POLITECNICO
MILANO 1863

CLup

Design Document

Authors:
Francesco Fulco Gonzales (10614882), Alberto Latino (10600138)

Version: 1.0

Date: January 10, 2020

Professor: Elisabetta Di Nitto

Contents

1. Introduction

1.1. Purpose

The purpose of this document is to underline and explain in detail the design of the hardware and software architecture of the CLup application. The analysis has been made at different levels to underline different views of the same system.
The following are the main levels explored in the document:

- an high level architecture
- the components that make up the system
- the deployment view of the system
- the interfaces provided by components
- the pattern and technologies used in the system
- the User Interfaces provided by the system

Moreover, this document describes the design characteristics required for the implementation, as well as an overview of the implementation, integration and testing plans.

1.2. Scope

Here we give a quick summary of the application's purpose. For a more thorough explanation you may refer to the RASD.

CLup is a software system that has the purpose of implementing social distancing rules in supermarkets, and to save customers' time by allowing them to book a virtual queue ticket directly from home instead of lining up physically. CLup includes two main functionalities:

- Basic service: allows users to virtually line up at the store by booking a digital ticket (with QR code) and notifies them shortly before their number is called;

- Book a visit: users can book a ticket (with QR code) for a visit at the store, indicating the time slot when they want to shop, the expected duration of the visit and product categories they intend to buy.

1.3. Definitions, Acronyms, Abbreviations

1.3.1. Definitions

- **Ticket:** Virtual item associated to a QR code that allows users to virtually line-up and uniquely identify their visit to the supermarket
- **Time Slot:** Specific span of time that can be chosen by the user to go shopping.
- **Statistics:** Facts about visit duration and time spent at the store inferred through previous data analysis
- **QR Code:** a machine-readable code consisting of an array of black and white squares, used to store information of the ticket that can be read by a

device equipped with a camera

1.3.2. Acronyms

- **UI:** User Interface through users require available functionalities.
- **UX:** User Experience, way in which the user interacts with the service.

1.3.3. Abbreviations

- **G.i**: i-th goal
- **R.i:** i-th requirement
- **SM:** Store Manager

1.4. Revision history

- Version 1.0, released on 10/01/2020

1.5. Reference Documents

- CLup Specification Document
- Course Slides

1.6. Document Structure

The other sections of the Design Document (DD) are organised in this way:

- **Chapter 1**, Introduction: presents the scope, purpose, and structure of the document and provides the definitions, acronyms and abbreviations used in it.
- **Chapter 2**, Architectural Design: presents an in-depth description of the System's architecture. It defines the main components, the relationship between them and the deployment of components. There are different views and levels of analysis of the components plus some subsections useful for identifying how the components interact and for the architectural styles and patterns.
- **Chapter 3**, User Interface Design: it's a complementary section of what was included in the RASD. It includes the definition of the UX processes through a model that represents the flows between interfaces.
- **Chapter 4**, Requirements Traceability: a complementary section of what was included in the RASD. It contains all the requirements and shows the relationship between them and design choices done in order to satisfy them.
- **Chapter 5**, Implementation, Integration and Test Plan: it shows the order of the implementation and integration of all the components and subcomponents, explaining how the application will be tested.
- **Chapter 6**, Effort Spent: it's a section that contains a table for identifying the hours and the effort spent by the team to deliver the Design Document.
- **Chapter 7**: in this last chapter are mentioned the references to documents or sites about technologies, patterns and architectures used in this document.

2. Architectural Design

2.1. Overview

The CLup software will be developed using a three layer architecture: Presentation, Business logic and Data layer. Here these layers are analyzed more in details:

- **Presentation Layer:** it shows the user the state of the application, allows him to access functionalities and manages the interactions.

- **Business Logic Layer:** it is the core of the application and implements the main functionalities. In particular, manages the data coming from users and checks their validity in order to subsequently perform operations. When operations made by users are logically accepted, this layer also allows data to flow from the other two layers.

- **Data Access Layer:** it stores all data useful for the application. From users' personal information such as username and password to other useful data used to carry out functionalities. This layer contains the database and makes information available to other layers.

