[1. Introduction 2](#_Toc60660401)

[1.1 Purpose 2](#_Toc60660402)

[1.2 Scope 2](#_Toc60660403)

[1.3 Definitions, Acronyms, Abbreviations 2](#_Toc60660404)

[1.3.1 Definitions 2](#_Toc60660405)

[1.3.2 Acronyms 2](#_Toc60660406)

[1.3.3 Abbreviations 3](#_Toc60660407)

[1.4 Revision History 3](#_Toc60660408)

[1.5 Reference Documents 3](#_Toc60660409)

[1.6 Document Structure 3](#_Toc60660410)

# 

# 1. Introduction

## 1.1 Purpose

This document aims to provide a detailed representation of the software described in the RASD document. it goes deeper into details and shows an overview of the architecture of the CLup application.

It aids in the critical analysis of the problem and the proposed solution by creating an overall guidance for the programmers who will develop the application.

For this scope in this document, it is possible to find all the components of the system with the interactions among them, their interfaces, the runtime behavior and the deployment. The design characteristics are provided together with the design patterns adopted and some algorithms which highlight the more critical aspects.

This document also gives a description of how the implementation, integration and testing plan will be like.

## 1.2 Scope

As previously shown in the RASD Document, the scope of CLup Application is preventing crowds and situations of possible danger during the period of Coronovirus emergency.

Store managers can regulate and monitor the affluence of people in the shop by adding their stores with all the related information to CLup. In this way the customers can do the shopping in a safer way with the possibility to:

* obtain a position in the queue of a shop by *taking the ticket* online for the current day. With this basic service the customer can check in any moment the status of the queue by receiving the remaining time he/she has to wait and the number of people above him/her. Moreover, periodical notifications are sent to him/her until his/her turn arrives.
* receive a time slot by *booking a visit* for the following days. In this case CLup Application gives the possibility to provide the categories of items the customer has the intention to buy and an estimation of the duration of the visit.

The system ideated prevents the customer from queuing in front of the buildings with other people and allows them to respect the distance rules imposed by the Government.   
To make the lining up mechanism effective, when a customer makes a reservation (ticket or visit) the system generates a QR code which must be scanned at the entrances and exits of the shop. This allows to track the number of accesses in the store every day, information needed to build statistics and to better organize the store.

CLup application also involves people who are not able to use the necessary technology or do not own a device: they can directly go to the store and exploit the fallback option of handing out tickets on the spot. This is made possible through the presence of one or more totems around the building.

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

* **Custome**r: who signs in the application with the purpose of doing shopping.
* **Store Manager**: who adds a store in the application and has the purpose of organizing it according to the new rules introduced to contrast the coronavirus pandemic.
* **User:** who signs in the application and uses the available services for him/her purposes. The user can be a customer or a store manager.
* **Demographic**: particular sector of population (children, seniors, adults, …).
* **Ticket:** the number received which corresponds to the position in the queue.
* **Totem**: multimedia structure which allows people who cannot use the application to take a ticket.
* **QR Scanner**: digital structure used by the customers to scan their tickets when they enter/exit the stores in order to make the shop managers monitor the affluence of people in their shop.
* **QR Code**: bidimensional matrix composed of black modules put in a square schema used to memorize information about a ticket.
* **Reservation**: is a general term used to indicate either a taken ticket or a booked visit.
* **Book a visit:** refers to the option for the user to book a visit in a selected shop for the following days.
* **Take a ticket:** refers to the option for the customer to have a position in the queue of a shop in the current day.
* **System**: it is the software that is the objective of the document.

