing resource materials

In order to share resource materials with other peers, CHOs have to select the "synch device" option from the Action Bar (menu). This will transition into the device synchronization screen where the CHOs can search for nearby devices.

3.1.2 Synchronizing resource materials between tablets

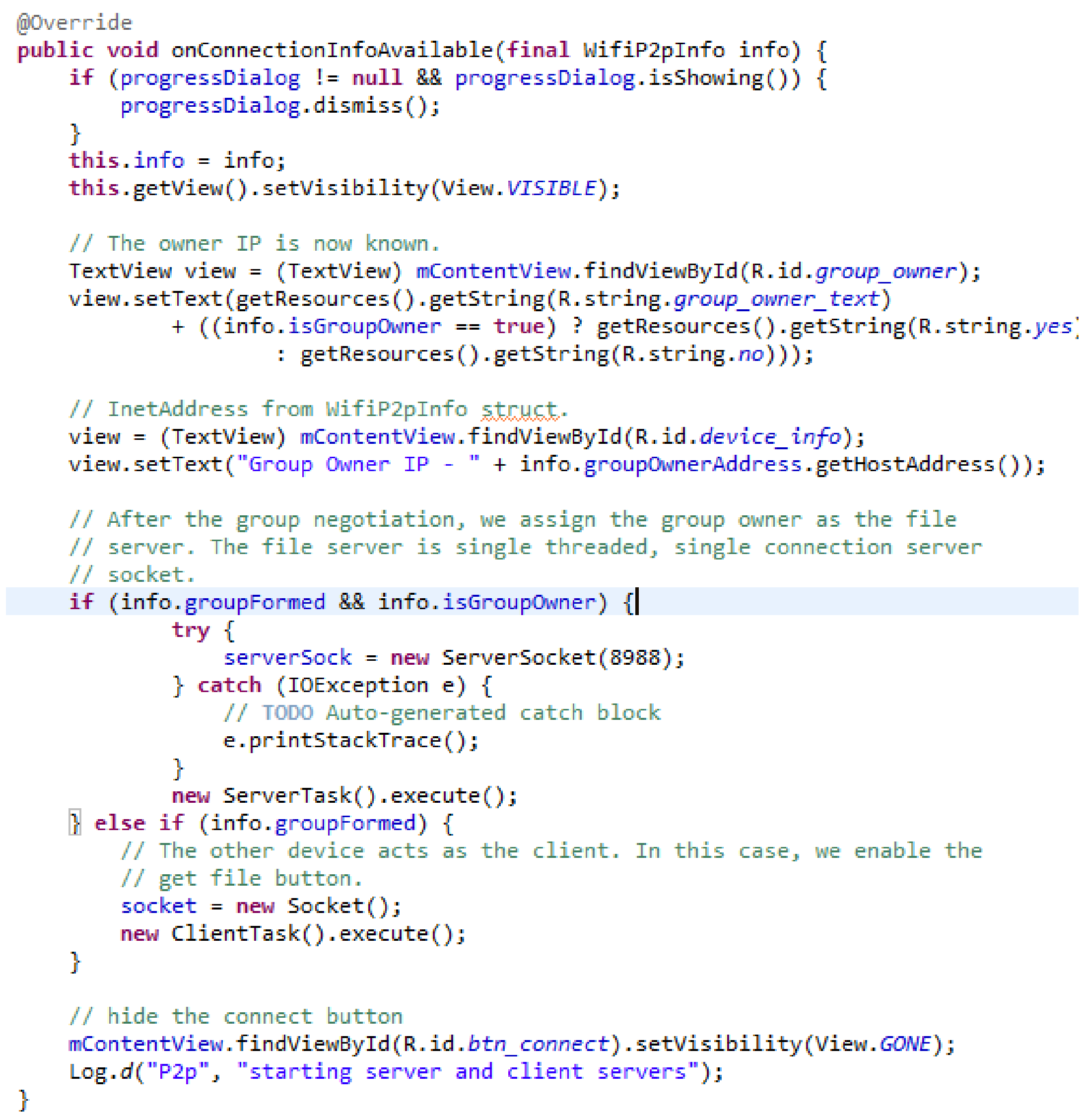
A major challenge faced in implementing this system is the transfer of resource materials to the tablets in use by CHOs. This is particularly difficult because in light of the poor connectivity and high cost associated with internet in the rural areas, transferring huge volumes of data via the internet is not a viable option. This challenge was amplified primarily because the resource materials include videos which are predominantly large in size. Hence other means of data transfer had to be considered for populating the tablets with resource materials which would be made accessible within the application. In order to overcome this challenge a few things were taken into consideration.

These considerations include:

- The mode of data transfer has to be cost effective
- This mode of transfer has to allow for high data transfer in a quick and efficient manner.
- The data has to be sequential i.e. video 1 should be the same across all devices.

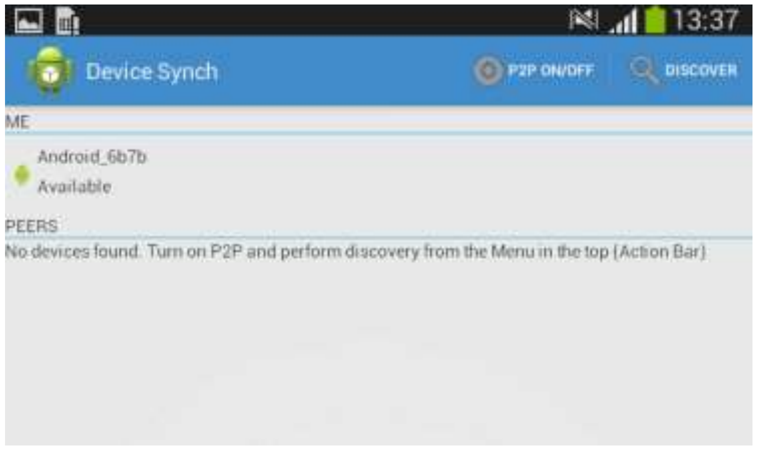
The Wi-Fi Direct abilities of the devices were leveraged to adhere to these considerations whilst overcoming the challenge outlined above. Wi-Fi Direct as stated earlier enables the connection of devices without an access point or a middle layer. This enables devices to be able to interact in a peer to peer (P2P) paradigm. The steps involved in implementing this paradigm are as follows:

1. Establish a P2P connection between peer devices. This involves establishing a group owner of the Wi-Fi direct connection whose IP address is known to everyone. This enables devices to then establish a route for data transfer using a socket address, a combination of an IP address and port number. This is outlined in the code snippet in figure 3.9.