The application has to follow the Client - Server architecture. Client and server are going to communicate with each other through a working internet connection. More in detail, the presentation layer will be implemented on the Client, while the Business layer and the Data Access layer will be on the application server and database server respectively. Some of the logic might be moved to the Client. The architecture will be event driven, meaning that the system that receives events has to react to them producing certain outputs. In particular, the interaction between client and server will work as follows:

1) The user or the store manager clicks on interactive components of the UI generating an event.

2) Once the View catches the event, it is turned into a method call to the server.

3) The server analyzes the request made by the user and eventually performs some operations that could require access to the database.

4) Then, operations' output will be sent back to the user.

Depending on the functionality required by the user the server reacts in different ways:

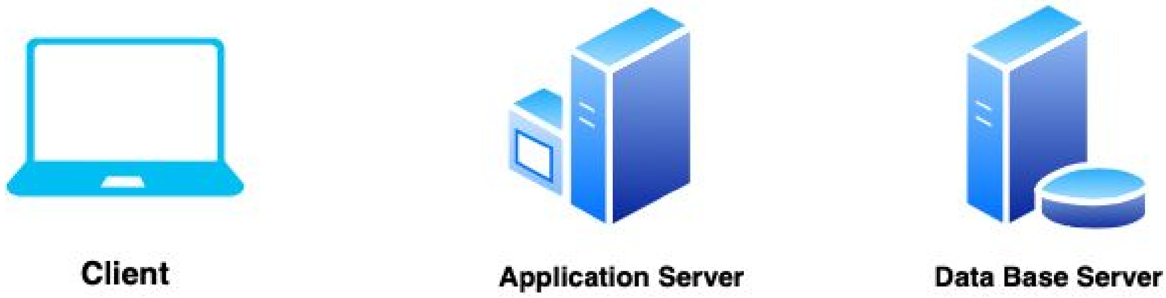
- **Line Up Service:** After the supermarket selection, the user clicks on the “Line-up” button and inserts the means of transport he is going to use to reach the store. Then, data from the request is sent to the server that analyzes them to check their validity. Afterwards, the server accesses the database through the Data Access layer to retrieve the current state of the virtual queue at the supermarket. Then if certain conditions are satisfied, a new ticket is generated, stored in the database and sent to the user.

- **Book a visit Service:** After the supermarket selection, the user clicks on the “Book a Visit “ button and inserts data regarding the timeslot and products’ categories chosen. Then data is sent to the server that checks their validity. When certain conditions are satisfied, the system generates a ticket and stores it into the database, updating other related information such as departments’ capacities and products’ availability. At the end, the ticket is sent to the user.

2.1.1. High Level Components

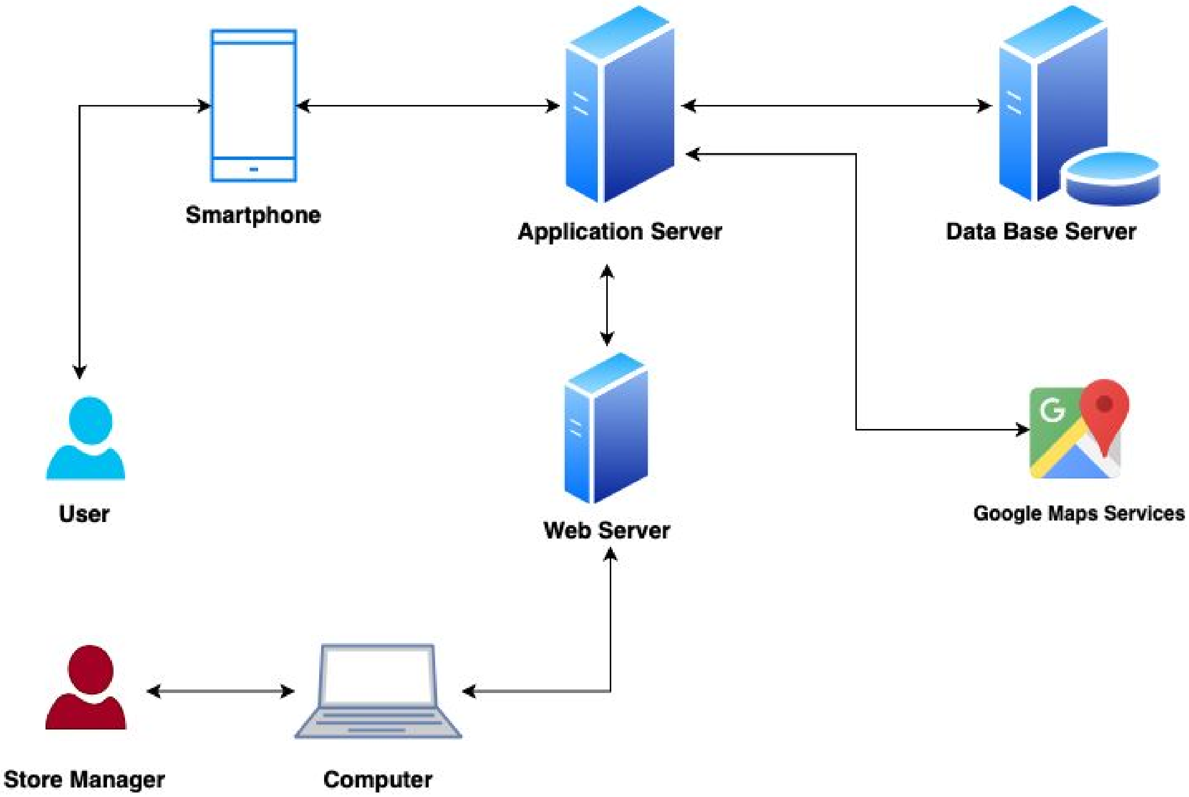
As already mentioned, the architecture will be developed on three tiers and has to follow the Client - Server paradigm. More precisely, the three tiers will be physically separated since their software is going to run on three devices in different places.

