



UNIVERSITÉ
DE LORRAINE



Nancy-Brabo
Réseaux et Télécommunications

BUT R&T - Develop communicating applications

Report SAE 3.02



Students: GENAY Loïs
HALABI Kinan
Groups : ROM 2

Teacher : Mr. Beysson Clement

Table of contents:

Introduction:	3
The Login page:	4
Code Implementation:	4
Principal page:	5
The Notepad page	6
How it works:	6
Code Implementation:	6
To save a note.....	6
TO RETRIEVE THE SAVED NOTE WHEN REOPENING THE APP	6
The speed test page:.....	7
Code implementations:	7
TO START THE DOWNLOAD:.....	7
TO CALCULATE THE DOWNLOAD SPEED:	7
The WIFI specs page:	8
The IP Address Calculator Page:.....	9
Code Implementation:	9
TO COMPUTE THE SUBNET MASK AND NETWORK ADDRESS	9
TO FORMAT THE IP ADDRESS	9
Multilingual Support:	10
Summary:	10

Saé302

Introduction:

During this SAE, our goal was to create an application of our choice using Android Studio in Java. We had restricted specifications, with three main conditions to respect:

- Integrate at least three different views, with two of them connected using intents (three pages).
- Use constraint layouts.
- Support at least two different languages (English and French).

To complete this project, we chose to develop an application with five distinct functionalities:

- 1) A **login** and **creation account** system
- 2) A **notepad** – To save and manage quick notes.
- 3) A **speed test** – To measure the internet connection speed.
- 4) A **Wifi checker** to see stats of a wireless network.
- 5) An **IP address calculator** – To help network engineers quickly compute network information.

Below is a detailed breakdown of the different pages in our application, with screenshots and code explanations.

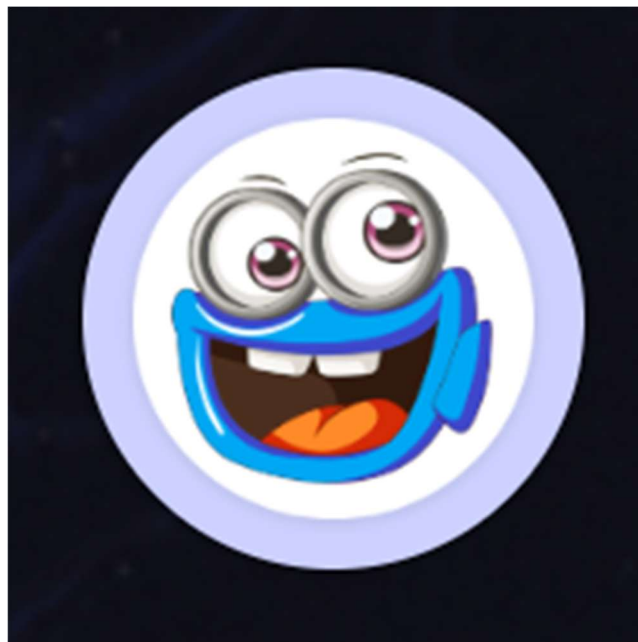


Figure 1 Logo of the application

The Login page:

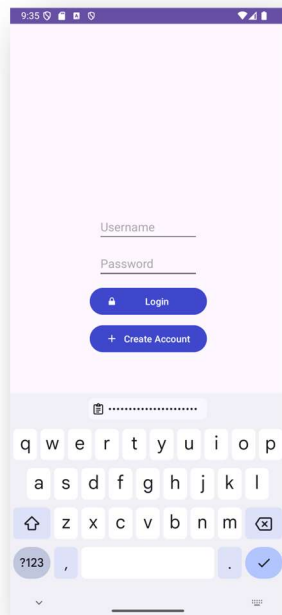


Figure 2 Login page

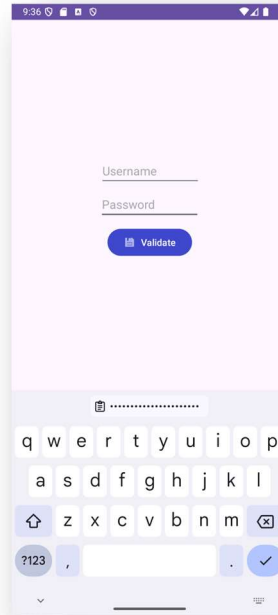


Figure 3 Create an account page

If you open the application for the first time, you must create an account before logging in. Once your account is created, it is stored in the phone's cache, allowing you to log in anytime.

We chose to use **SharedPreferences** for storing login data because it is easier to implement compared to an **ArrayList** or an **SQLite** database. This approach allows us to save the username and password locally in a simple key-value format but only one account can be saved at the same time with this method.