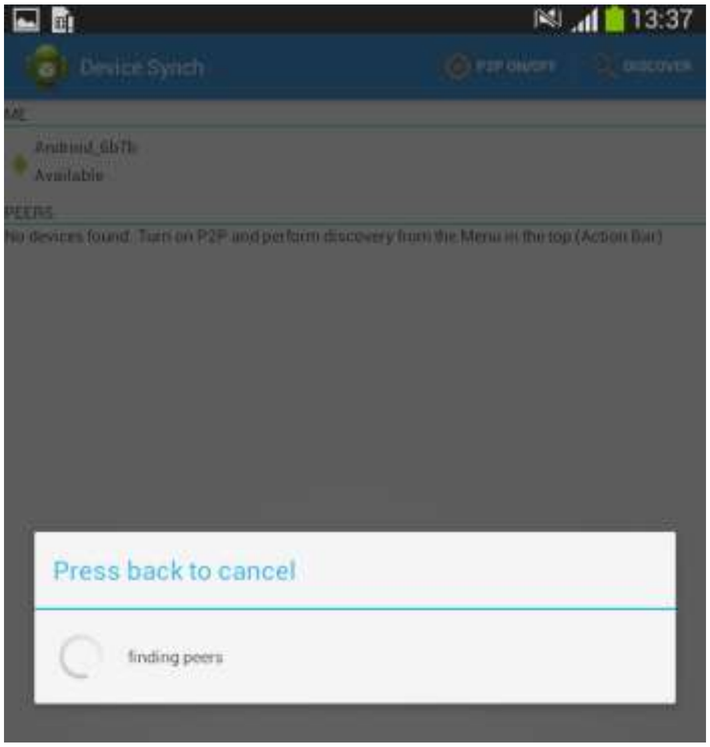
```
@Override
public void onConnectionInfoAvailable(final WifiP2pInfo info) {
    if (progressDialog != null && progressDialog.isShowing()) {
        progressDialog.dismiss();
    }
    this.info = info;
    this.getView().setVisibility(View.VISIBLE);

    // The owner IP is now known.
    TextView view = (TextView) mContentView.findViewById(R.id.group_owner);
    view.setText(getResources().getString(R.string.group_owner_text)
            + ((info.isGroupOwner == true) ? getResources().getString(R.string.yes
                    : getResources().getString(R.string.no)));

    // InetAddress from WifiP2pInfo struct.
    view = (TextView) mContentView.findViewById(R.id.device_info);
    view.setText("Group Owner IP - " + info.groupOwnerAddress.getHostAddress());

    // After the group negotiation, we assign the group owner as the file
    // server. The file server is single threaded, single connection server
    // socket.
    if (info.groupFormed && info.isGroupOwner) {
            try {
                serverSock = new ServerSocket(8988);
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            new ServerTask().execute();
    } else if (info.groupFormed) {
        // The other device acts as the client. In this case, we enable the
        // get file button.
        socket = new Socket();
        new ClientTask().execute();
    }

    // hide the connect button
    mContentView.findViewById(R.id.btn_connect).setVisibility(View.GONE);
    Log.d("P2p", "starting server and client servers");
}
```

Figure 3.9: Code snippet for establishing a Wi-Fi direct partnership



No devices found. Turn on P2P and perform discovery from the Menu in the top (Action Bar)

Figure 3.10: Device synchronization Activity


13:37

Device Synch

P2P ON/OFF

DISCOVER

ME

Android_6b7b

Available

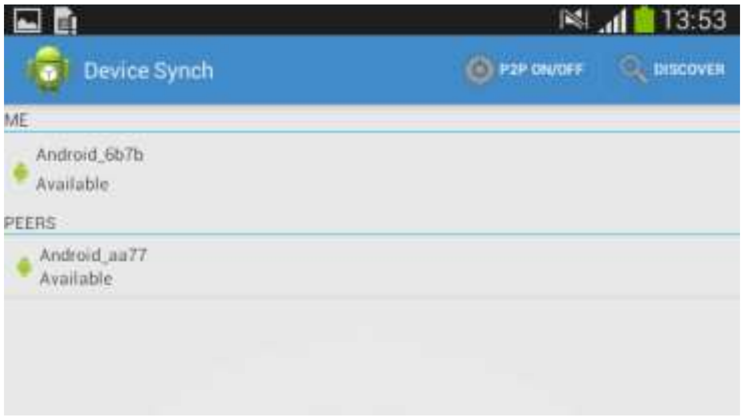
PEERS

No devices found. Turn on P2P and perform discovery from the Menu in the top (Action Bar)

Press back to cancel

finding peers

Figure 3.11: Progress Dialog for finding nearby peers.


