Project Documentation

“Explainity”

Team 2

Names :-

1- Ahmed Salah

2- Khaled Sabry

3- Mahmoud Youssri

4- Muhmad Abd Al-Aziz

Idea: -
The app has been built on the idea of the IRC service, that you can chat with group of people just using the relay you send and the server relay to any one with you in the room.
As we hope to call our app the “big S”

Secure:
Provide the security you want to have, Explainity doesn’t store any messages in the room, so you don’t have to worry on your info and chats you send and receive.

Speed:
Faster and speed communication as we don’t make a lot of overhead on the client nor on the server side, which ‘ll make you send and receive on the fly.

Simple:
As you can see from the moment you open the app the simplicity of the design and implementation makes the user interacts with the app easily and faster without too much complicated stuff and also gives you the ability to add more stuff and functionality to the app easily as the project structure built essentially on that.

We call our app Explainity as the app provides fast communication between you and anyone else you want to explain to him or them anything you just need to make a room and a password and send that to the people you want them to join to you.

Features: -
[1] create a room with only your nickname and room name and room password.
[2] join a room with only your nickname and room name and room password.
[3] create more than one room with the same room name but different password.
[4] info on how to use the apps features inside the app.
[5] send and receive messages and know how many clients received this message.
[6] delete your messages from your side and others side.
[7] clear all the history of your room on your side.
[8] live functionality provides to you the ability to show only messages to the connected people in the room right at this moment only.
[9] show count of how many messages you have sent and received.
[10] log out so you don’t receive messages any more from this room.
[11] more about us button to know other apps we provide and connections to us.

Programs we have used: -

Program	Usage
Android Studio	To build the client side (what user uses)
IntelliJ IDEA	To build the server side (the controller
Notepad++	To write the php layer to access the database
Navicat	To build MySQL database
GitHub	To save and store the apps step miles
Word	To write project document

Programming Languages we have used: -

Language	Usage
Java	- in the client-side app to write an android app. - in the server-side app to handle connections and messages.
php	- in layer between the controller in the server side and the database to access (insert, update, select and delete).
MySQL	- to write the database and it's queries.

Markup Languages we have used: -

Language	Usage
xml	- in the client-side app to write the user interface

Tested On: -

Device	Spec.
Lenovo Tab S8	
Lenovo K6	

Project in detail: -

- Android app (client side)
- Java App (server side)
- php layer (accessing the database)
- MySQL database (store connections and rooms)

Android App

As Explainity has been designed on modularity so you can add more functionalities easily so until now these are the modules inside the app.

The whole code is commented and organized in modules and here is the description for the modules: -

Android Modules	Usage
Adapters	Until now it has one class to use it to show the messages in the fragment chatFragment - MessageAdapter : to show the messages in the fragment chatFragment.
Connections	Has the receiver and sender connection and used to receive and send to clients
Controllers	Responsible on the logic in the client app -ChatController : responsible on all the logic in the chatFragment - SignInUpController : responsible on all the logic in CreateRoomFramgent - Controller: have the common functionalities and variables among other controllers
Data	To preserve the temporary data to in the client - Data : save current room name, room id and nickname - Message : same as what in the server side to send it to clients - Server : store the ip and ports of the server
Fragments	The view side - AboutMeFragment : identifies ourselfs and apps and connections to us -ChatFragment : to call the queries you want - CreateRoomFragment : the create and join layout view - InfoFragment : show to you how to use the program
Functions	Include all the generic functions - Toasts : to show you beautiful success/error message - UpdateConnection : to periodically send to the server the alive message
Processing	To decode and encode the messages before sending it to server - MsgDecoder : responsible for all the decoding of the messages requests and responses from the server - MsgEncoder : to encode all the messages and add the functionality of the message before sending to the server
MainActivity	To load all the fragments from it and open the port of the client to listen to
Processing	To process the requests - ConnectedIps : to delete if a client didn't connect to the server from a while - MsgDecoder : to decode all the messages from clients - MsgEncoder : to encode all the messages to the clients

Android XML	Usage
activity_main	To open on it all the fragments later
fragment_chat	The view to see the room chat
view_chat_header	The header of the chat view
view_chat_input	The text you put into the message you send
view_chat_message	The cardview of the message
view_chats	The recyclerview that you see the message in
fragment_create_room	The first view appears to you when you open the app
fragment_info	How to use explainity

- The client listens on the port 5000.

Java App
As Explainity has been designed on modularity so you can add more functionalities easily so until now these are the modules inside the app.

The whole code is commented and organized in modules and here is the description for the modules: -

Java Module	Usage
Connections	Has the receiver and sender connection and used to receive and send to clients
Data	To preserve the temporary data to the server - Client : save current connections and functions to deal with it - Message : same as what in the client side to send it to clients
Database	To access the php files so you can insert, update, select and delete - DatabaseConnection : to connect to php files -DatabaseQuery : to call the queries you want - DatabaseResultDecoder : layer only to make a check and gets the result if there was an error happened
Main	Has the main class - Main : to run the thread and open ports to listen to
Processing	To process the requests - ConnectedIps : to delete if a client didn't connect to the server from a while - MsgDecoder : to decode all the messages from clients - MsgEncoder : to encode all the messages to the clients

- The server listens on the 10000, 10002, 10004 ports.

php layer: -
We have 8 important php files: -

Php file	Usage
connection	This file is imported in all other php files as this ;'' make a connection with the MySQL server to connect to database
insert_room	To create new room - params : room name and password - returns : true (created) – false (not created)
insert_connection	To join already existing room -params : room_id, nick_name, ip, last_time_entered -returns : true (joined) – false (not joined)
select_room_name	To get room id - params : room_name, room_password - returns : room_id
select_all_connections_in_room	Gets an array of all ips Inside the room - params : room id - returns : array of ips
delete_Connection	To delete connection if he didn't connect to the server in a period of time or he logged out from channel - params : ip - returns : true (deleted) – false (not deleted)
update_connection_time	@Deprecated : was to update connection time - params : room_id, nick_name, ip, last_time_entered - returns : true (updated) – false (not updated)
select_connection_nick_name	@Deprecated: was to check if the nick name inside this room or not - params : room_id, nick_name

- All these files connect to MySQL database and returns the result to the java app (server side)

MySQL Database: -
- you can make rooms with the same name but different passwords.
- you need to make only one nickname inside the room id you can't make two with the same name.

Database Table	Columns
rooms	id, room_name, password
connections	room_id, nick_name, ip, last_time_entered

Images from Tests: -