### Acronyms

* **RASD:** Requirement Analysis and Specification Document
* **DD:** Design Document
* **GPS:** Global Positioning System
* **UI:** User Interface
* **MVC:** Model View Controller
* **API:** Application Programming Interface
* **HTTP:** Hypertext Transfer Protocol
* **TLS:** Transport Layer Security
* **JSON:** JavaScript Object Notation
* **REST:** Representational State Transfer
* **RDB**: Relational Database
* **DBMS:** Database Management System
* **RDBMS:** Relational DBMS

### Abbreviations

* **Gn:** n-thgoal
* **Rn:** n-th requirement

## Revision History

## 1.5 Reference Documents

* available slides on Beep
* R&DD Assignment A. Y. 2020-2021

## Document Structure

**Chapter 1:** the first chapter - which is an introduction of the design document - provides the purpose and scope of the document. It also includes the definitions, abbreviations and acronyms useful to help the reader to better understand what is written. It is also here that it is possible to check the structure of the document.   
  
**Chapter 2:** it includes all the architectural choices taken for the development of the application by highlighting the components and interfaces of the system, presented both from a high level and detailed point of view (component view, deployment view, runtime view). All the design decisions are provided in this chapter, together with the patterns adopted.

# **Chapter 3:** this chapter shows how the system will look like by providing the user interfaces of the main functionalities. Some of the mockups – which were presented in the RASD Document – are shown also in the Design Document. **Chapter 4**: it maps the requirements and goals defined in the RASD document with the design components provided in the Design Document. **Chapter 5:** this chapter provides the strategy used to decide the order of implementation, integration and testing of the subcomponents of the system. It also provides the type of system testing. **Chapter 6:** it shows the effort (number of hours) spent by each component of the group for the draft of the document.

# 2. Architectural Design

## 2.1 Overview

The architecture of the application is a distributed one organized according to the three-tier architecture, that divides it into three logical and physical computing tiers:

-**Presentation tier:** the presentation tier is the front end layer and hosts  the user interface. It is the  communication tier of the application and consequently it manages the interactions of the end user with the system.

Its main purpose is to display information and highlight in a clear way  the functions to the users

-**Application  tier**

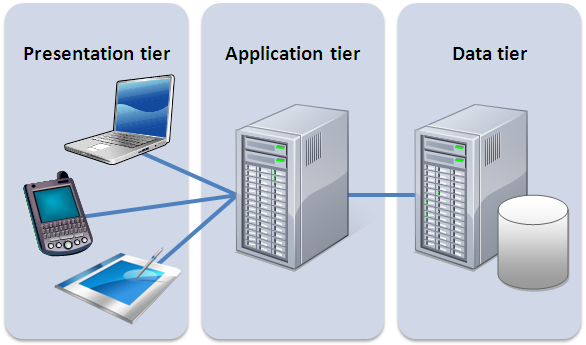
The application tier is also known as the business logic tier or middle tier and represents the core of the application.

In this tier, information collected in the presentation tier is processed by handling  business logic, that is a specific set of business rules.

It also moves and processes data between the two surrounding layers: the presentation tier and the data tier cannot communicate directly with one another.

**-Data tier**

The data tier, sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored, managedand retrieved from a database or file system. It hosts the data( which are persistently stored in a DBMS) and it is invoked to provide one or more services.