13:53

Device Synch

P2P ON/OFF

DISCOVER

ME

Android_6b7b

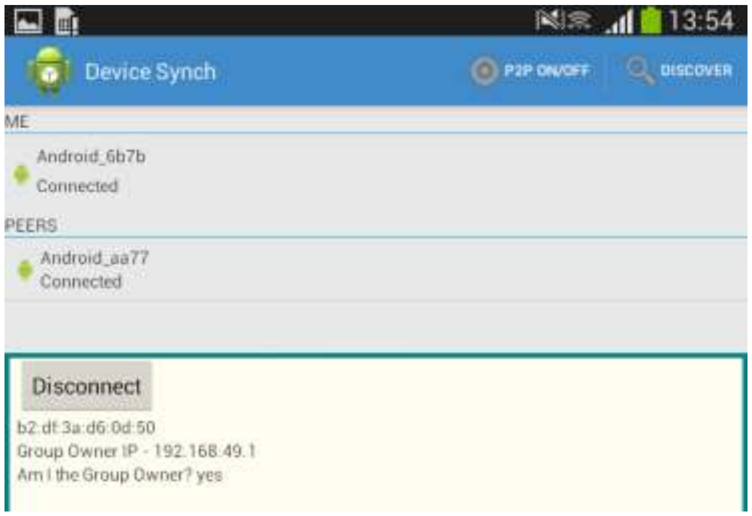
Available

PEERS

Android_aa77

Available

Figure 3.12: Display available peers for Data sharing



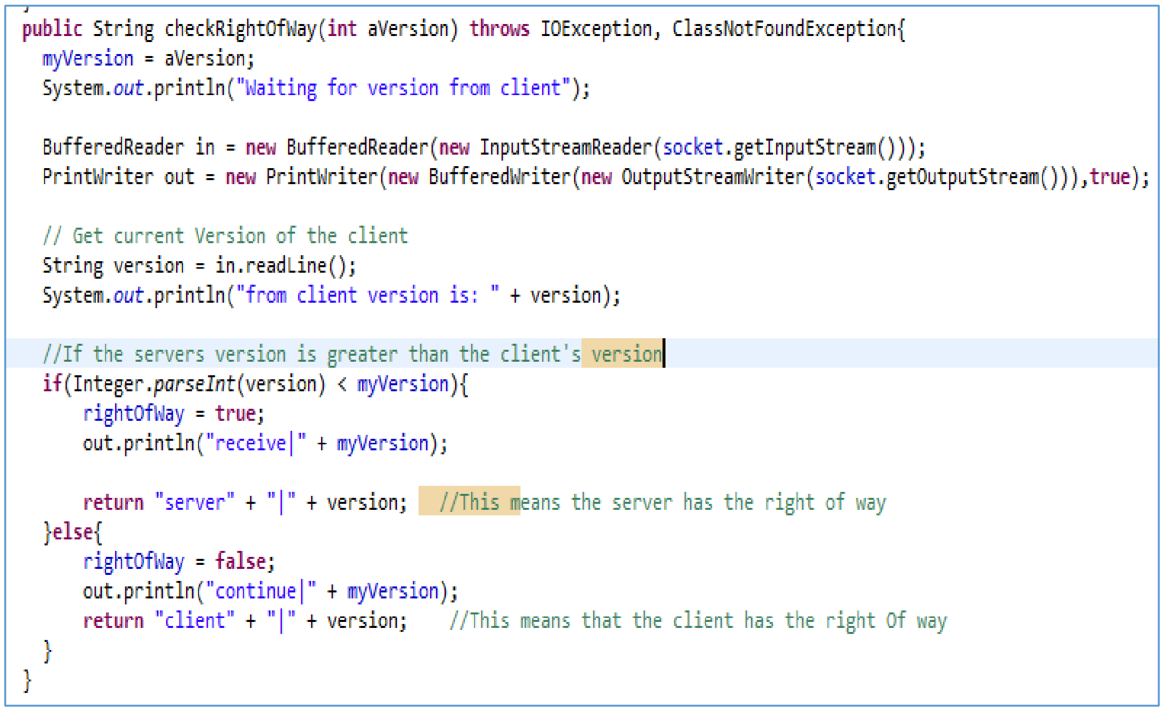
Android_6b7b
Connected

Android_aa77
Connected

b2:df:3a:d6:0d:50
Group Owner IP - 192.168.49.1
Am I the Group Owner? yes

Figure 3.13: Connected state of devices

2. Negotiate between devices to determine which device has the "right of way" to send data and which device will receive data. This involves the devices exchanging the current version number of their respective resource materials to each other in this case whichever device acts as a client will send the server its version number and the server will respond with the server's version number as well a message as to whether the client is to do the sending or the receiving as shown in figure 3.14. That is the server decides which of the devices will provide the latest data.



```java
public String checkRightOfWay(int aVersion) throws IOException, ClassNotFoundException{
  myVersion = aVersion;
  System.out.println("Waiting for version from client");

  BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
  PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())),true);

  // Get current Version of the client
  String version = in.readLine();
  System.out.println("from client version is: " + version);

  //If the servers version is greater than the client's version
  if(Integer.parseInt(version) < myVersion){
      rightOfWay = true;
      out.println("receive|" + myVersion);

      return "server" + "|" + version;  //This means the server has the right of way
  }else{
      rightOfWay = false;
      out.println("continue|" + myVersion);
      return "client" + "|" + version;   //This means that the client has the right Of way
  }
}
```

Figure 3.14: Code snippet for determining which device has the right of way

3. Begin synchronization by sending files from the device with the right of way to the device on the receiving end, whilst giving feedback to the user at each moment about the progress of the synchronization as shown in figure 3.15. This was accomplished by getting the difference between the versions of the two devices and using a counter to increment the progress upon each cycle as shown in figure 3.16. In order to send the files between the devices a TCP Server and Client were used to send the files in a byte stream after a connection has been established between the devices as shown in the figure 1.1 in the Appendix.

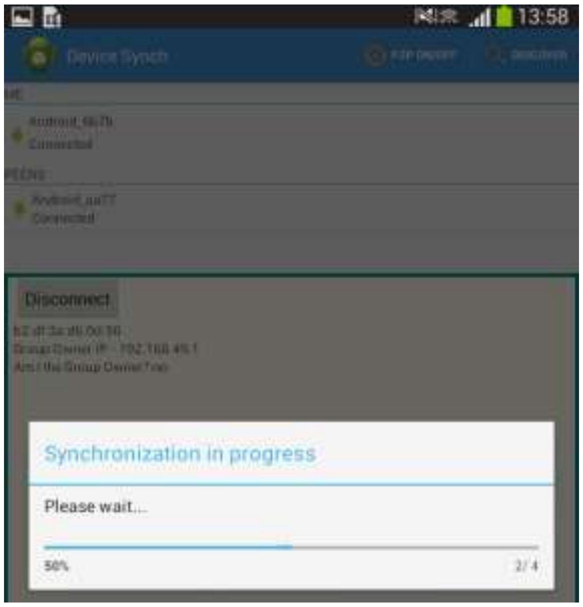
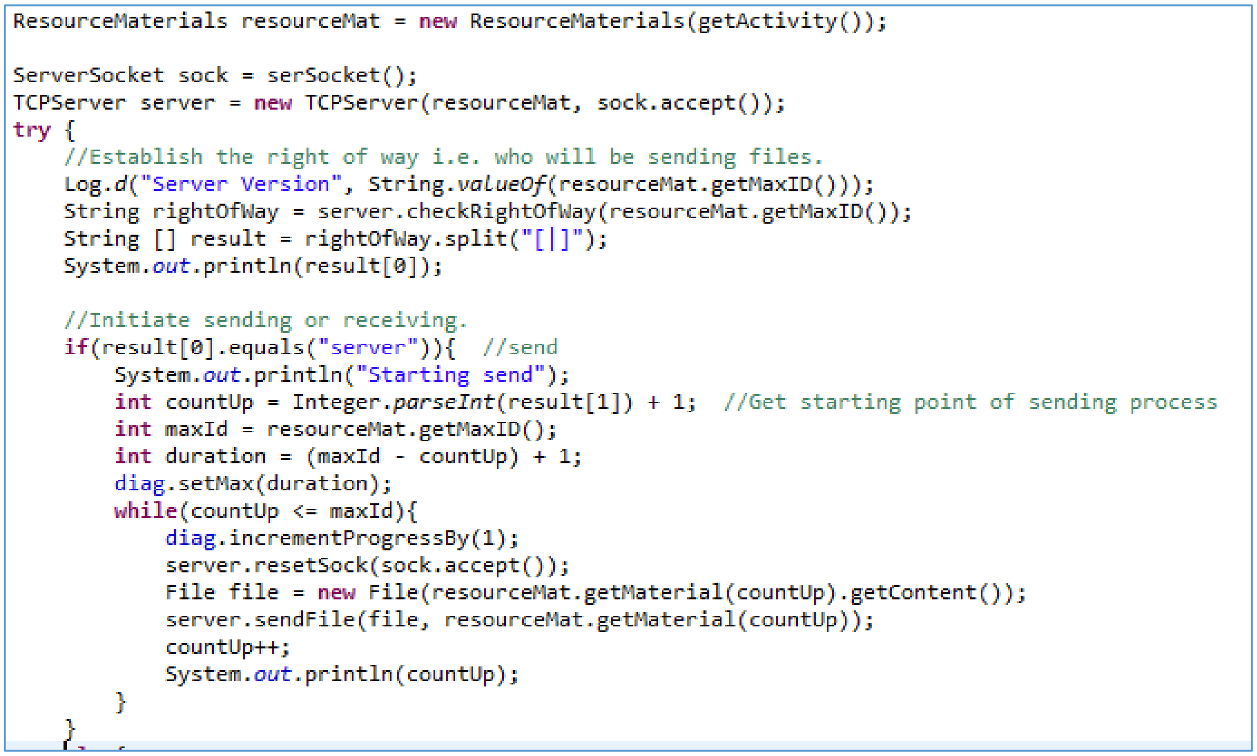


