**Project P5 – Project Design**

Nick Forquer, William Greer, Gerard Puhalla, Trevor Rambacher

**Overview:**

As stated in the project update, the group has implemented an app that gives the user the option to send a file via WiFi-direct or Bluetooth connection. Files from the users download documents can be used for transferring on both the WiFi-direct and Bluetooth connections. The app sets up a connection between two devises utilizing the selected communication format. Once this connection has been established, both devises begin performing the Diffie-Hellman algorithm by creating public keys that they send to one another and private keys that they use to generate the secret shared key. After both sides have calculated the secret shared key, they can send encrypted messages across the connection. In our case, the messages are the file we are transferring. When they receive a message, they know it came from the other party and can easily decrypt it to read the message. Aside from establishing the connection between phones and sending messages between the phones, the two connection types use the same code to perform Diffie-Hellman and encrypt/decrypt the messages.

**Security Approach:**

In order to function properly, the group needed to receive eight different permissions from the users. These permissions include Bluetooth and Bluetooth Admin permissions to allow Bluetooth connections and perform data transfer over Bluetooth sockets. Allow a location permission was required to aid in the finding of devices through Bluetooth discovery. Wifi and Network access as well as Wifi change permissions were required to setup Wifi-Direct to be used as a file transfer medium. Lastly, read and write permissions on external storage were used. This allowed us to access each user’s folders to allow us to transfer a file from one user’s folder to another user’s folder. We limited transfers to the download folder for this application.

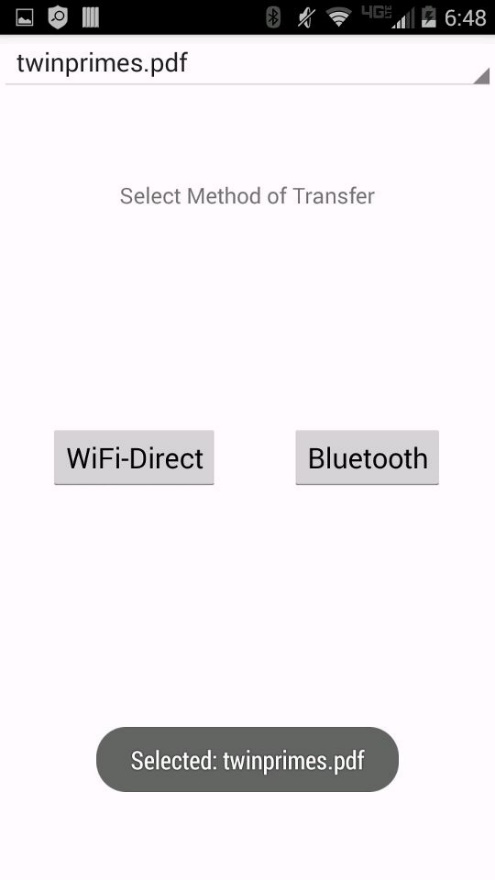
For Diffie Hellman key exchange, we decided to implement this at the application layer. This makes the most sense as we could easily share and reuse the same code for both Wifi-Direct and Bluetooth. It would make little sense to hack into the Bluetooth stack and try to implement there as we would still need some method of securing the messages send across Wifi-Direct. The encryption algorithm that was used was AES as it is the modern standard for symmetric key exchange. Another thing to mention in terms of security is that we set up the receiver as the server. This means that the MAC address of the receiver is exposed as opposed to the sender. Although this does stop all attacks on the person sending the file, it is better than had we done it the other way around where the possibility of sending a file to a complete stranger is much more likely.