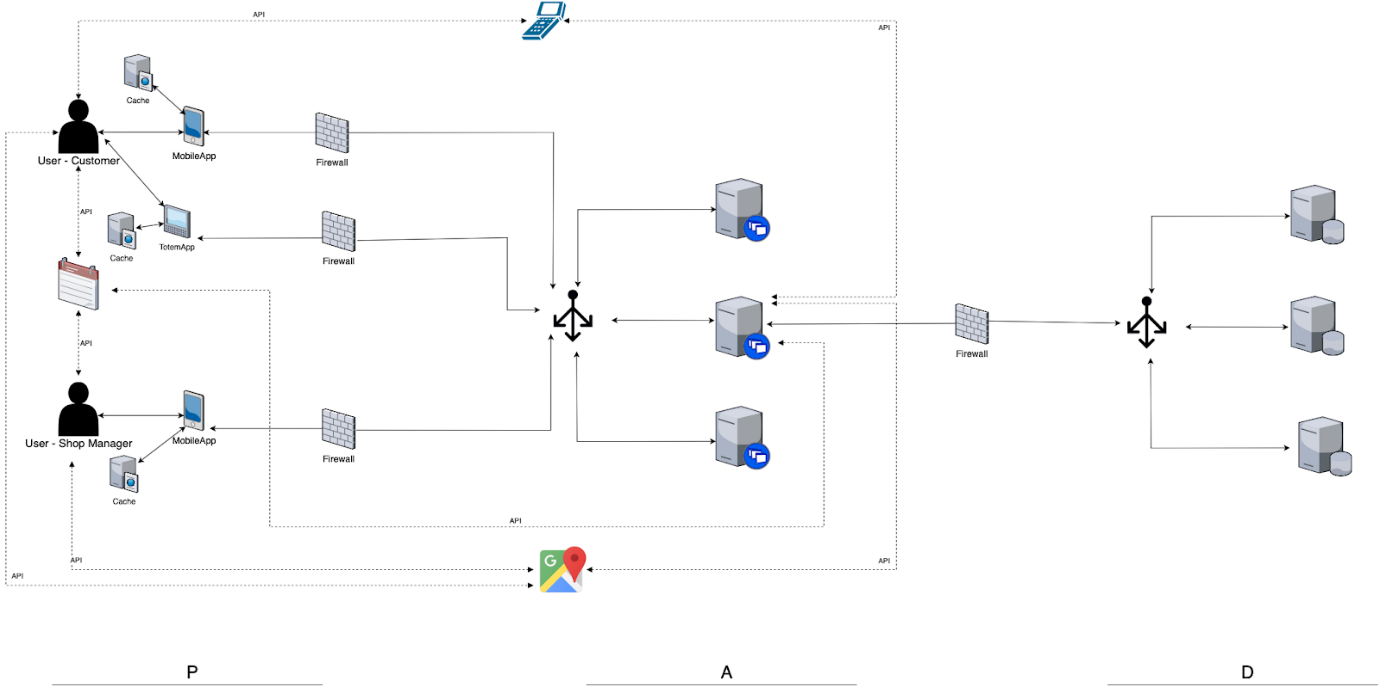
****

The hardware tiers provided above are the machines on which the three logic layers that structure the application are subdivided: each logic layer (Presentation layer, Business Logic Layer, Data Access Layer) has its own dedicated hardware.

Three tier application was the prevailing one for client-server application and because of its benefits. The main one is the fact that each tier runs on its own infrastructure, can be developed simultaneously by a separate development team, and can be updated or scaled as needed without impacting the other tiers.  
Moreover, this kind of architecture allows interaction with the backends of many different applications despite having one browser.

### 2.1.1 High-level components

The following diagram provides a high level representation of the components of the system with their interactions also by highlighting the correspondence with the three Presentation, Application and  Data levels.

****

**Figure 1. System Architecture**

The presentation level is represented by the Mobile App, the Totem App and the QRScannerApp  that the users interact with in order to communicate with the Application level.

Specifically, the user customer can interact with all of them, while the user - shop manager only with the Mobile App.    
The Application level is the main component, since it holds the business logic of the system: it interacts with the users and it has access to the DB servers, where all the useful data is stored.

Moreover, in case of Mobile Apps, the Application Servers, which are the middle tiers, have access to the APIs of Maps - provided by Google Maps - and Calendar - provided by Google Calendar.

To go into further details, the Presentation Layer exploits a Cache to make the requested data available when needed and to fasten the communication.  The communication between this layer and the Application one involves the use of a Firewall - which establishes a barrier between the internal network and incoming traffic from external sources - and of a Load Balancer. The latter is used to distribute the working load among the various Application Servers, that - due to security reasons and to improve reliability - are replicated. That’s exactly why the Application Servers cannot exploit the use of caches.

The Business Logic layer communicates with the  Data Access one both to retrieve information and to store data from/into  the Database Server (DBMS). Also in this case and for the same reasons explained before, the Firewall and Load Balancer components are exploited.  Also Database Servers are replicated to manage the risk of data loss and improve the safety of the system.