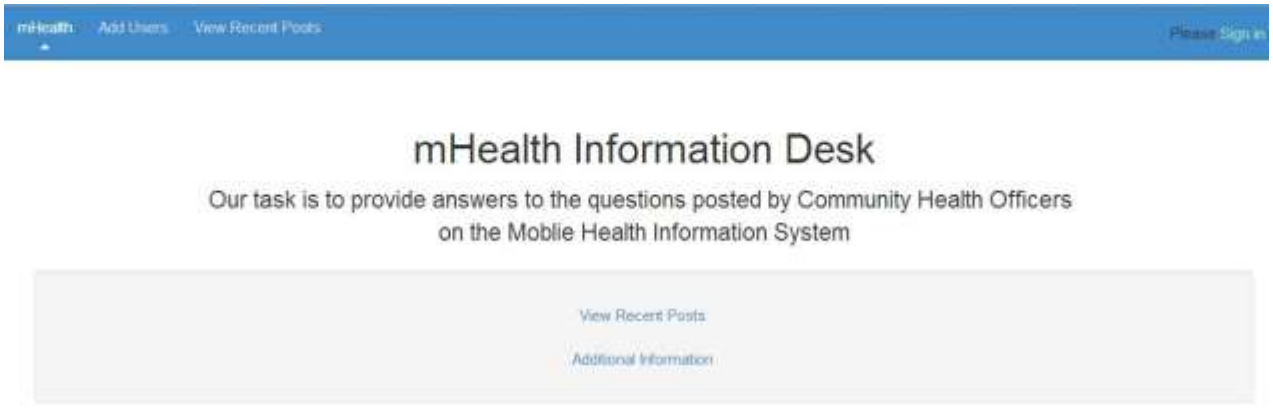
Figure 3.15: Synchronization process



```java
ResourceMaterials resourceMat = new ResourceMaterials(getActivity());

ServerSocket sock = serSocket();
TCPServer server = new TCPServer(resourceMat, sock.accept());
try {
    //Establish the right of way i.e. who will be sending files.
    Log.d("Server Version", String.valueOf(resourceMat.getMaxID()));
    String rightOfWay = server.checkRightOfWay(resourceMat.getMaxID());
    String [] result = rightOfWay.split("[|]");
    System.out.println(result[0]);

    //Initiate sending or receiving.
    if(result[0].equals("server")){  //send
        System.out.println("Starting send");
        int countUp = Integer.parseInt(result[1]) + 1;  //Get starting point of sending process
        int maxId = resourceMat.getMaxID();
        int duration = (maxId - countUp) + 1;
        diag.setMax(duration);
        while(countUp <= maxId){
            diag.incrementProgressBy(1);
            server.resetSock(sock.accept());
            File file = new File(resourceMat.getMaterial(countUp).getContent());
            server.sendFile(file, resourceMat.getMaterial(countUp));
            countUp++;
            System.out.println(countUp);
        }
    }
```

Figure 3.16: Code Snippet for sending of files to peer device

Web Interface for Specialists



mHealth Information Desk

Our task is to provide answers to the questions posted by Community Health Officers on the Moblie Health Information System

Figure 3.17: Home page of the web interface

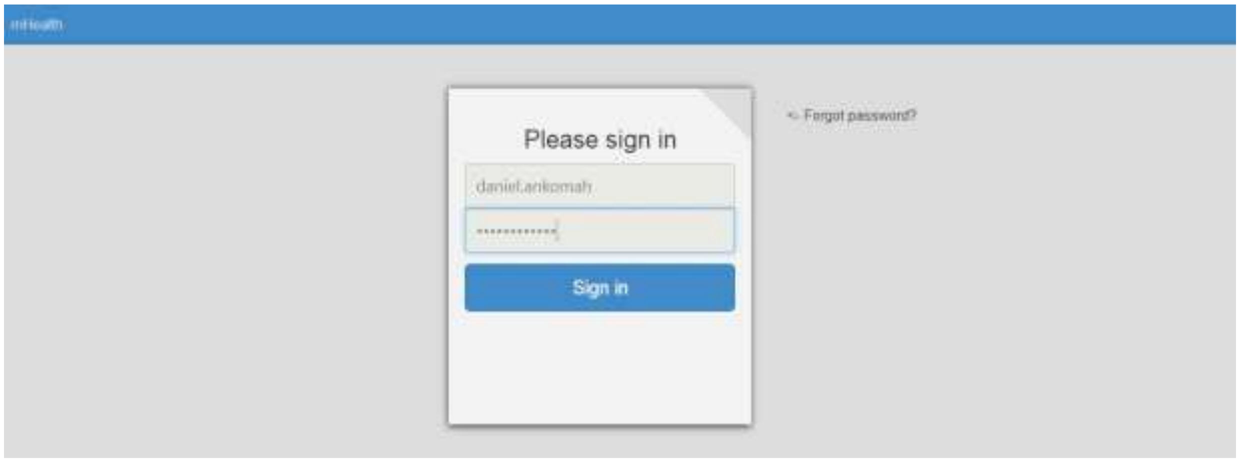
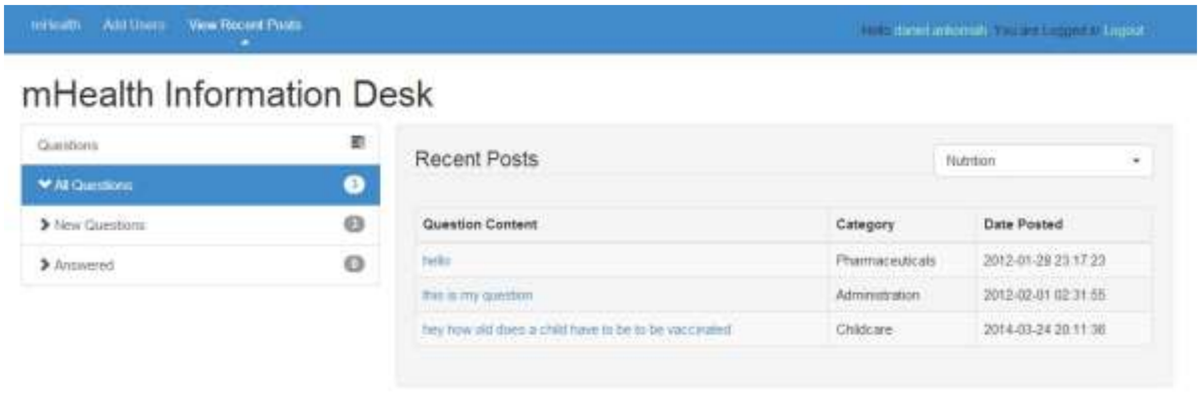


