Mobile Computing

Practical Assignment #1 / Design and Development

Order and pay Android app for an electronics shop

The Acme Electronics Shop

1. Scenario

An electronics' products shop functions currently using a system where the clients can choose the wanted products in terminals available in the shop, pay for them, and collect them in a counter, already packed, before leaving the shop.

Clients shopping products (old way)

In order to improve the shopping experience, the company wants to allow the clients to observe, test and put the products virtually in a shopping list, using a supplied **app** and their own smartphones. To put a product (or some quantity of them) in the list, all the client must do is to read a barcode that is printed near the product. Info like the characteristics, make and model, and price can also be given, at any time, for the products in the list.

After finishing shopping (and eventually correcting the shopping basket, for instance eliminating some item) the customer should do the payment electronically using the smartphone, and a communication to a service. If the payment was validated, he will receive a validation token (a unique number) for the transaction.

After, he should now go to a printer terminal that will accept that token transmitted by NFC or using a QR-Code, producing a printed receipt (with the client identification and the purchased products and value).

Only in the possession of this receipt, the customer can now get the packed products in the shop counter.

The purpose of this exercise is to simulate the described scenario.

Clients perusing the available products

Receipt printer

When using the **app** the customers should first make a registration (only once, when they use the app for the first time) in the shop remote service, supplying some personal data (name, address, fiscal number (NIF) (9 digits)), and credit card data for payments.

2. Needed applications

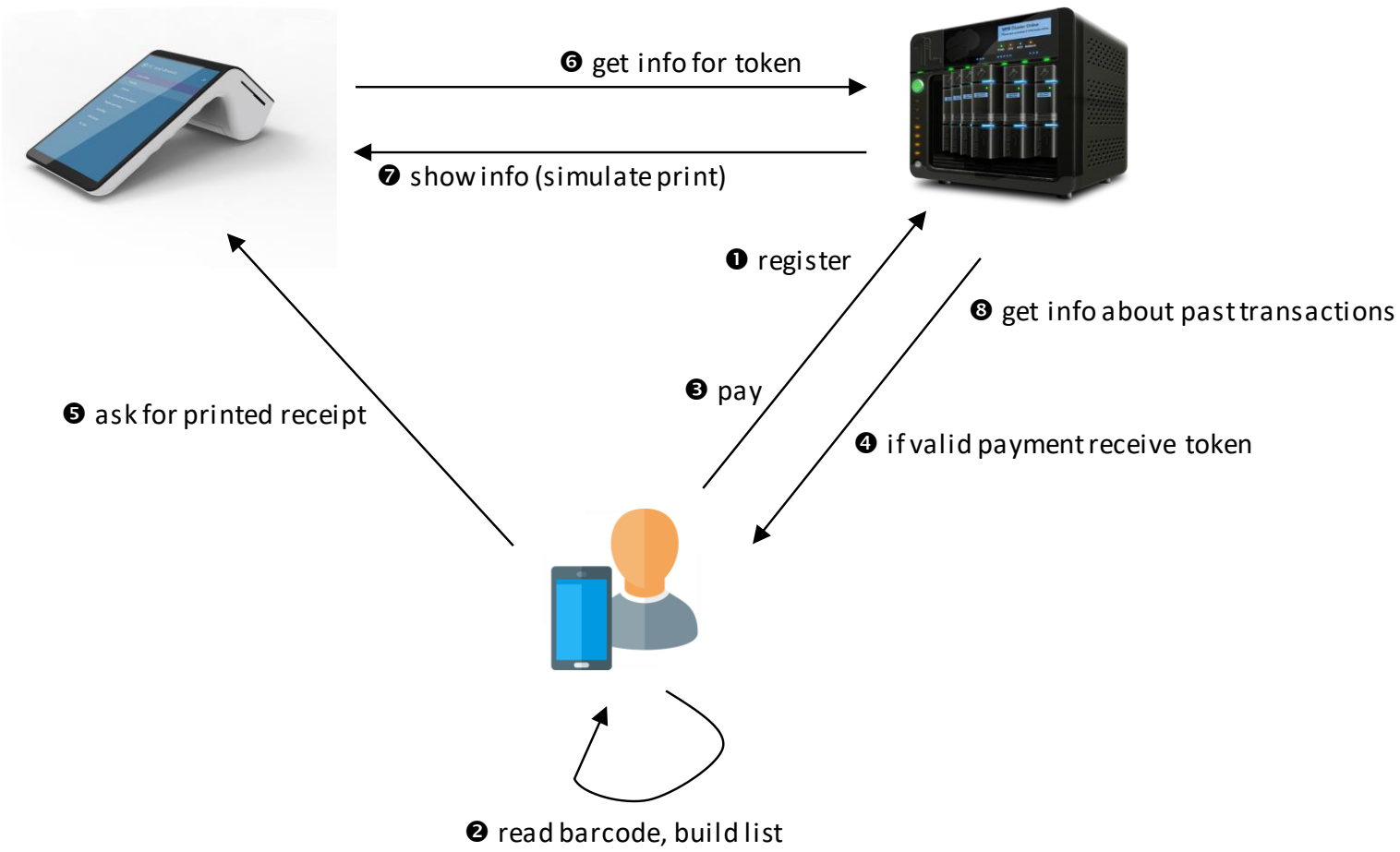
The system is composed of three different applications, namely:

- A remote service (a REST service with a defined API) located on the company server and available through Wi-Fi in the shop (it can be divided into several groups of operations: register, validate a payment corresponding to an order, emit a token, validate tokens and retrieve the information relative to that token for the printer, list previous purchases).
- The **Customer App** allowing him to register himself in the system, read product codes, show info for the products, show and manage the shopping list, make a payment, print the receipt, and also consult past transactions.

- The printer terminal should be emulated in an Android device with an app that can read tokens (using NFC and/or a QR-Code), consult the info corresponding to the token (if it is valid), and simulating the printing, showing the info on the screen.

3. Operations and interactions

The minimum set of operations and interactions between these applications are depicted in the following diagram:



They should perform, at least, the following:

1. Registration - the first operation the customer app should do is to register the customer in the shop service. The customer should supply his name, address, fiscal number (NIF) (9 digits), validation credentials and credit card information (type, number, validity, at least). For the validation credentials, the device should generate a 512 bit RSA key pair, store securely the private key and send the public key with the registration info. The server should take note of this key together with the rest of the info, generates a 128-bit UUID (which will serve as his user ID) and sends it back to the **app**. This value should be kept securely.

2. The user can now read barcodes (with the device camera) corresponding to products and collect them in the shopping 'basket' (supplying also a quantity). For each product in the list he can also retrieve more info from the shop server (like characteristics, price, make and model, etc.).

3. When the customer is ready to pay, the device contacts the shop service, and sends his user ID, and the basket content, signed with the private key stored when the registration was performed. Use "SHA256WithRSA" as the signature algorithm.

4. The server should now verify the signature, and if correct try to do the payment with the credit card associated at the registration (verify first the validity date). For this purpose, and in this simulation, just consider the payment done in 95% of the cases in a valid date. If the payment was performed, generate a unique token (an UUID) and send it back to the customer. Take note of products and total price, date and time, together with the generated token. If not, reply with an error message. The token, when emitted, is transmitted back encrypted with the user public key, as it will give access to the purchased goods.

5. The customer now obtains a printed receipt, transmitting the token to the printer using NFC and/or a QR-Code.

6. Before printing, the printer asks the server for the token verification, customer info, purchase info and total value. The token is valid only once.

7. If the token is verified, the printer shows the previous info on the screen (simulated printing).

8. At any time, after registration, the customer can ask for past transactions info, and display it. Before supplying it the service must be certain of the customer identity (without asking any info or passwords, and possibly without danger of replay attacks).