### 2.2 Component View

The Component Diagram below shows the main components of the system and the services provided by them. It captures the physical structure of the implementation and it is a static diagram since it describes several instances without explaining how they collaborate with each other, but highlighting that at a certain point in time they will collaborate.

The focus is  on the Application Layer - which contains the business logic - while the other two layers (Presentation and Data) are represented in a simplified way (as black boxes) only to show the interaction with the Application level.

Since the Totem is part of our system and we have the intention to implement them, it is represented as a Component in the diagram. Instead the QR scanner is external and so we only include in our system the interface provided by it.

Regarding the Application level, the components proposed are the following ones:

Copia disegno

* **Router:** this component takes the invocations from the clients and directs them to the right components of the Application Server. It is the dispatcher of the requests received and for this reason it must be always active.

* **EnterAppManager:** this component is used to allow the users to enter the application and exploit the services provided. It regards the Registration and the Login activities and it interacts with the DBMs both in writing - to store data of the users - and in reading - to check the correctness of the inserted values.

The services provided are:

-*LoginService*: it manages the authentication of a user to the application.

*-RegistrationService*: it manages the registration of a user to the application

* **TakeTicketManager**: this component deals with the requests of the tickets done by the customer-users.  It stores the data related to the ticket in the database and communicates with the ScheduleManager:
  + to get the queue corresponding to the selected shop
  + to check if the request of the ticket can be accepted or refused (in case the queue is full until the closing time, the user selects a shop when it is closed, a customer has already taken a ticket or booked a visit at the same time of the ticket)

* **TotemManager:** this component manages the requests of taking a ticket from the users who want to take it on the spot. WIt includes the printing of the QR codes.

* **BookVisitManager:** this component manages the requests of the reservations for a visit made by the customer-User. It saves the data related to the booked visit in the database and communicates with the ScheduleManager:
  + to identify the available time slots for the selected day for the visit
  + to decide whether to accept or refuse the request (rejected in case the user books two or more overlapping visits, takes a ticket which overlaps with the visit, selects a day that has not available time slots)

* **DataManager:** this component handles different requests made by the users  concerning the visualization and/or modification of  data.

The related services are the following ones:

-*checkPersonalInfoService*: it allows the user to visualize the data related to his/her registration and decides whether to modify it or not.

-*QueueStatusService:* it allows the user customer to check how many people are above him/her in the queue of the shop related to the taken ticket and the remaining waiting time

-*BookedVisitsService:* it allows the user customer to check all the booked visits with the related QR codes

*-QRsService:* it allows the user customer to visualize all the QRcodes related to the reservations done and to download them. It is also used during a “take a ticket ” or a “book a visit” action if the customer decides to download the QR code of the reservation just made.

-*ShopsInfoService:* it allows the user Customer to visualize the list of shops with the related information(how many people are already in the corresponding queue, which categories of items the store provides, the address and the opening and closing time)

. In case of GPS localization provided, it also sorts the list from the closest to the furthest shop   
-*myShopsService:* it allows the user Shop Manager to visualize and modify the information of the shops added by him/her, including the daily reports.

-*addShopService:* it allows the user Shop Manager to add a shop to CLup

-*CancelTicketService:* it allows the user-customer to cancel the ticket taken

*-CancelVIsitService:* it allows the user-customer to cancel the visit booked

-*ModifyShopService:* it allows the user-shop manager to modify the info of one of his/her shop. It also includes the deletion of a shop.

* **ScheduleManager:** this component has the purpose to manage the schedule of each inserted shop and, in case of Book a visit option, to compute the free slots of a selected day. It interacts with the DBMS in reading to get the data related to a reservation (also when needed by other components) and to send periodical notification to the user customer about the reservations when it is almost his/her turn(in case of take a ticket) and two hours before (in case of Book a visit). For this scope the ScheduleManager interacts with the GoogleMapsService Component.

* **DBMS:** it allows all the other components to interact - to store, read, modify data - with the database. It must be always active to receive the requests of the components.