Figure 3.18: Login Screen for Resource Personnel

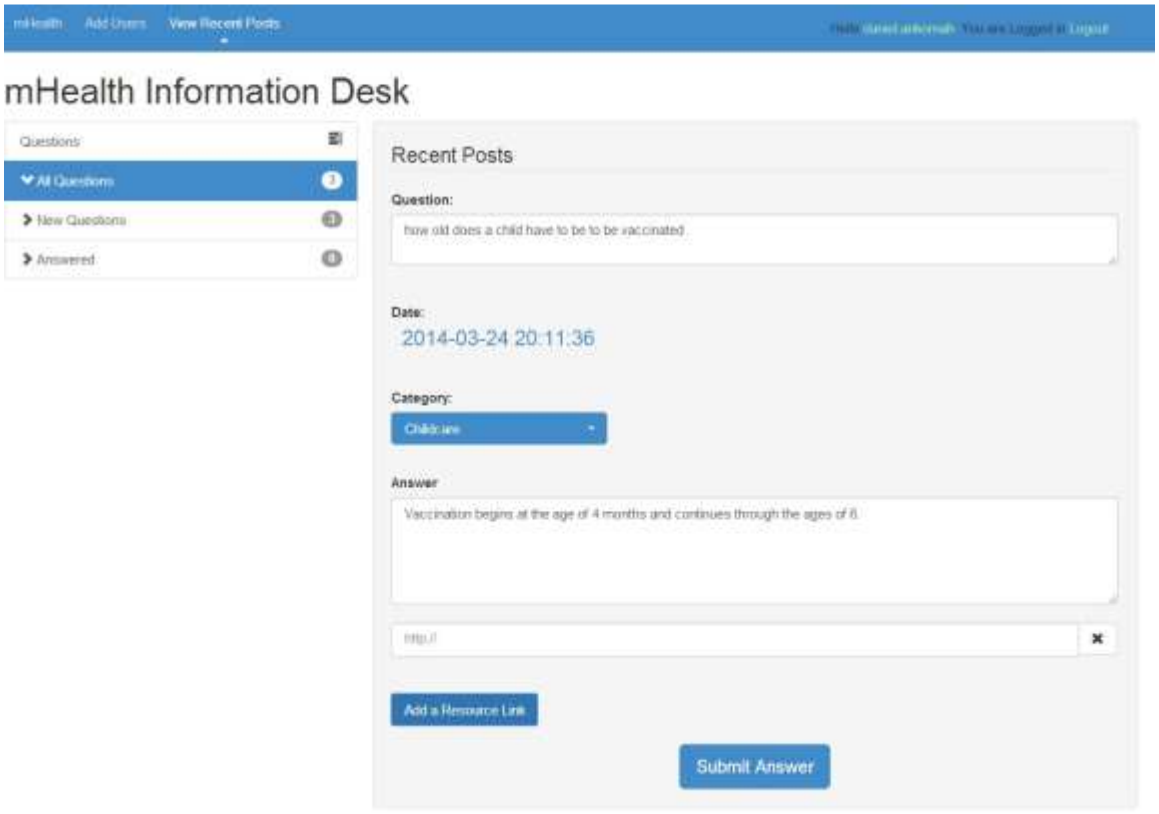


mHealth Information Desk

Recent Posts

Question Content	Category	Date Posted
hello	Pharmaceuticals	2012-01-28 23:17:23
this is my question	Administration	2012-02-01 02:31:55
hey how old does a child have to be to be vaccinated	Childcare	2014-03-24 20:11:36

Figure 3.19: Questions view



mHealth Information Desk

Recent Posts

Question:
how old does a child have to be to be vaccinated

Date:
2014-03-24 20:11:36

Category:
Childcare

Answer
Vaccination begins at the age of 4 months and continues through the ages of 6.

http://

Figure 3.20: Page for answering questions

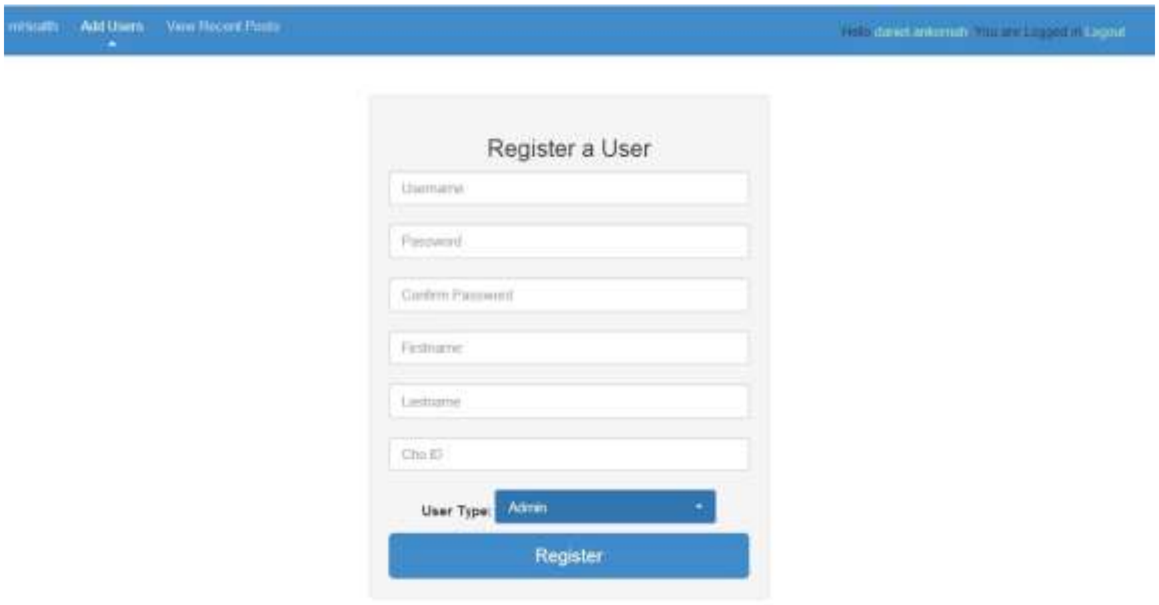


Figure 3.20: Registration page for adding new users

3.1.2 Technology, Tools, and Platform

The technologies used to implement this application are listed below. A detailed explanation of these technologies are included in the appendix

- JSON
- SQLite
- PHP
- MySQL
- Wi-Fi Direct
- Twitter Bootstrap
- CSS
- Android (Eclipse Android Development Tool)
- Windows
- GIT

Chapter 4: Test and Result

Testing is an integral part of every software development because it helps with validation and verification. Validation and verification is the process of determining two main goals namely:

- "Validation: Are we building the right product?" [7]
- "Verification: Are we building the product right? [7]

The application has several interacting components hence the means of testing was primarily component testing. The main focus of this testing was to ensure that components interacting with each other behaved according to the specifications. All the components of the application were also carefully examined to ensure that they adhered to the requirements listed in functional and non-functional section.