Three Tiers



The figure above briefly describes the three tiers architecture. In particular there are the Client that implements the presentation layer, the Application Server that implements the business logic and the DBMS server that manages the data flow from and to the database.

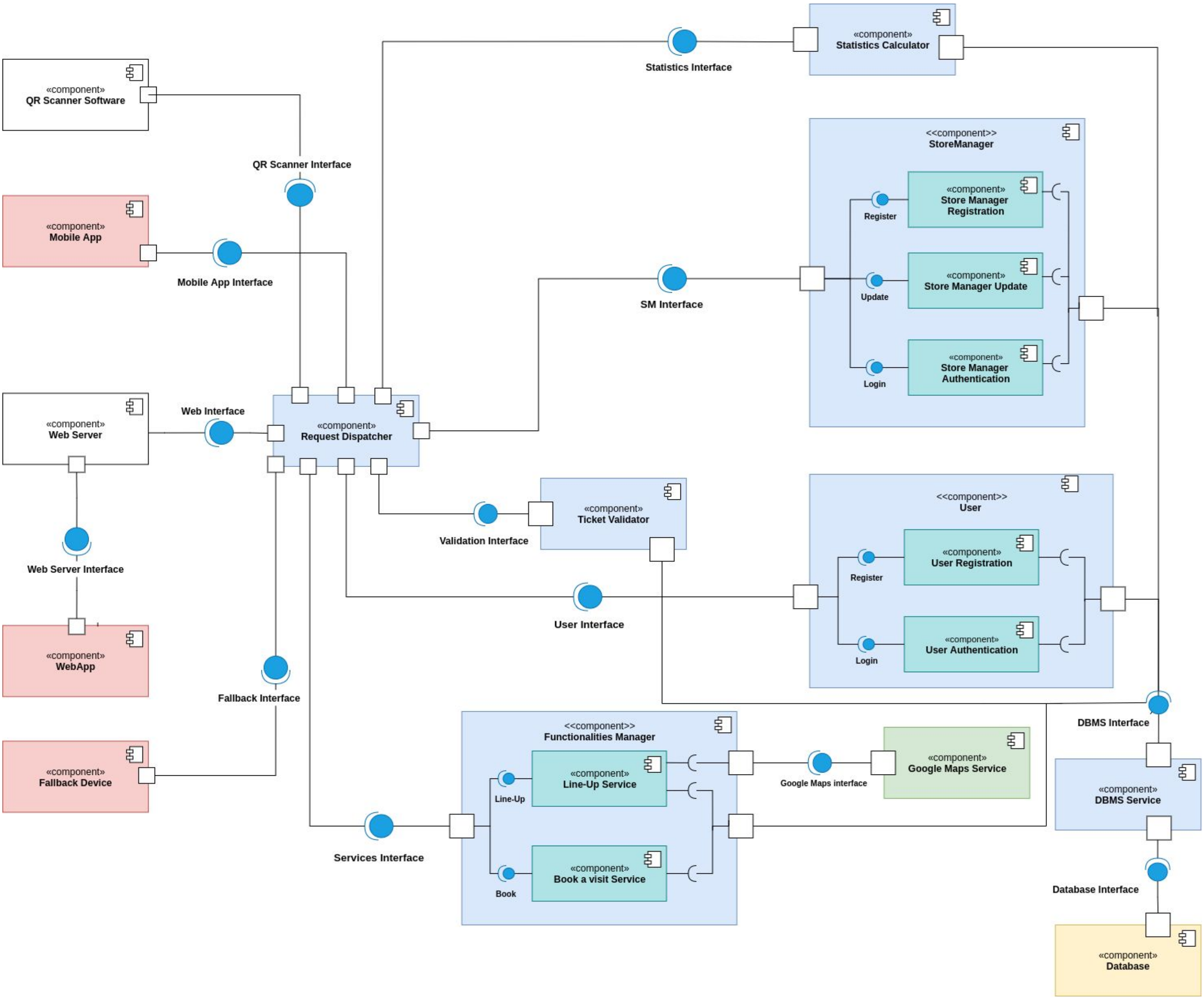
Three Tiers in detail



The figure above describes the architecture in detail, highlighting the devices used by two different users and the servers' structure. Google external service is also shown.

2.2. Component View

Here is the component view of the software. All the business layer's components are colored in blue, the presentation layer is colored in red, the data layer in yellow.



In the figure above, the main components of the CLup software are shown to make more understandable how the business logic works.

In particular, the software's components are:

- **SM Access Manager:** This software component, that belongs to the Business Layer, has been created to manage all the access requests from Store Managers. More in Details there are two subcomponents:
  - ❑ **SM Registration**: It has the role of accepting registration requests from store managers. This component checks whether the data inserted by the manager are valid to complete the registration of the store and the manager himself.
  - ❑ **SM Authentication**: Once the store managers have registered themselves and their store, they can access the personal area. This component checks login credentials' validity.
  - ❑ **SM Update:** This component allows a store manager to update his store's info, such as the name of the store, location and opening hours, departments and product categories.
- **User Access Manager:** This component of the Business Layer allows users to register and login. It is made of two subcomponents:
  - ❑ **User Registration:** It checks whether data inserted by the user are valid or not in order to allow him to sign up. In particular, it is fundamental to check the uniqueness of the email associated with the account.
  - ❑ **User Authentication:** Once the user has signed up, he can login into the mobile app by inserting username and password previously specified during the registration process.