The external components of the system are the following ones:

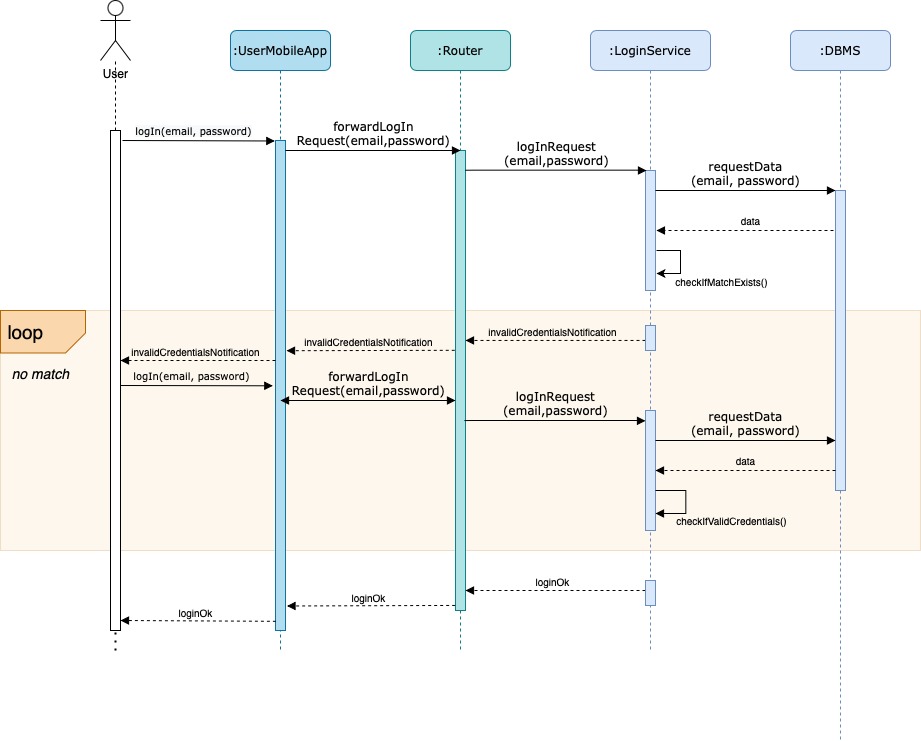
* - *GoogleMapsService:* it is the component that provides the interface through which the user can visualize the map of the city
* *-GoogleCalendarService:* it  is the component that provides the interface through which the user can visualize the days of the month in order to select a date for a visit.
* -*QRScannerService:*it is the component that provides the interface required for managing the entrances and exits of customers from/to a shop.

2.3 RunTime View

The sequence diagrams provided above are useful to show the dynamic behavior of the system by highlighting its internal components (the external ones are not considered in this section).   
In addition to modeling the behaviors, the diagrams also model the flow of control and illustrate typical scenarios.

1. **Login**

This sequence diagram shows the Login made by a User (both customer and Shop Manager).



