Section 3 in this section we will discuss the architecture and implementation of the projectZ we will also quickly walk through the code amd understand the basics of the implementation so as to make reading the code easier.

[Pause 1 s]

So far we had decided to implement a desktop application and an android one both of them were supposed to contain features as follows

[pause 1s]

The desktop application will contain

1. A command server - that can take input from a socket connection then interpret and emulate the same on the computer.
2. An FTP Server - That can be used to share the files between the devices with or without the app.
3. A GUI for ease of use

[Pause 1s]

While the android application will contain following :

1. Ability to connect to the socket server.
2. Ability to send commands to the server in a sequential order
3. Ability to detect gestures for ease of use
4. And ability to open the FTP Web page directly from the app.

[Pause 1s]

Let’s start then with the desktop application in this we have 3 separate components a command server an FTP server and the GUI to control them

- The command server is built with java. Its code is in the Server.java file in the desktop app folder. It is single class that performs following operations
    - Initiate a socket server
    - Accept connections
    - Accept and wait for a password.
    - If a password is provided activate the connection
    - If the connection is activates parse the command
    - And emulate subsequent behavior

[Pause 1s]

The next part is the FTP server that is actually built with the express server framework available in nodejs. The files for this are the app.js and index.ejs in the views folder of the desktop app. Some utility functions also come from the util.js file.

- The express js is then imported to create a server that responds to get requests made on the “/” end point by rendering the current state of the shared folder
- And “/download/:file” by downloading the file on the user's computer.

[Pause 1s]

The ejs templating engine was chosen as it was very similar to JSP that we learned in this semester. It allows the use of javascript code inside HTML files like jsp does for JAVA.

[Pause 1s]

The GUI is built with electron js. This node js framework allows us to build Desktop applications using HTML CSS and JAVASCRIPT. The code for the GUI is found in the index.js file in the desktop application

folder and the index.html, style.css and script.js files inside the views folder.

[Pause 1 s]

As for the android application we have a single activity android app that can be used to connect to the server send commands and open the open FTP web page

[Pause 1s]
Building this was a much bigger learning experience and significantly more complicated than the Desktop system. The android app has 4 sub packages namely

1. Commands
2. Communicator
3. Connector and
4. Utils

Here we will go through them all one by one

[Pause 1s]

Starting with the commands sub package

It uses 5 java files

1. Command Type is an enum that contains types of commands
2. Command Value is an empty interface so as to build type compatibility between the mouse and keyboard commands
3. Command class uses these high level enums and interfaces to implement a basic structure
4. Whereas the values folder contains 2 more files that implement the command value interface to create key value and move value for mouse and keyboard events respectively.

This structure creates a custom data type for commands that can be used to build commands, convert them to text and send them out.

[Pause 1s]

The connection subpackage is used to implement a custom event handling system inspired by java swing.
This has 3 java files

1. Connection state as an enumeration
2. Connection Listener as an interface for the custom event listener and
3. Connection Manager is a singleton class that create and maintains a single socket connection that can be used to communicate with the command server
4. We can also create Connection event listeners and register them with the Connection manager class that will execute the call backs when events occur

[Pause 1s]

The utils subpackage is a collection of 3 classes that do 3 different things

1. Settings class which is singleton class that is used for storing settings temporarily and using them across multiple classes
2. Command Queue class implements a queue data structure to which commands can be pushed and they are then sent to the server sequentially and then
3. GestureTap.java that creates a custom gesture listener for our project by extending the androids high level gesture detector and creates callbacks that automatically add commands to the queue as events occur.

[Pause 1s]

The final package we will talk about is Communicator which has only one class: the MessageSender. This class essentially runs a separate

thread that continuously watches the commandQueue and dispatches commands if they exist

[Pause 1s]

After this all we have the activity class that uses all above classes to create connections and run commands. It is the easiest part of the code and need not be discussed architecturally

[Pause 1s]

As a side note we have had a claim of using hardware in this project. As it was the last component to be created some of it was delivered just before the covid lockdown and is currently at the hostel. Some orders were cancelled. We have reordered the essential components that will be delivered to me in pune by monday. I will be creating that demo separately as soon as I receive them and I will mail it separately. having already waited too long in the absence of any hard notifications on the submissions we are sending this video in advance.

[Pause 1s]
Handing over for demonstration and conclusion to priyanka more.