4.1 Data entry and retrieval

The core functionality of this application involves the entry and retrieval of data from and to a database both locally (SQLite) and remotely (MySQL). Hence a major part of the testing phase involved this functionality. This involved testing the input of illegal characters and blank fields during periods of internet connectivity and no internet connectivity.

Results.

The web application passed all tests by saving and retrieving saved data from the database at all times. The android application also passed all the tests as expected however, it unearthed one limitation of the application in that it is unable to prevent the entry of some illegal inputs such as bad grammar.

4.2 Interface misuse

Interface misuse testing is undergone to determine if components which call other components are not using the interfaces in the wrong manner [7]. This was particularly relevant during the synchronization of resource materials between devices because the input of the wrong fields into the resource materials table would in term cause the resource materials in the ResourceFragment to either fail or display the wrong data to the user.

Results

After running multiple synchronization sessions as well as navigating through the android application all the tests passed successfully.

Chapter 5: Challenges, Conclusion and Future Work

5.1 Challenges

During the course of development some challenges surfaced. These challenges include but are not limited to issues of data synchronization, and error handling.

5.1.1 Data Synchronization

In order to transfer data between devices a Socket was required to transfer data in byte streams to and from devices. This was particularly challenging because the sockets had to be closed after each file was transferred in order to allow for a new input and output stream to be instantiated from the socket for each new file. Therefore, the Socket had to be reset after each file transfer and this was not as intuitive as it should have been. Also entering and retrieving data into the SQLite database required the creation of multiple Data classes which was time consuming.

5.1.2 Error Handling

The android platform includes a lot of components each separated into different sections. Hence the learning curve is a little steep for users who are not used to the creation of a user interface using XML files. Furthermore, because functions are not connection directly to the components in one file, tracing errors was quite cumbersome and time consuming.

5.3 Future Work

The application has some limitations with transferring resource materials from the PCs to the tablet devices. Currently, the process of transferring resource materials is done manually by inputting the details of the materials in a text file and then copying the materials to a specified directory. When the devices are connected to the PC they are mounted as portable devices. This makes the devices inaccessible programmatically via Java.   Further work can focus on implementing a desktop application to automate the process of transferring new resource materials to the devices via a PC.

5.3 Conclusion

mHealth Knowledge was developed primarily to further work undergone by Florence Jones to provide a mobile health based information system to aid community health workers provide valuable information to the district office. The core functionality of this system is to serve as a platform for the interaction between CHOs and Resource personnel at the district health office to aid CHOs to effectively perform their daily duties. The system includes an android application for CHOs to post questions and receive answers as well as access resource materials provided by the district health office. It also includes a web application for resource personnel to access questions posted by CHOs and provide answers to those questions via internet. Collectively this system will aid CHOs improve their skills as well as provide real time information to the District office where the type of questions posted can be analyzed to determine the areas of operation which require more training.

References

[1] Ministry of Health. (2011, June) National E-Health Strategy. Document. [Online]. http://www.moh-ghana.org/UploadFiles/Publications/Ghana_E-Health120504121543.pdf

[2] World Health Organization. (2011, June) mHealth: New Horizons for health through mobile technologies: second global survey on eHealth. Document. [Online]. www.who.int/goe/publications/goe_mhealth_web.pdf

[3] Ghana Health Service. POTENTIAL OF MOBILE PHONES TO IMPROVE HEALTH IN THE DEVELOPING WORLD. Document. [Online]. http://www.ghanahealthservice.org/includes/upload/publications/Mobile%20health.pdf

[4] Patricia N. Mechael, "The Case for mHealth in Developing Countries," *Innovations*, vol. 4, no. 1, pp. 103-118, April 2009.

[5] Frank K Nyonator, Koku J Awoonor-Williams, James F Phillips, Tanya C Jones, and Robert A. Miller, "The Ghana Community-based Health Planning and Services Initiative for scaling up service delivery innovation," *Health Policy and Planning* , pp. 25-34, 2005.

[6] Florence L. Jones, "Health Records Management System for Community Health Centers," Ashesi University, Berekuso, Undergraduate Dessertation 2013.

[7] Ian Sommerville, "Software Testing," in *Software Engineering*. Boston, United States of America: Addison-Wesley, 2011, ch. 8, pp. 205-228.

[8] ECMA, *The JSON Data Interchange Format*, 1st ed. Geneva, 2013.

[9] Grant Allen and Mike Owens, *The Definitive Guide to SQLite*, 2nd ed. Ney York, United States of America: Apress, 2010.

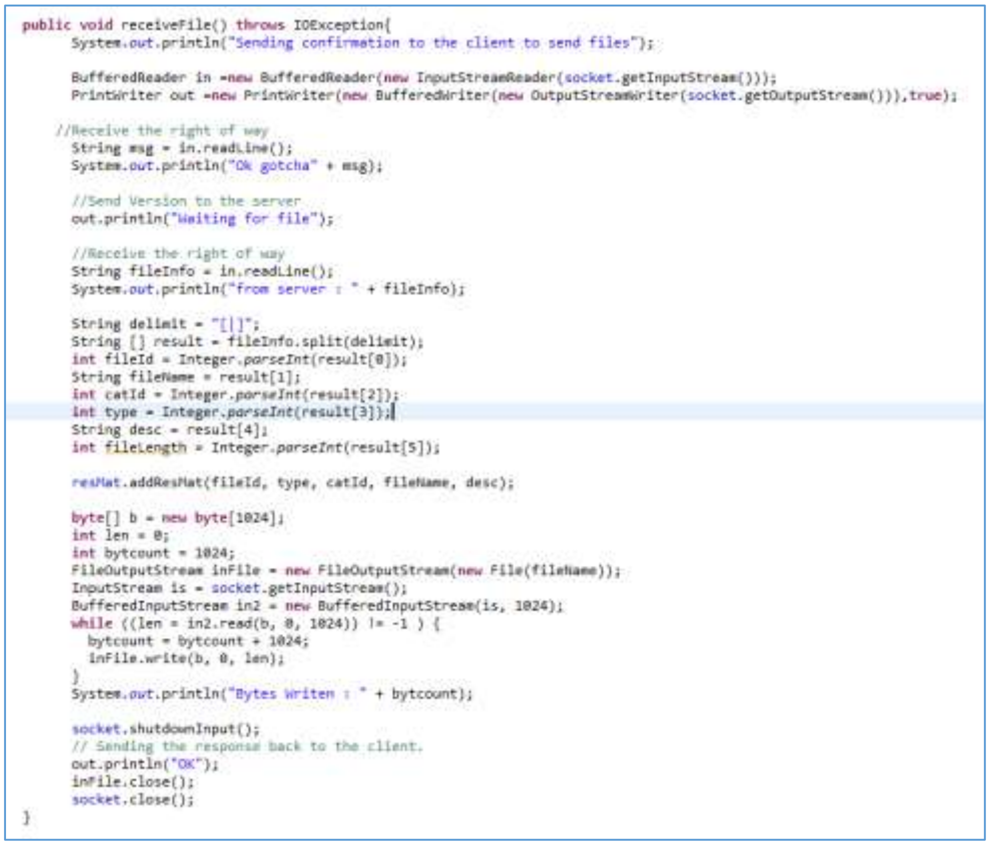
[10] Daniel Camps-Mur, Andres Garcia-Saavedra, and Pablo Serrano. (2010) Device to device communications with WiFi Direct: overview and experimentation. Document. [Online]. http://www.campsmur.cat/files/camps_ssavedra_serrano_WCM_11_00112_final_version.pdf

[11] (2014, April) Android Development Guide. [Online]. http://developer.android.com/samples/index.html

[12] Wei-Meng Lee, *Beginning Android Application Development*. Indianapolis, United States of America: Wiley Publishing, 2011.