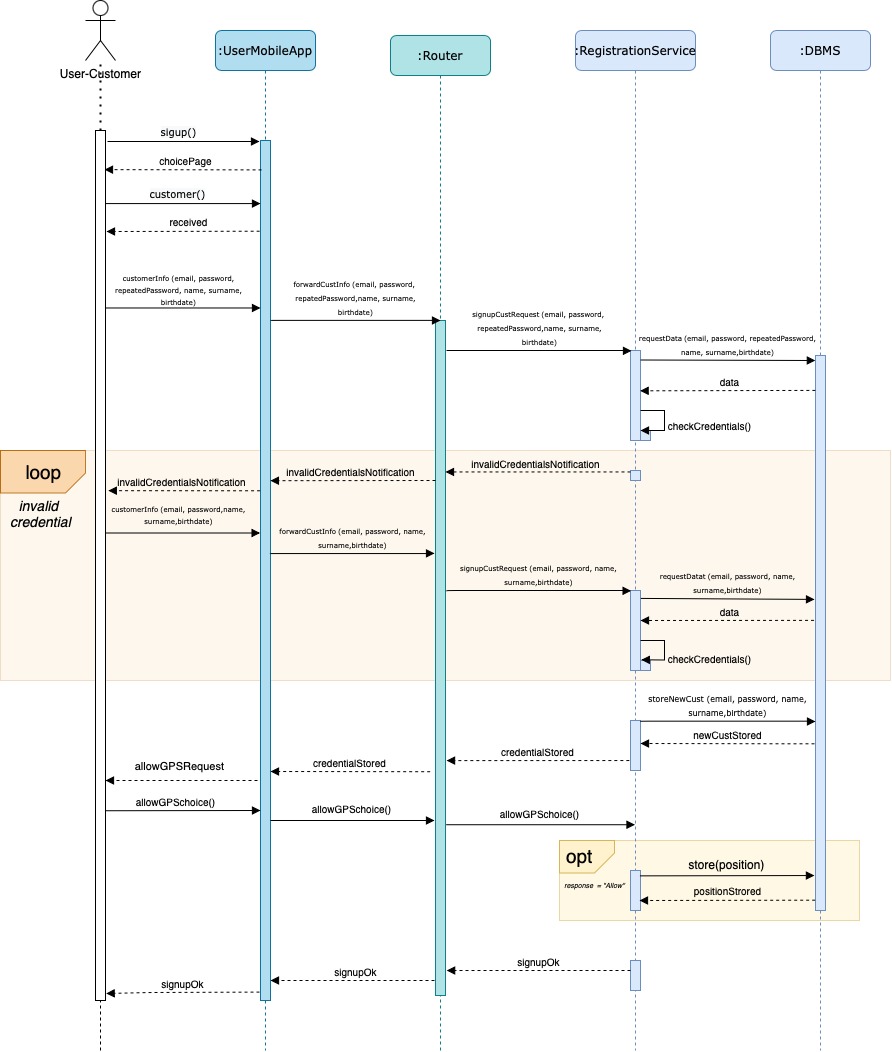
The user opens the mobile application on his/her device and fills the mandatory fields with the email and password, already set during the registration.

After receiving the login request by the UserMobileApp, the Router sends it to the LoginService component, which manages the authentication of the user. To do this the component asks the data(email,password) of the user to the DBMS - who sends it back- and checks if there is a correspondence between the email and password inserted. If there is a match, the user can successfully login to CLup application, otherwise he/she has to try again.