**Functionality:**

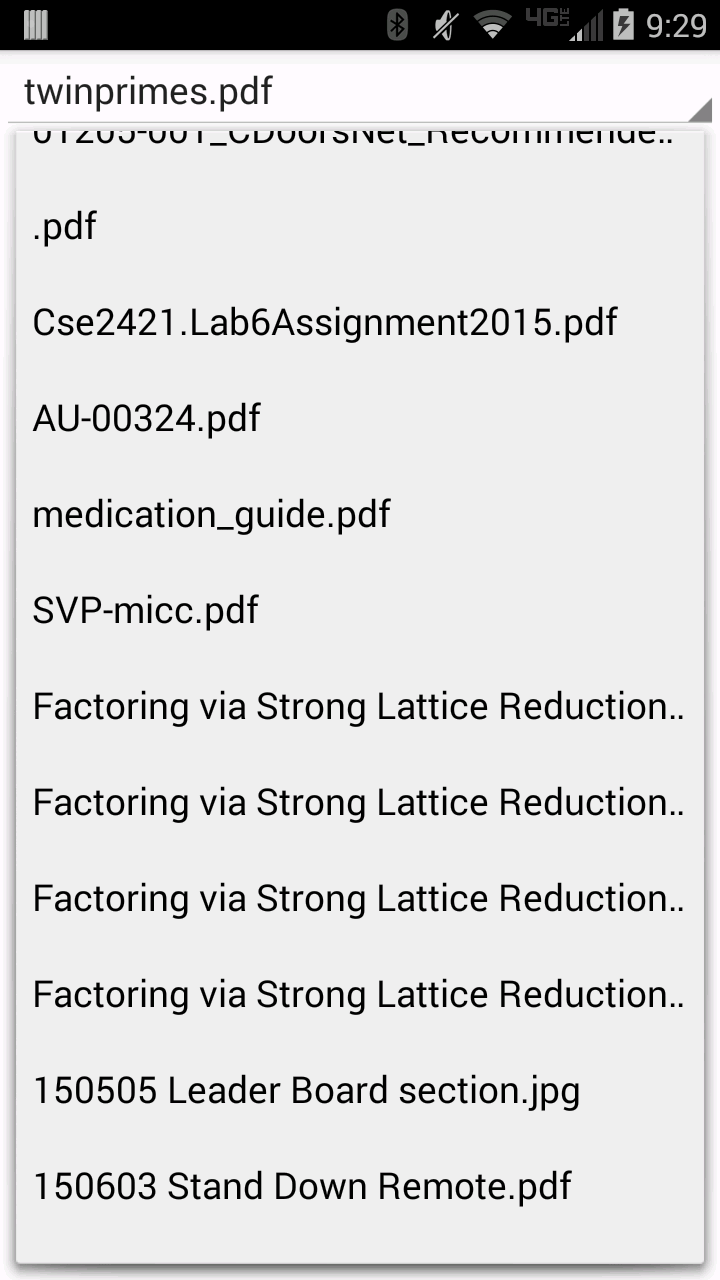
The app was tested after being downloaded onto the phones of two group members. One phone was set up as the receiver and the other phone was set up as the sender. During this initial test, we did find a bug in sender, the sender was not looking for a receiver properly we also had to set up permissions due to one of the phones having above android 6, but the receiver seemed to work properly. We then ran into the problem that the phones would properly generate the shared keys but weren’t receiving the other phones key properly. Another issue the team faced during development was the inconsistent data stream we were receiving from the sender. This corrupt files and made them unusable. Attempts were made to fix this issue and the code logic is now correct, but larger files struggle to make the transfer as the program quits if the connection is lost. The team did not omit anything from project that was mentioned in the original project description and added the challenge of sending files via Wifi-Direct to make the project more complicated for four people to work on.

Screenshots of what our application looks like and a demo have been included below with descriptions, some images may look different due to coming from screenshots of an emulator opposed to from a phone.

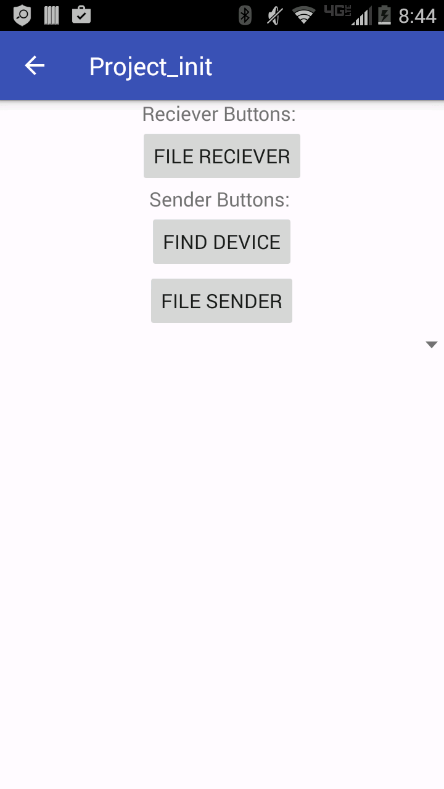
**Bluetooth Screenshots:**



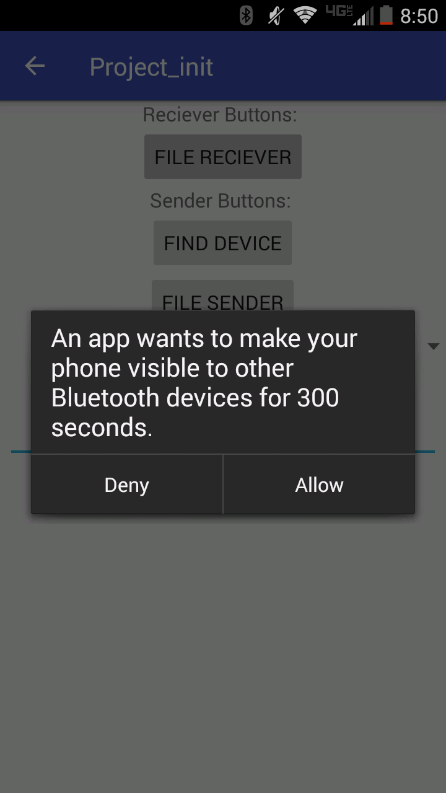
Choice page for Wifi-Direct or Bluetooth connection with the file selected.