- **Functionalities Manager:** This component is the core of the CLup's Business Logic. It has the role of providing the main services to users:
  - ❑ **Line-Up Service:** Whenever a user requests a ticket for virtually lining-up, this component firstly sends GPS data and chosen means of transport to the Google Maps Service in order to get back the commute time. Then the Line-Up component accesses the DB through the DBMS in order to get the queue status and to issue a new ticket with the associated waiting time.
  - ❑ **Book a Visit Service:** Whenever a user requests a ticket for the selected time slot and for the selected products' categories, this component does all the necessary checks, accessing the database, to issue the requested ticket.
- **Ticket Validator:** This component of the Business Logic has access to the database through the DBMS service and serves as controller for the QRCode Scanners placed at the entrance of the stores to check whether the scanned ticket is valid or not. Furthermore, once a ticket has been scanned and validated, it is used to store the entry time and the exit time of users.

- **Google Maps Service:** Third party service provided by Google that computes the time needed to reach the store, given the means of transport specified by the user and location of the user and the selected store.

- **DBMS Service:** This component is the interface between the business layer and the data layer and is the only one belonging to the former that can directly access the database. It receives requests from other components of the Business Layer and stores, retrieves and updates data.

- **Web Server:** Server that responds to the requests sent from the web app of the store manager.

- **Web App:** This component interacts with the store manager and communicates with the Web Server through http protocol in order to provide the software's functionalities.

- **Mobile App:** Software downloaded and installed on the user's mobile device. Once the user has logged in, this component directly interacts with the user, registering events generated by him. It is able to make requests to the server in order to provide the CLup's main services to the user.