Code Implementation:

Here is the main logic for storing and retrieving user credentials:

SAVING LOGIN CREDENTIAL

```
editor.putString(s: "username", username);  
editor.putString(s: "password", password);  
editor.apply();
```

RETRIEVING LOGIN CREDENTIAL

```
SharedPreferences prefs = getSharedPreferences(name: "MyAppPrefs", MODE_PRIVATE);  
String savedUsername = prefs.getString(s: "username", s1: null);  
String savedPassword = prefs.getString(s: "password", s1: null);
```

Principal page:

Once you log in, you can choose from four features:

Notepad, Speed Test, My Network, or Network Calculator.

- **Notepad** allows you to take notes quickly and easily. You can delete any note by holding it and selecting the delete option. Your notes are always accessible.
- **Speed Test** measures the speed of your Wi-Fi connection, helping you determine how fast your internet is.
- **My Network** analyzes your proximity to your network box, indicating whether you have a good signal or if you should move closer for a better Wi-Fi connection.
- **Network Calculator** helps you calculate network details. Simply enter an IP address with its subnet mask, and the app will provide key information such as the subnet mask, default gateway, and the range of usable addresses.

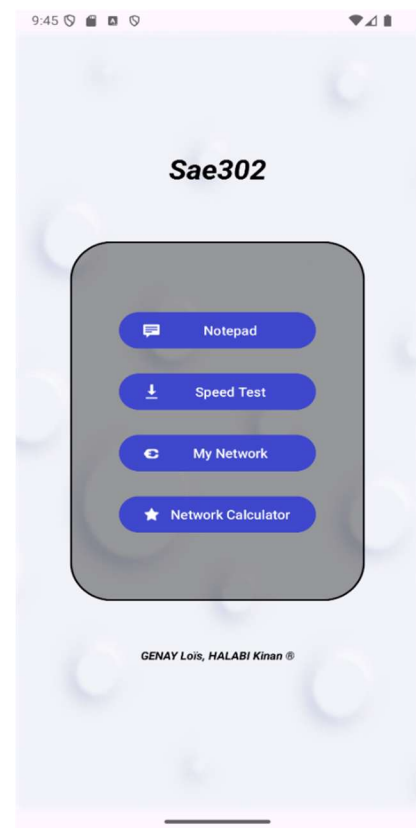


Figure 4: Menu page

We chose to put a video background and that was rude because we needed to change de video codec to **H264** and down the resolution to **720p** for better performance and adding a **onResume** when we change the view.

```
VideoView videoView = findViewById(R.id.background_video);
Uri videoUri = Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.bubbles_video);
videoView.setVideoURI(videoUri);
videoView.setOnPreparedListener(mp -> {
    mp.setLooping(true); // Boucler la vidéo
});
videoView.start();
```

```
protected void onResume() {
    super.onResume();
    VideoView videoView = findViewById(R.id.background_video);
    if (!videoView.isPlaying()) {
        videoView.start();
    }
}
```

Also, we use the **onClick** method to change view with **intents**.

```
public void openNotepad(View view) { 1usage
    Intent intent = new Intent(packageContext: MainActivity.this, NotepadActivity.class);
    startActivity(intent);
}
```

The Notepad page:

The notepad feature allows users to create, edit and delete notes. Notes are saved locally (in the cache) so that they persist even after closing the app. There is an add note button (+) to add, if we short press the note we can edit it and if we long press on it we can delete it. We used **SharedPreferences** to store notes because it provides a simple way to save key-value data. However, for a more scalable approach we could have used a local SQLite database.

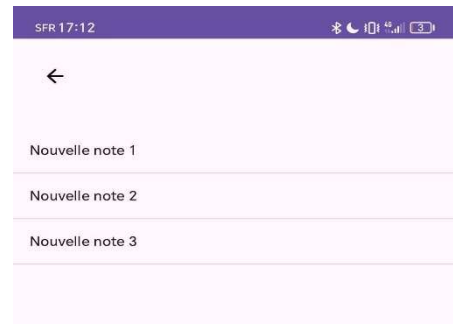


Figure 4: Notepad page

How it works:

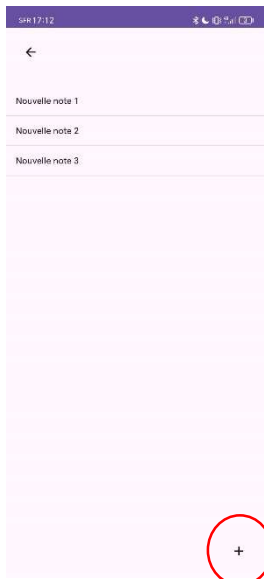


Figure 5: Add note

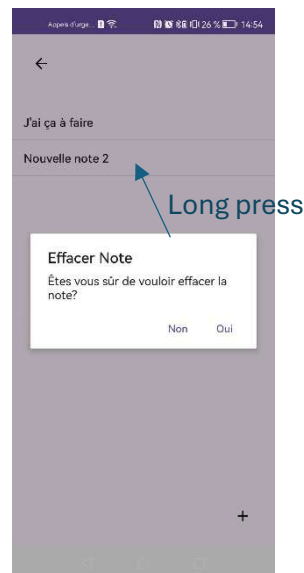


Figure 6: Delete note



Figure 7: Edit note

Code Implementation:

To save a note

```
sharedPreferences = getSharedPreferences( name: "NotepadPrefs", Context.MODE_PRIVATE);
notesList = new ArrayList<>(sharedPreferences.getStringSet( s: "notes", new HashSet<>()));
Set<String> notesSet = new HashSet<>(notesList);
sharedPreferences.edit().putStringSet( s: "notes", notesSet).apply();
```

TO RETRIEVE THE SAVED NOTE WHEN REOPENING THE APP

```
ListView listView = findViewById(R.id.list_notes);
adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, notesList);
listView.setAdapter(adapter);
```

The speed test page:

The speed test page was a great idea, but unfortunately, we didn't get accurate results. We initially started with a simple code snippet from StackOverflow, but it didn't work correctly.

To improve it, we decided to run the **ping method** and the **download method** in two separate threads, but this didn't make any difference. The results were still inconsistent and far from reality.

We finally managed to get accurate measurements after multiple attempts. We realized that the **ping method** was not effective, so we removed it from the app. Instead, we kept **HTTP requests**, which provided more reliable and realistic results.

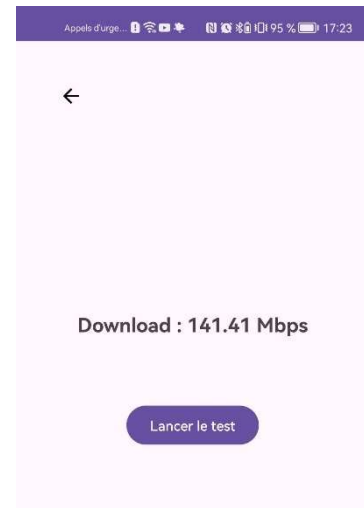


Figure 8: Speed test

The speed test is a tool designed to **measure** the performance of an internet **connection** by analyzing key factors. By sending and receiving data packets, it evaluates the **bandwidth** and overall **network reliability**. This test helps users determine if their **internet speed** matches their provider's promises and if their **connection** is stable for activities like streaming, gaming, or video calls.

The application downloads a set of data from a remote server and calculates how fast the data is received, measured in **megabits per second (Mbps)**. The test often selects the **nearest server** to ensure the most accurate results, reducing interference from other network factors. The results help users understand their **internet performance** and whether their connection meets their needs. In our case, the nearest server is in Metz, which is quite near which makes the results more accurate.

Code implementations:

TO START THE DOWNLOAD:

```
InputStream inputStream = connection.getInputStream();
byte[] buffer = new byte[1024 * 8];
long totalBytes = 0;
long startTime = System.currentTimeMillis();

int bytesRead;
while ((bytesRead = inputStream.read(buffer)) != -1) {
    totalBytes += bytesRead;
}
```

TO CALCULATE THE DOWNLOAD SPEED:

```
if (System.currentTimeMillis() - startTime > 3000) {
    break;
}
}
```

The WIFI specs page:

The **Wi-Fi Specs** page provides key information about the current Wi-Fi connection. It displays:

- The **RSSI** (Received Signal Strength Indicator) in **dBm**
- The **IPv4 address**
- The **MAC address** (which, unfortunately, does not work despite granting the necessary permissions)

Below this information, we included a **Wi-Fi signal icon** that visually represents the connection quality in different screen sizes. The icon changes **color** and **bar levels** depending on the **RSSI value**:

- **Red**: Weak signal (**-60 dBm or higher**)
- **Orange**: Moderate signal (**between -59 dBm and -50 dBm**)
- **Green**: Strong signal (**better than -50 dBm**)
- **X**: if there is no internet connection (**Wi-Fi on**)

We are still investigating why the **MAC address** does not appear as expected. Despite granting the required permissions, the value remains unavailable. This might be due to Android's increasing security restrictions on accessing device information.