4. Communications

All the communications with the server are done using the internet and the http protocol (in a REST service), over Wi-Fi (you can use your PC or phone as a Wi-Fi hotspot, or if using emulators the PC internal network).
The communication between the customer app and printer (operation 5) should use NFC if your group has two NFC phones, or a QR-Code and camera, if not (optionally you can implement both).
The QR code technique can also be used between a physical phone and an emulator (QR-Code shown on the emulator).
If you don't have any Android physical phone available, you can establish and use a TCP/IP connection between two different emulators.

Note: if you are using only the Google emulators the channel between them has to be TCP. See https://developer.android.com/studio/run/emulator-networking.html and the section "Interconnecting emulator instances" for instructions.

5. Design and development

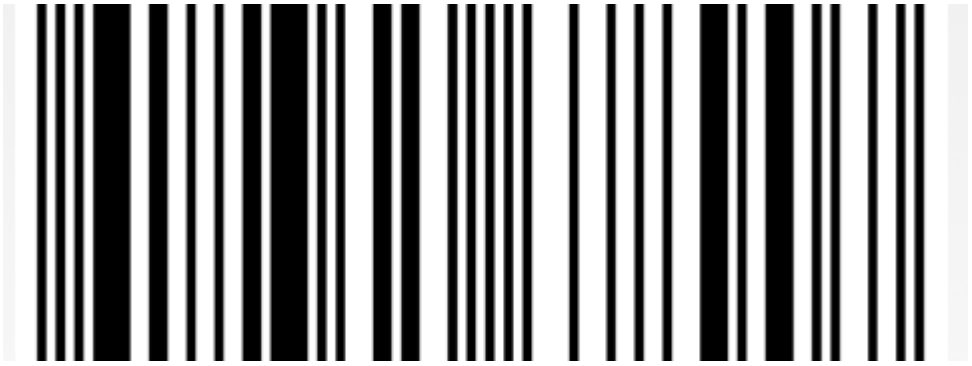
You should design and implement the set of applications capable of simulating the described scenario. The applications should have a comfortable and easy to use interface. You can add any functionalities considered convenient and fill any gaps not detailed. For instance, to increase security, you can implement a local login, to insure that only the user who has performed the registration has access to the app.

You should also write a report, describing the architecture, data schema, included features specially security and data representation, and performed tests. The applications way of use should also be included in the report, presenting a meaningful screen capture sequence illustrating the operations.

Notes:
For barcode and QR-Code production and reading you can use the **Google ZXing** scanner app and library.

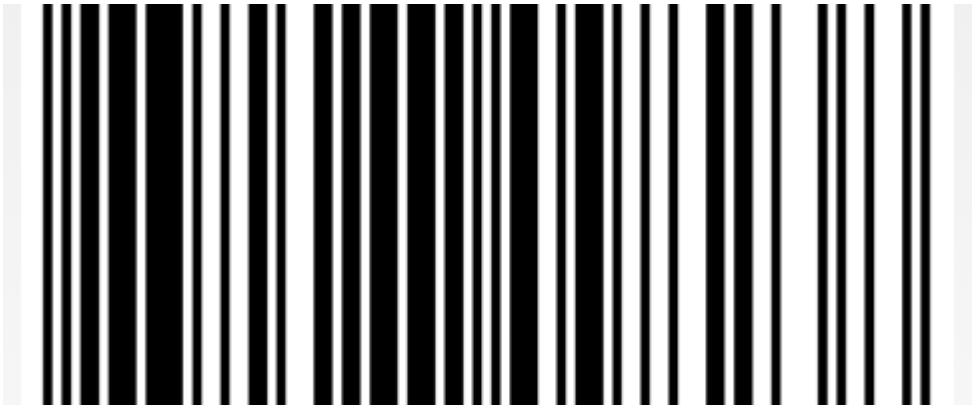
The barcode to use should be in the UPC-A format representing 12 digits. The 12th digit is only an error detection digit and is derived from the other 11. So the value represented in the UPC-A is composed by the first 11 digits.

The next page contains some sample barcodes with their values.



61234567890

12853478357



83248709823