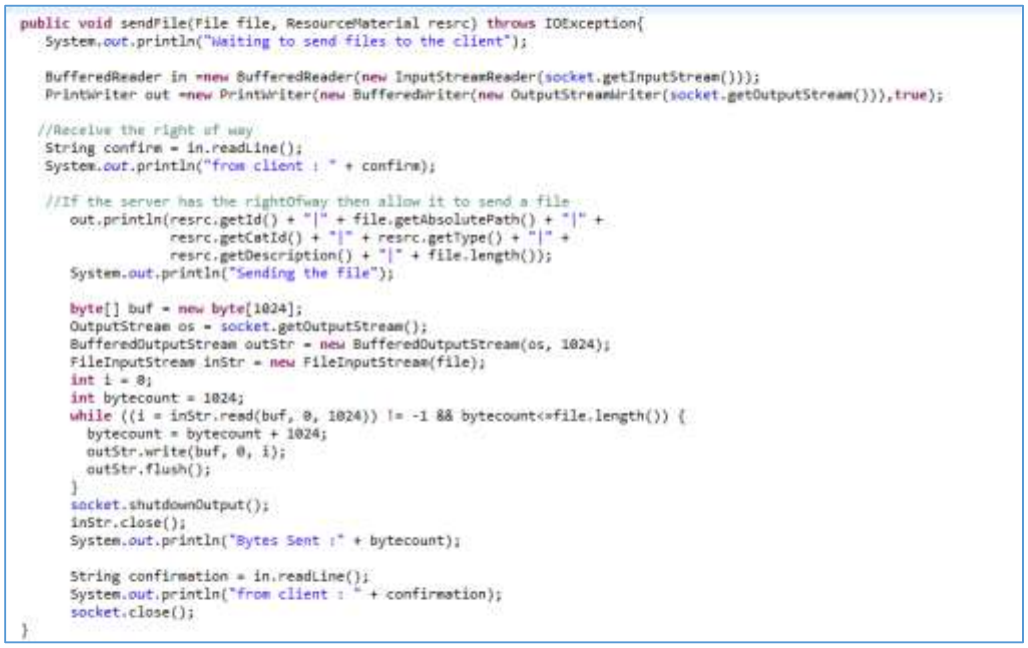
APPENDIX

File Transfer



```java
public void receiveFile() throws IOException{
      System.out.println("Sending confirmation to the client to send files");

      BufferedReader in =new BufferedReader(new InputStreamReader(socket.getInputStream()));
      PrintWriter out =new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())),true);

    //Receive the right of way
      String msg = in.readLine();
      System.out.println("Ok gotcha" + msg);

      //Send Version to the server
      out.println("Waiting for file");

      //Receive the right of way
      String fileInfo = in.readLine();
      System.out.println("from server : " + fileInfo);

      String delimit = "[|]";
      String [] result = fileInfo.split(delimit);
      int fileId = Integer.parseInt(result[0]);
      String fileName = result[1];
      int catId = Integer.parseInt(result[2]);
      int type = Integer.parseInt(result[3]);
      String desc = result[4];
      int fileLength = Integer.parseInt(result[5]);

      resMat.addResMat(fileId, type, catId, fileName, desc);

      byte[] b = new byte[1024];
      int len = 0;
      int bytcount = 1024;
      FileOutputStream inFile = new FileOutputStream(new File(fileName));
      InputStream is = socket.getInputStream();
      BufferedInputStream in2 = new BufferedInputStream(is, 1024);
      while ((len = in2.read(b, 0, 1024)) != -1 ) {
        bytcount = bytcount + 1024;
        inFile.write(b, 0, len);
      }
      System.out.println("Bytes Writen : " + bytcount);

      socket.shutdownInput();
      // Sending the response back to the client.
      out.println("OK");
      inFile.close();
      socket.close();
}
```