- **Request Dispatcher:** Once requests have arrived, this component dispatches them to the several components that have to complete the required task. Every method call is forwarded to another specific component.

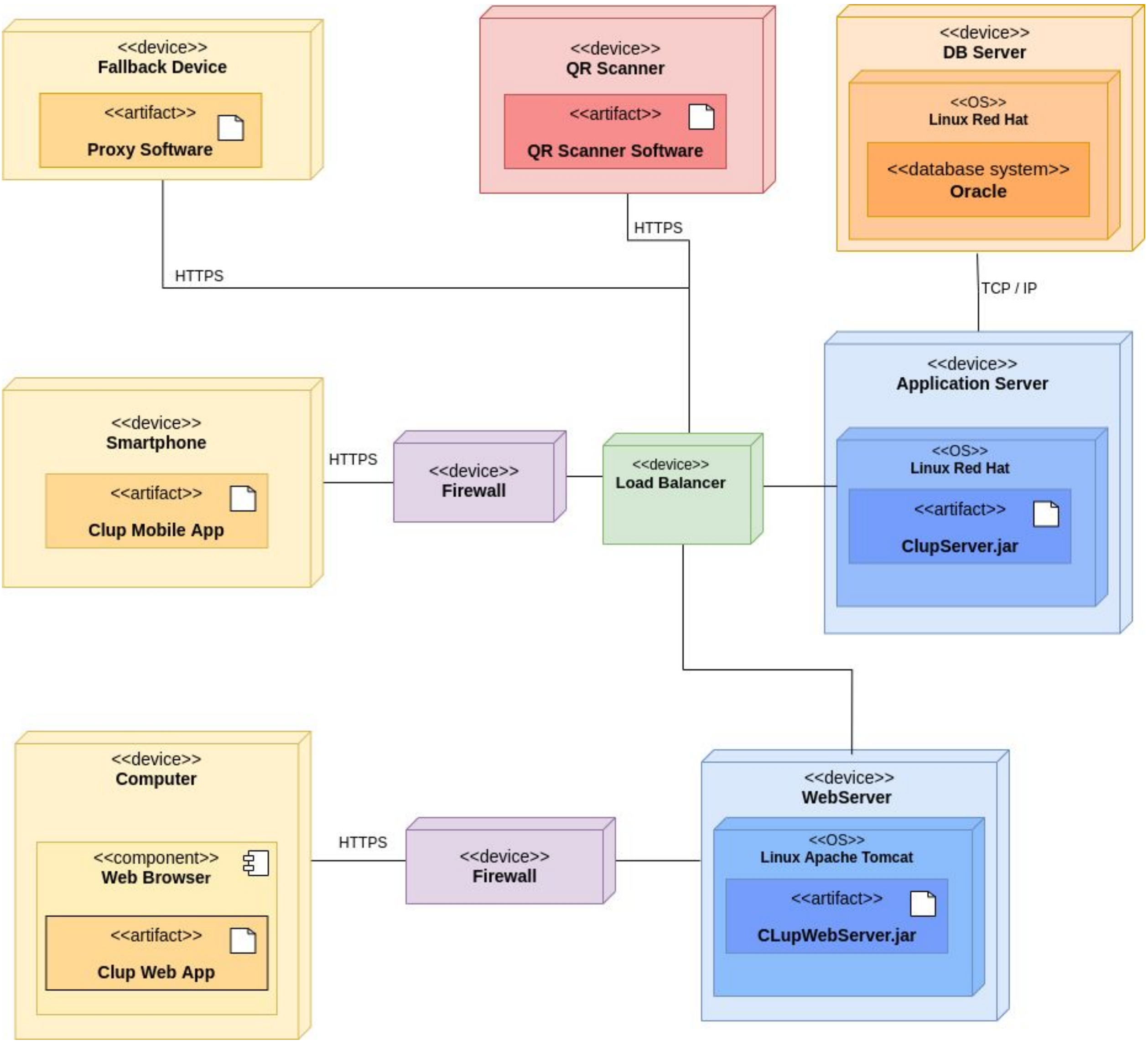
- **QR Scanner:** This software component is installed into the QR Scanner machines. Its role is to check whether the provided ticket is valid or not for entering or exiting the store. It also has to send the entry and exit time to the server. The only component it interacts with is the Data Manager that provides directly the required tickets' data.