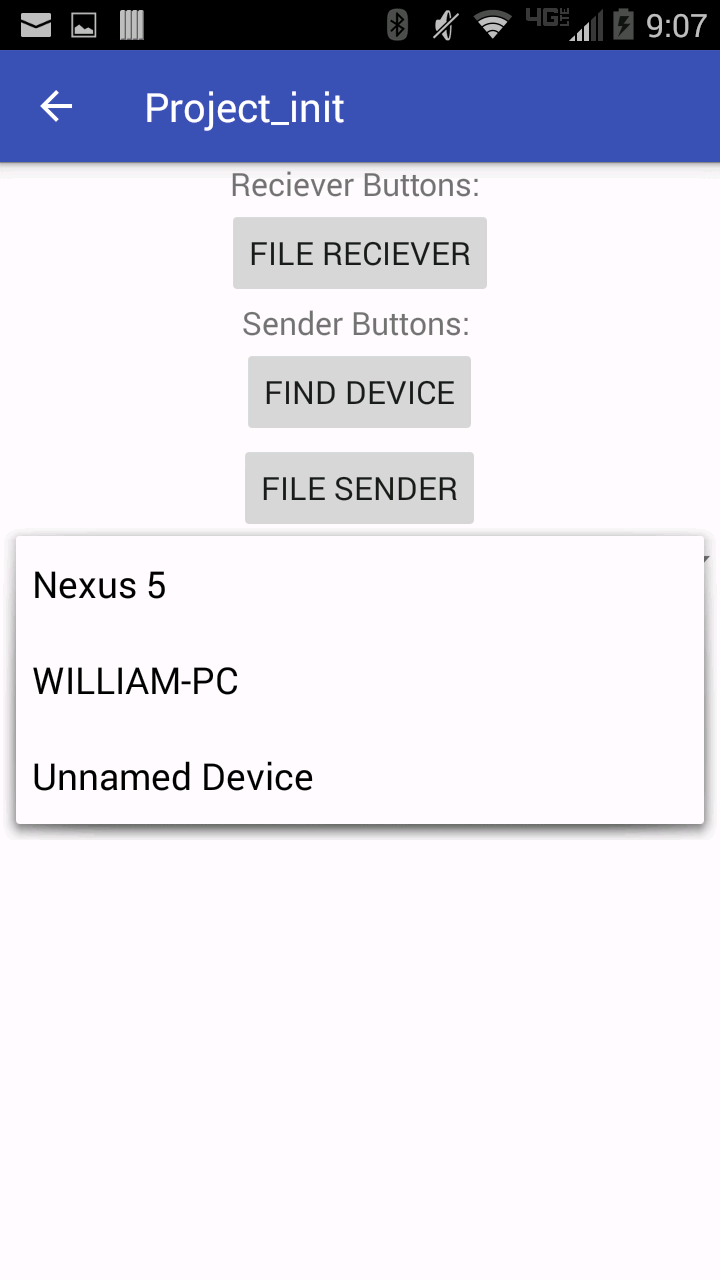
Basic file browser for selecting file to transfer