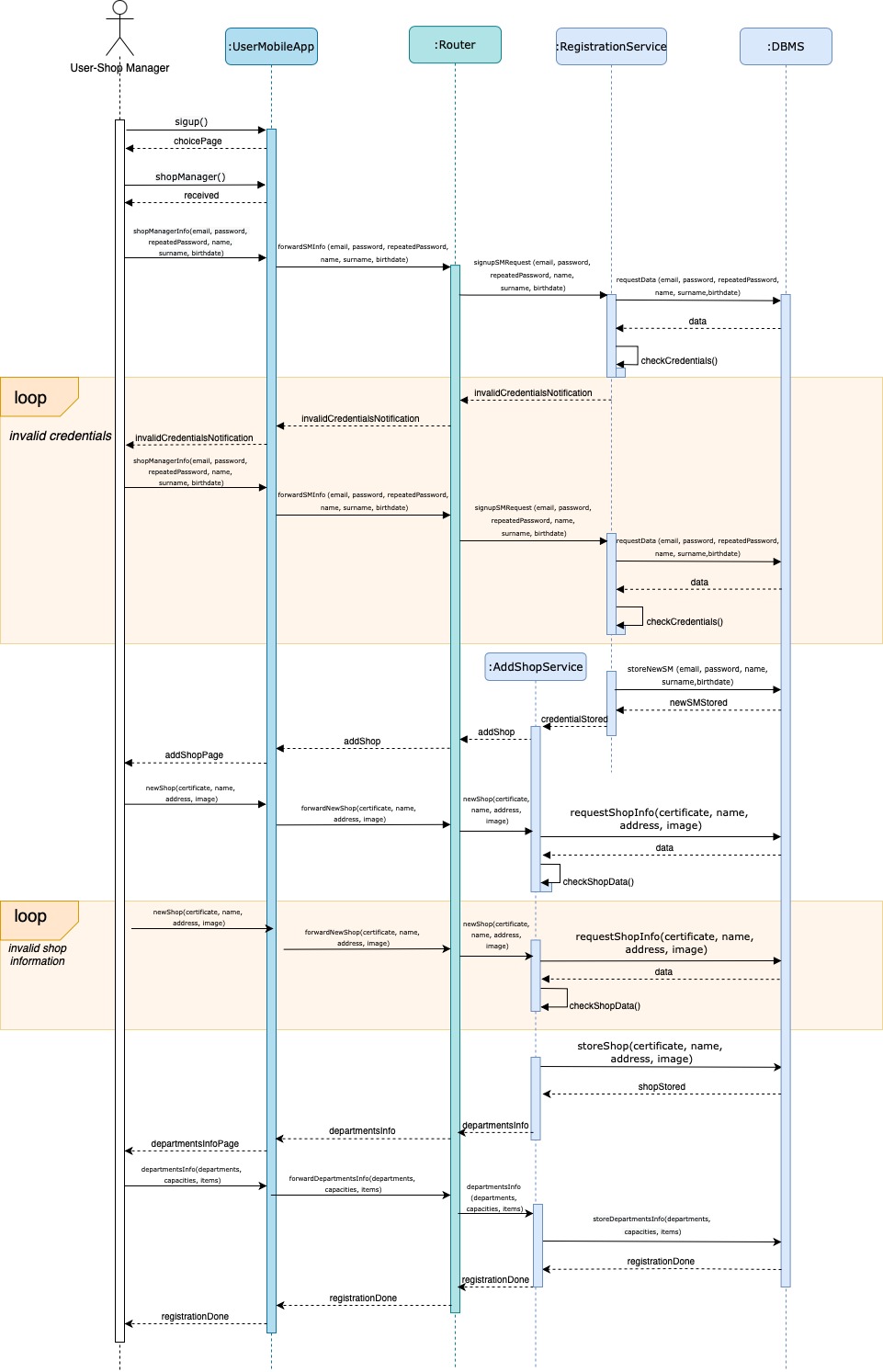
1. **Sign up User**

The following two diagrams show the registration of the user.

The first one represents the Sign up of the User Customer.

****

The user customer signs up through the mobile application. The MobileApp sends as response the page with the choice between “Customer” and “Shop Manager”. Once selected the “Customer” option, the request with the mandatory fields (email, password, Name, Surname) and the optional field (Birthdate) is forwarded by the MobileApp to the RegistrationService (by passing through the router). To check if all the data inserted is valid, the RegistrationService component exploits the DBMS which sends it back the requested data.   
-If the result is negative (invalid credentials) the user has to make the registration again.   
-If the result is positive (*credentials ok)*  the data is saved in the database and the RegistrationService component, after receiving the response from the DBMS, forwards it to the Router and User Mobile app, which sends the request of GPS localization to the customer who can decide whether to provide it or not (by clicking on “Allow” or “Do not allow” button). If the user gives access to it, the information is stored in the database.



The following diagram shows instead the SignUp of the User-Shop Manager. As in the previous case the user, after selecting the “Shop Manager” option, fills the mandatory and optional fields about him/herself. After the credentials are checked through the DBMS, if they are correct, the RegistrationService component forwards the response received by the DBMS to the AddShopService component, which stores the information about the shop manager in the database and  asks the user to add a Shop with the related information(certificate, name, address, picture) about it. This data is sent back to the addShopService (passing through the mobile app and router), which creates a shop and asks the Shop information to  DBMS to check whether the shop already exists and if the certificate is valid.    
- if everything is correct, data is stored in the database and the addShopService component sends the request of inserting Departments’ info(capacities,items) to the user, which provides it.   
- if the information of the shop is invalid, the user has to try again. The same happens if the credentials inserted by the user are not valid.