- **Fallback Device:** This component represents the software running on the device that provides tickets for the fallback option. Once a user requests a ticket it accesses the Line-Up Service to release tickets.

- **Statistics calculator:** This software component computes statistics regarding the time spent at the store by users and about the average number of people who enter the store every day. Statistics are then stored to the database and used either to infer the amount of time users are going to spend for shopping or to allow the store manager to visualize weekly statistics.

- **Database**
  This component represents the DBMS, which provides the interfaces to retrieve and store the data. Data about the application, supermarkets and users are securely stored and encrypted.

2.3. Deployment View



In the Deployment Diagram above are shown the most principal components of the system. In particular, all the business layer's components are in the application server while the data layer is located in the DB Server. More in detail:

DB Server: The DB server is one of the most important components of the system, as it contains a relational DBMS, that can be interrogated with SQL queries. This component stores all the information of the CLup software such as users' data , tickets requested by users, supermarkets' and store managers' information, virtual queue status etc. The DB Server is also equipped with a relational DBMS and can be interrogated with SQL queries.

Application Server: This component incorporates the business logic of the application. Its role is to receive requests and send back responses. The mobile app is directly connected to it whereas the Web app can only make requests to the WebServer that will be forwarded to the Application Server in order to be processed. The application server retrieves data by communicating with the DB Server through a TCP connection.

WebServer: This server component has to respond to webapp requests. In particular it directly communicates with the web application and then eventually asks the Application server to process data received from the webapp.

Load Balancer: This device has the role of easing the computational load between multiple application servers. Whenever a user's request arrives here, this component decides which application server should process it in order to avoid overloading and to keep computation balanced. Load balancer is not used for the Web Server since the number of store managers should be considerably lower with respect to the number of users.

Smartphone: This is the device used by users to get tickets for lining up or for booking a visit at the supermarket through the CLup mobile app. It gets connected to the system through a HTTPS connection.

Computer: Device used by store managers to register themselves and their store into the system and used to visualize statistics and data. Since store managers access the system by using a webapp, it is connected to the Web Server through a HTTPS connection.

Firewall: This component has the role to filter requests from not secure sources. It is part of the security of the system.

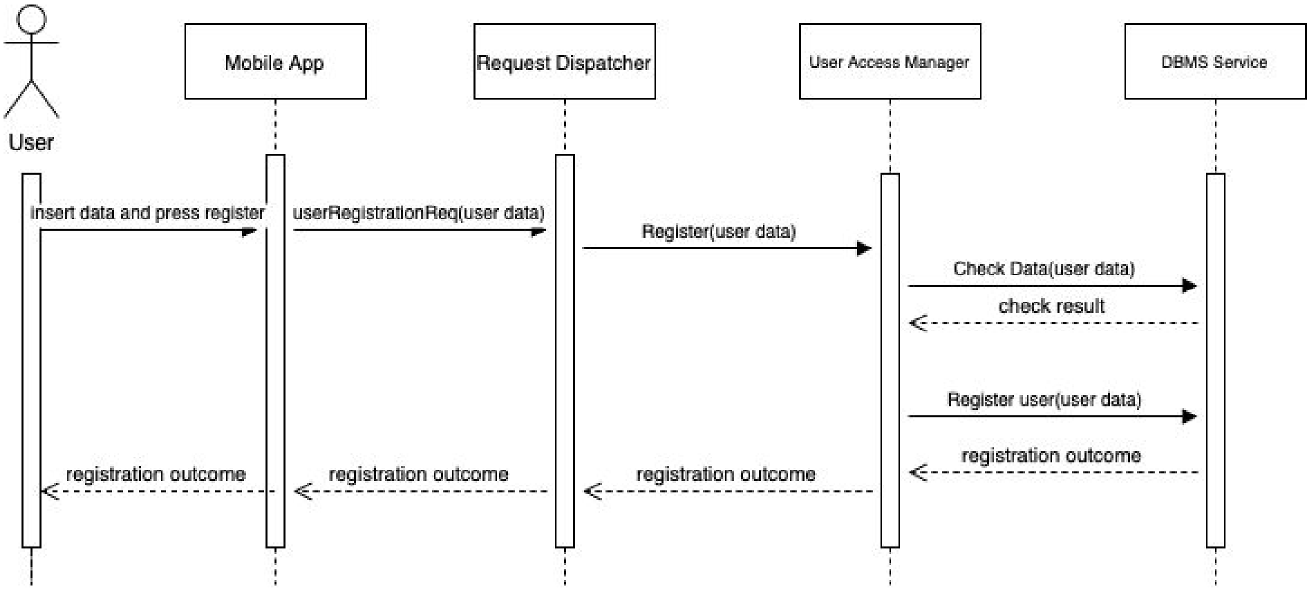
Fallback Device: This device, used to provide the fallback option, is connected to the system through a HTTPS connection. In particular, it requests the generation of a ticket for lining-up to the application server and receives the ticket and QR code from it.