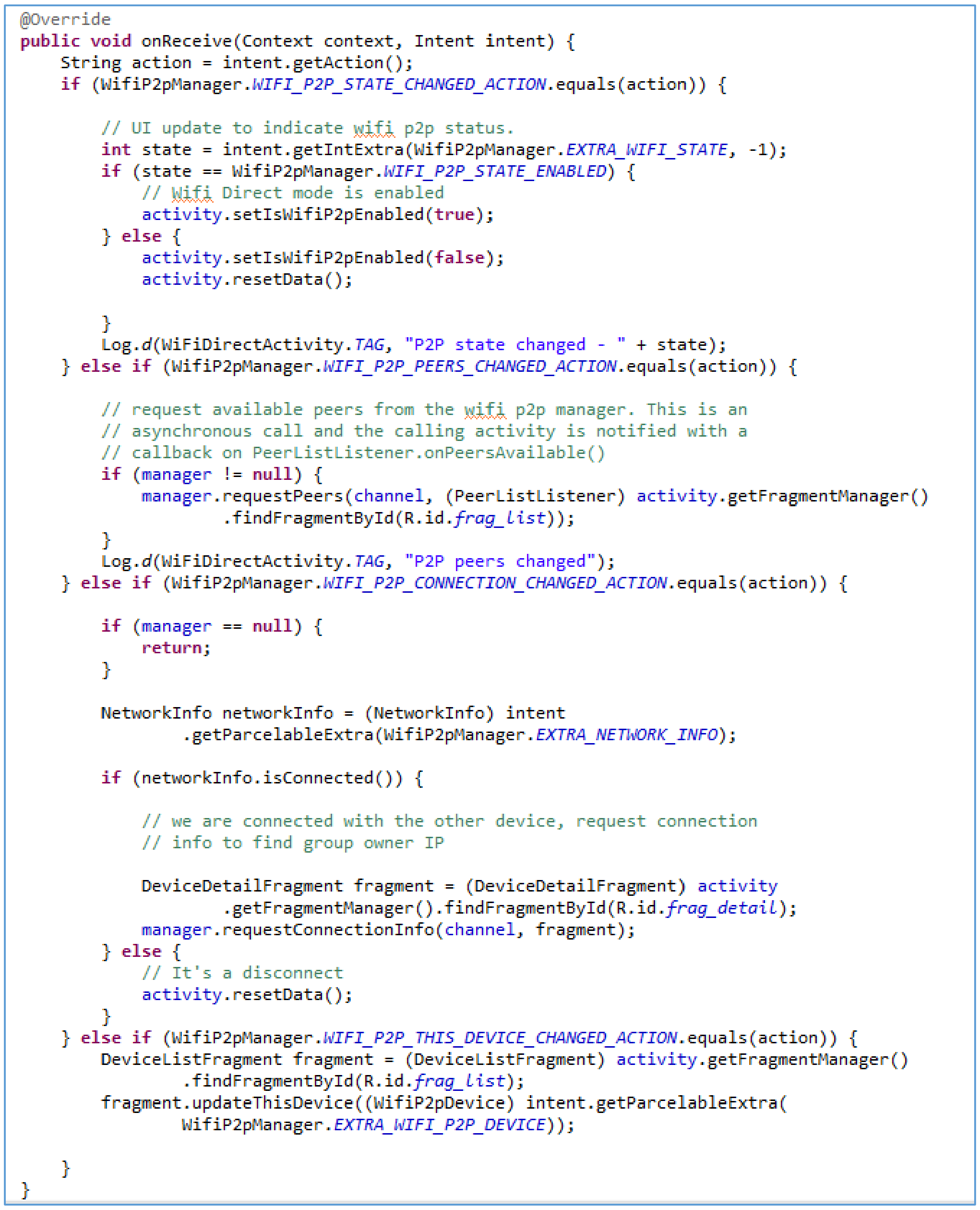
Figure 1: Code Snippet for receiving a file



```java
public void sendFile(File file, ResourceMaterial resrc) throws IOException{
   System.out.println("Waiting to send files to the client");

   BufferedReader in =new BufferedReader(new InputStreamReader(socket.getInputStream()));
   PrintWriter out =new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())),true);

  //Receive the right of way
   String confirm = in.readLine();
   System.out.println("from client : " + confirm);

   //If the server has the rightOfway then allow it to send a file
      out.println(resrc.getId() + "|" + file.getAbsolutePath() + "|" +
                  resrc.getCatId() + "|" + resrc.getType() + "|" +
                  resrc.getDescription() + "|" + file.length());
      System.out.println("Sending the file");

      byte[] buf = new byte[1024];
      OutputStream os = socket.getOutputStream();
      BufferedOutputStream outStr = new BufferedOutputStream(os, 1024);
      FileInputStream inStr = new FileInputStream(file);
      int i = 0;
      int bytecount = 1024;
      while ((i = inStr.read(buf, 0, 1024)) != -1 && bytecount<=file.length()) {
        bytecount = bytecount + 1024;
        outStr.write(buf, 0, i);
        outStr.flush();
      }
      socket.shutdownOutput();
      inStr.close();
      System.out.println("Bytes Sent :" + bytecount);

      String confirmation = in.readLine();
      System.out.println("from client : " + confirmation);
      socket.close();
}
```

Figure 1.1: Code Snippet for sending a file

Wi-Fi Direct Activity



```
@Override
public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {

        // UI update to indicate wifi p2p status.
        int state = intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, -1);
        if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {
            // Wifi Direct mode is enabled
            activity.setIsWifiP2pEnabled(true);
        } else {
            activity.setIsWifiP2pEnabled(false);
            activity.resetData();

        }
        Log.d(WiFiDirectActivity.TAG, "P2P state changed - " + state);
    } else if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {

        // request available peers from the wifi p2p manager. This is an
        // asynchronous call and the calling activity is notified with a
        // callback on PeerListListener.onPeersAvailable()
        if (manager != null) {
            manager.requestPeers(channel, (PeerListListener) activity.getFragmentManager()
                    .findFragmentById(R.id.frag_list));
        }
        Log.d(WiFiDirectActivity.TAG, "P2P peers changed");
    } else if (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {

        if (manager == null) {
            return;
        }

        NetworkInfo networkInfo = (NetworkInfo) intent
                .getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);

        if (networkInfo.isConnected()) {

            // we are connected with the other device, request connection
            // info to find group owner IP

            DeviceDetailFragment fragment = (DeviceDetailFragment) activity
                    .getFragmentManager().findFragmentById(R.id.frag_detail);
            manager.requestConnectionInfo(channel, fragment);
        } else {
            // It's a disconnect
            activity.resetData();
        }
    } else if (WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION.equals(action)) {
        DeviceListFragment fragment = (DeviceListFragment) activity.getFragmentManager()
                .findFragmentById(R.id.frag_list);
        fragment.updateThisDevice((WifiP2pDevice) intent.getParcelableExtra(
                WifiP2pManager.EXTRA_WIFI_P2P_DEVICE));

    }
}
```

Figure 2: Code Snippet showing what happens within the Wi-Fi Direct Broadcast receiver.

JSON

JSON is a text format that facilitates structured data interchange between all programming languages. JSON is syntax of braces, brackets, colons, and commas that is useful in many contexts, profiles, and applications [8].

SQLite

“SQLite is an embedded database. Rather than running independently as a stand-alone process, it symbiotically coexists inside the application it serves—within its process space. Its code is intertwined, or embedded, as part of the program that hosts it. To an outside observer, it would never be apparent that such a program had a relational database management system (RDBMS) on board. The program would just do its job and manage its data somehow, making no fanfare about how it went about doing so. But inside, there is a complete, self-contained database engine at work. [9]”

PHP

PHP stands for Preprocessor HyperText Preprocessor. It is a programming language used for server side scripting for web applications. Its versatility as a rich programming language which can either be deployed on a web server or as a standalone shell makes it very attractive for web development.

MySQL

MySQL is one of the most widely used open-source relational database management systems.

Wi-Fi Direct

“Wi-Fi Direct is a new technology defined by the Wi-Fi Alliance aimed at enhancing direct device to device communications in Wi-Fi. Thus, given the wide base of devices with Wi-Fi capabilities, and the fact that it can be entirely implemented in software over traditional Wi-Fi radios, this technology is expected to have a significant impact” [10].

Twitter Bootstrap

Twitter Bootstrap is a framework developed by Twitter that allows for rapid development of web pages with standard UI elements. It is developed using JavaScript, JQuery and CSS. It customizable and allows for quick and timely delivery of User Interfaces without much implementation.

CSS

CSS stands for cascading style sheet. This is a markup language for styling and developing the user interface of web applications. It was used heavily to style the web application developed for the answering of questions by Resource personnel.

Android (Eclipse Android Development Tool)

The application was developed using the Android Developer Tools (ADT) provided by Eclipse for development in Android. It is open source does not require licensing or purchase. Documentation for the ADT is highly available and accessible on the internet. This makes it a good choice for development purposes.

Windows

The windows operating system is used to host the web server as well as the resource materials for the CHOs.

GIT

GIT is a version control tool used to keep track of source code during development. The Eclipse ADT has a plugin that integrates the GIT functionality into the development environment thereby allowing for easy backup and storage on Google Code.