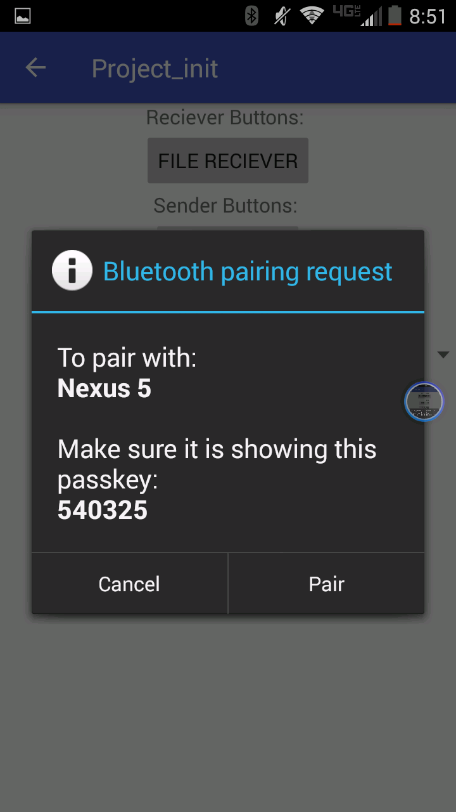
File Bluetooth File Transfer Design



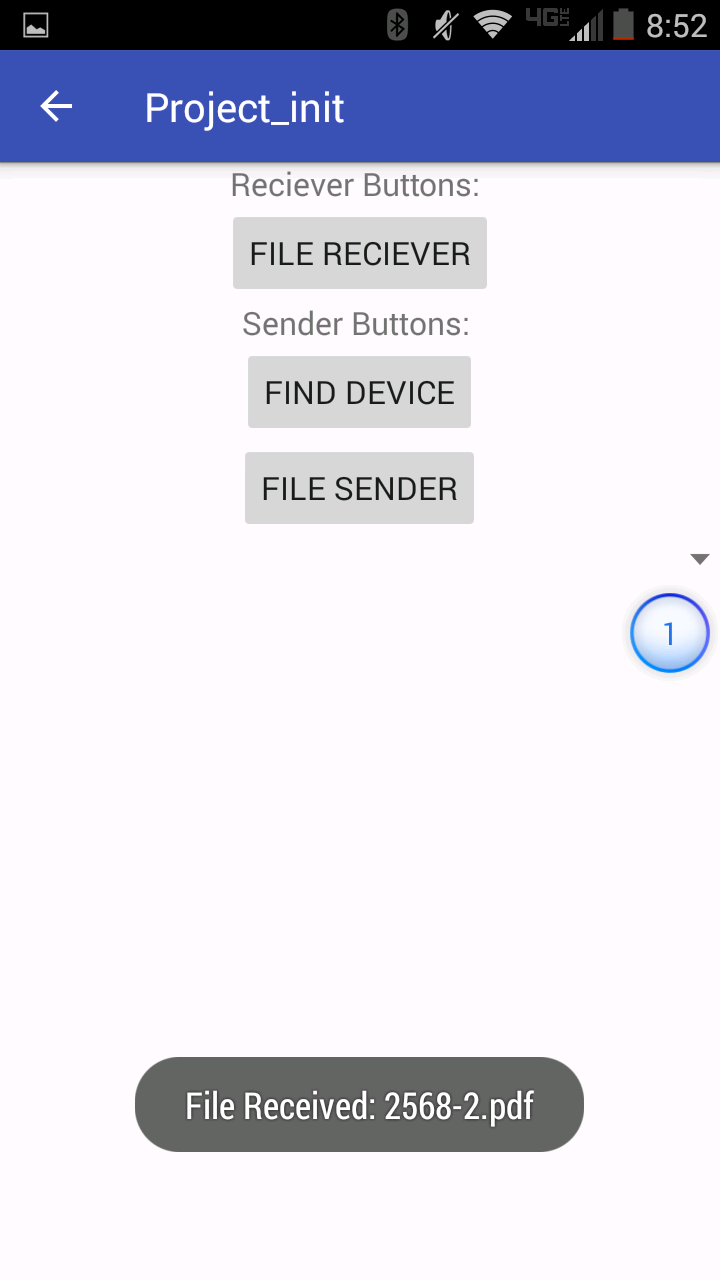
Receiver making the application discoverable for sender.



Sender finding available devices.

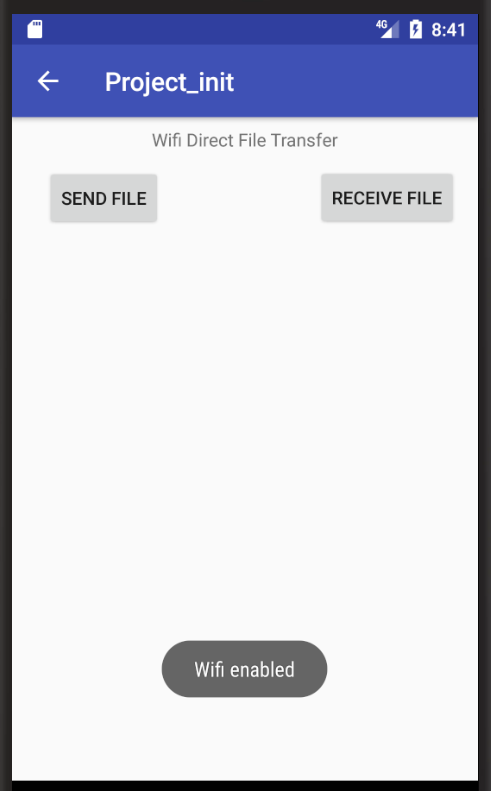


Bluetooth Pairing occurring.