Below is a snippet of the code used to retrieve the **RSSI, IPv4 address and MAC**



Figure 9: Weak signal



Figure 10: Moderate signal



Figure 11: Strong signal

The IP Address Calculator Page:

The IP calculator is a useful tool designed to compute essential network-related details such as subnet masks, network addresses, broadcast addresses, and usable IP address ranges.

This tool is particularly important for **network administration**, as it helps in efficiently allocating IP addresses and managing network traffic effectively.

Functionality:

When a user inputs an IP address and a subnet mask, the tool calculates:

- The **network address** (the base address of the subnet),
- The **broadcast address** (used to communicate with all devices on the subnet),
- The **usable host range** (the range of IPs available for devices).

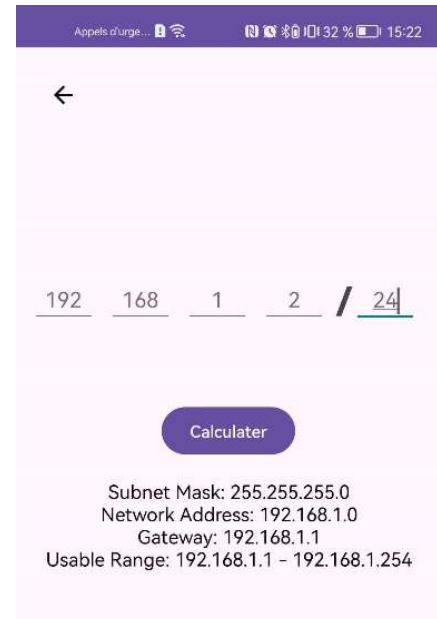


Figure 12: Mask calculator

Code Implementation:

TO COMPUTE THE SUBNET MASK AND NETWORK ADDRESS

```
// Calculer le masque réseau
int maskValue = 0xFFFFFFFF << (32 - mask);
String subnetMask = formatIp(maskValue);

// Calculer l'adresse réseau
int ip = (ip1 << 24) | (ip2 << 16) | (ip3 << 8) | ip4;
int network = ip & maskValue;
String networkAddress = formatIp(network);
```

TO FORMAT THE IP ADDRESS

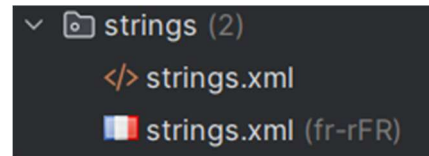
```
@SuppressWarnings("DefaultLocale") 4 usages
private String formatIp(int value) {
    return String.format("%d.%d.%d.%d",
        (value >> 24) & 0xFF,
        (value >> 16) & 0xFF,
        (value >> 8) & 0xFF,
        value & 0xFF);
}
```

Multilingual Support:

Our application supports **multiple languages** to enhance accessibility. By default, the app is in **English**, but users with a **French system language** will see the interface automatically translated.

To achieve this, we used **strings.xml** files:

- res/values/strings.xml (for English)
- res/values-fr/strings.xml (for French)



This method makes translations **easy to manage** and allows us to add more languages in the future if needed.

Summary:

This project allowed us to gain experience in **Android development**, particularly in:

- Managing multiple **activities** and using **intents** to pass data between them.
- Do **multilingual** applications.
- Implementing networking functionalities like **WIFI specs**.
- Calculating **IP subnetting** for networking applications.

While the application meets the basic requirements, future improvements could include:

- Adding **SQLite database** supports better data management.
- Fixing the **speed and ping tests** and the **MAC** address in the **WIFI specs**.

Overall, the project helped us build key skills in app development, problem-solving, and implementing networking features.

References:

- Stack Overflow: Shared preferences for creating one time activity, [online]. Available at: <https://stackoverflow.com/questions/23024831/shared-preferences-for-creating-one-time-activity> [Accessed on 05 February 2025].
- Stack Overflow: Save ArrayList to SharedPreferences, [online]. Available at: <https://stackoverflow.com/questions/7057845/save-arraylist-to-sharedpreferences> [Accessed on 05 February 2025].
- Youtube: Simple Login System with SharedPreferences | Android, by Khuram Memon, Available at: <https://www.youtube.com/watch?v=oT6wcWmHnNU> [Accessed on 05 February 2025].
- Medium: Get RSSI in Android Fragments [online]. Available at: <https://isurunuwanthilaka.medium.com/get-rssi-in-android-fragments-e2cef62948f2> [Accessed on 05 February 2025].
- Stack Overflow: How to get IP address of the device from code? [online]. Available at: <https://stackoverflow.com/questions/6064510/how-to-get-ip-address-of-the-device-from-code> [Accessed on 05 February 2025].