User guide for running the *Secure Remote Control* application

NOTE: For the instructions on running your local database, please see the accompanying document SRC_DB_Manual.pdf. We recommend reading the whole document once before running the application.

For running the application locally, firstly start web admin with instructions under segment Group 2. After that, start Communication layer app with steps under Group 3. After that is done, you can start Android app with instructions under Group 1.
Keep in mind that if you want to start locally, both web admin and communication layer need to use same database source (local or remote).
Also, when running projects locally, Web admin and communication layer support localhost resolving, but android app requires full ip address when connecting to communication layer. You can find local ip address of the machine where you start web admin and communication layer by using ifconfig/ipconifg command in your terminal.

Group 1 - Android

1. Install Android Studio Flamingo

Go to the official Android Studio website (https://developer.android.com/studio) and download the version called "Flamingo".
Follow the installation steps for your operating system (Windows, macOS, or Linux).

2. Clone and Pull the Code from the Repository

Go to the following link:
https://github.com/SI-SecureRemoteControl
Open Client-Side-Android-app
Open Android Studio
Click on "Get from VCS" (Version Control System)
Paste the repository link and click "Clone"
After cloning, right-click on the project folder in the Project view and choose "Git → Pull" to make sure you have the latest code

3. Sync the Gradle

Gradle is the tool that builds your project.

When the project opens, a bar will appear at the top saying "Sync Now" – click it
If not, go to the menu and click "File → Sync Project with Gradle Files"
Wait for it to finish downloading everything it needs

4. Connect Your Phone or Build an APK

You can either connect your Android phone with a USB cable (make sure Developer Mode and USB Debugging are turned on), or
Go to "Build → Build Bundle(s) / APK(s) → Build APK(s)" to create a file you can install manually on your phone.

5. Run the Application

If your phone is connected, click the green "Play" button in Android Studio.
If you created an APK, install it on your phone and open the app manually.

6. Enter Server Address in the App

When the app opens, you'll see a field asking for a server address
Type:

ws://your_computer_ip_address:8080

For example, if your computer's IP address is **192.168.1.5**, you would enter: **ws://192.168.1.5:8080**

Group 2 - Web app

1. Clone the Repository and Pull Latest Changes

Go to the following link:
https://github.com/SI-SecureRemoteControl/web-app

Open a terminal (or Git Bash), navigate to a folder where you want the project to be and run the following commands:

```
git clone https://github.com/SI-SecureRemoteControl/web-app.git
cd [project-folder]
git pull
```

2. Install Node.js

Download Node.js from https://nodejs.org and install it if you don't have it
Make sure it includes npm (Node Package Manager)

3. Add the .env Files

There are **two** `.env` files to create:

- A backend `.env` file (inside the `backend` folder)
- A frontend-specific `.env` file (inside the `frontend` folder)

Do the following:

For the **backend** .env, create a new .env file with no name, inside project (web-app folder).

In the file, paste the following:

```
PORT=9000
DB_URI='mongodb+srv://root:root@cluster.qciyr2x.mongodb.net/Cluster?re
tryWrites=true&w=majo rity&appName=Cluster'
DB_URI_LOCAL='mongodb://localhost:27017/'
USE_LOCAL_DB=true
SECRET_KEY='10429b28ab5cb5042e393b2f20d76bb1'

Secret key can be anything you want, this is just an example.
```

Do not be confused with two database URIs, the local URI is used by developers when testing and adding new features. If you want to use local database you can run docker compose which will pull Mongo image and start up a Mongo instance in your local docker container.

Brief explanations for each field in the .env file for backend:

1. PORT: Specifies the port on which the server should listen.
2. DB_URI: This is the connection string for a cloud-hosted database, such as MongoDB.
3. DB_URI_LOCAL: Connection string for a **locally hosted NoSQL database instance** (e.g. MongoDB), such as one running in Docker or directly installed.
4. USE_LOCAL_DB: A flag that determines **which database URI** to use — the **local** one or the **cloud** one.
5. SECRET_KEY: Used to sign and verify JWT tokens for authentication

For the **frontend** .env, create a new .env file with no name, inside frontend folder of the repository

In the file, paste the following:

```
VITE_BASE_URL=http://localhost:9000
VITE_WS_URL=ws://localhost:9000
VITE_API_UPLOAD_URL=http://localhost:5000/api/upload
```

Brief explanations for each field in .env:

1. **VITE_BASE_URL**: base URL for the backend API or server during local development
2. **VITE_WS_URL**: WebSocket URL of backend for real-time communication.
3. **VITE_API_UPLOAD_URL**: This is the full URL to the upload API endpoint, which points to local instance of communication layer.

6. Start the Backend

Open a terminal or command prompt
Navigate to the backend folder (usually the root folder of the project)

Run the following commands:

```
npm install
npm start
```

7. Start the Frontend

Open a new terminal or command prompt
Navigate to the `frontend` folder inside the project

Run the following commands:

```
npm install
npm run dev
```

8. Access the App

Once both backend and frontend are running, open your browser and go to the address shown in the terminal

Group 3 - Communication Layer

The Secure Remote Control Gateway is a Node.js-based server that acts as a communication layer between IT admins and Android devices. It facilitates secure WebSocket-based real-time interactions, including device registration, WebRTC signaling, and remote control commands.

1. ***Ensure you have node.js installed on your machine:***

```
node -v
npm -v
```

2. ***Clone the repository:***

```
git clone
https://github.com/SI-SecureRemoteControl/secure-remote-control-gateway.git

cd secure-remote-control-gateway
git pull
```

3. ***Install dependencies:***

```
npm install
```

4. ***Create a .env file and add the following fields:***

```
DB_URI_LOCAL=mongodb://localhost:27017/
DB_URI=mongodb+srv://root:root@cluster.qciyr2x.mongodb.net/Cluster?retryWrites=true&w=majority&appName=Cluster
JWT_SECRET=23181ac6f726f8d199b4c6732de22cc8b1869102118b74eab4dd8fbb7d18fc492fee2016f3f483ce369a3c0bfa9228daa8d0926865d8505d2607de6253930596
PORT=5000
SERVICE_URL=http://localhost:5000
USE_LOCAL_DB=true
WEBSOCKET_URL=ws://localhost:9000/ws/control/comm
```

Field explanations:

1. **DB_URI_LOCAL**: A database connection string to an instance of a NoSQL database (such as MongoDB) for local development
2. **DB_URI**: Database connection string for production or remote use (such as MongoDB)
3. **JWT_SECRET**: Secret key used for signing and verifying JWT (JSON Web Tokens). This is used for authentication and should be kept secure.
4. **PORT**: Port on which the backend Node.js server (API) will run.
5. **SERVICE_URL**=Base URL for the backend server. The frontend will use this to send API requests. In local development, this points to the backend running on port 5000.
6. **USE_LOCAL_DB**: Boolean flag that tells your application whether to use the local database URI or the remote one. If true, DB_URI_LOCAL is used. If false, DB_URI is used.
7. **WEBSOCKET_URL**: WebSocket endpoint for real-time communication between admin and Android devices. This is used by the communication layer to connect to the web admin backend. In local development, the web admin backend server runs on port 8080.

5. Start the server:

```
npm run start
```

Running this will trigger database migrations in migrate.js file in Communication layer project. Those migrations are not necessary in project, rather they are used for cleaner database handling and changes history in Database scheme.