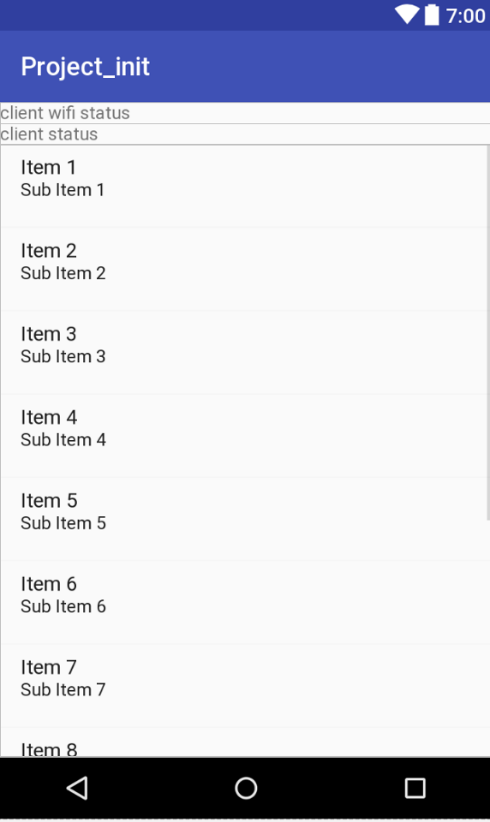


A file successfully transferred over Bluetooth.

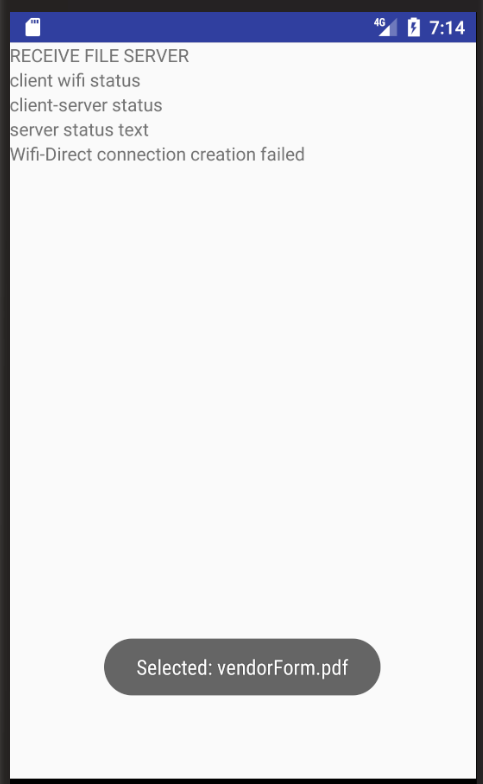
**WIFI Screenshots**



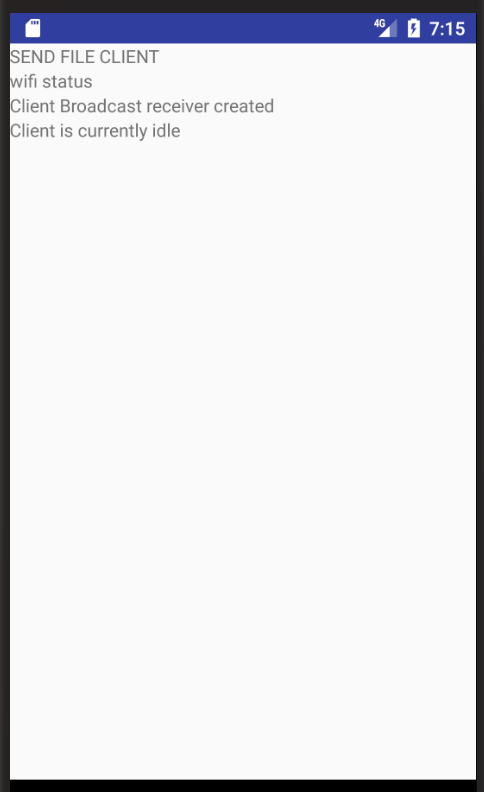
Page indicating whether to send or receive a file. When the user selects Wifi-Direct from the initial screen, the phone’s wifi will be enabled.



Sample screenshot of the list available Wifi-Direct connectable devices



Sample Screenshot of File Receive page. In this image, the connection is marked as failed, as this is a screenshot from the emulator which cannot run the wifi service.



Sample Screenshot of File Sending page