QR Scanner: This component, used to check whether users provide valid tickets for entering or exiting the store, needs a HTTPS connection to the system. In fact, it should send tickets' data to the system to check their validity and receive a response that says whether the user is allowed or not to enter/exit the store. Furthermore, during the validation request, the entry or exit time is also sent to the application server in order to be possibly stored into the database in case of a positive check of the ticket.

2.4. Runtime View

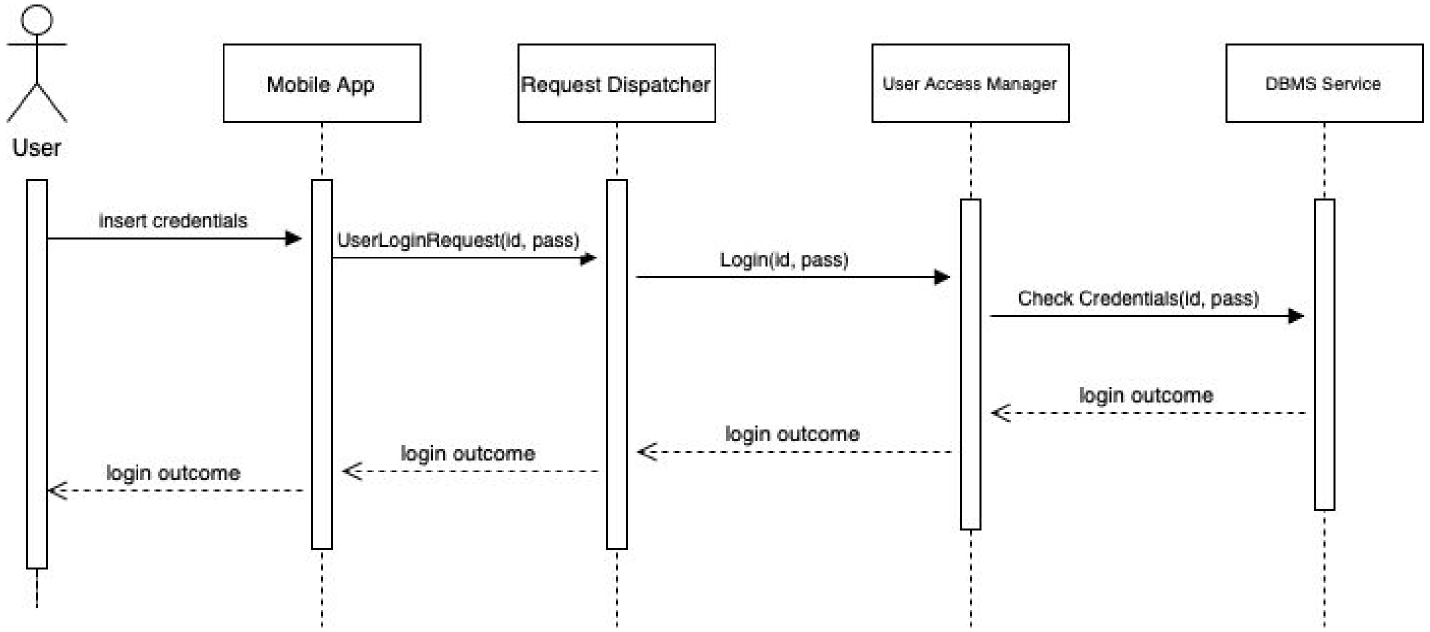
Here we use sequence diagrams to describe the way components interact to accomplish the main tasks of the CLup software.

- **User Registration**



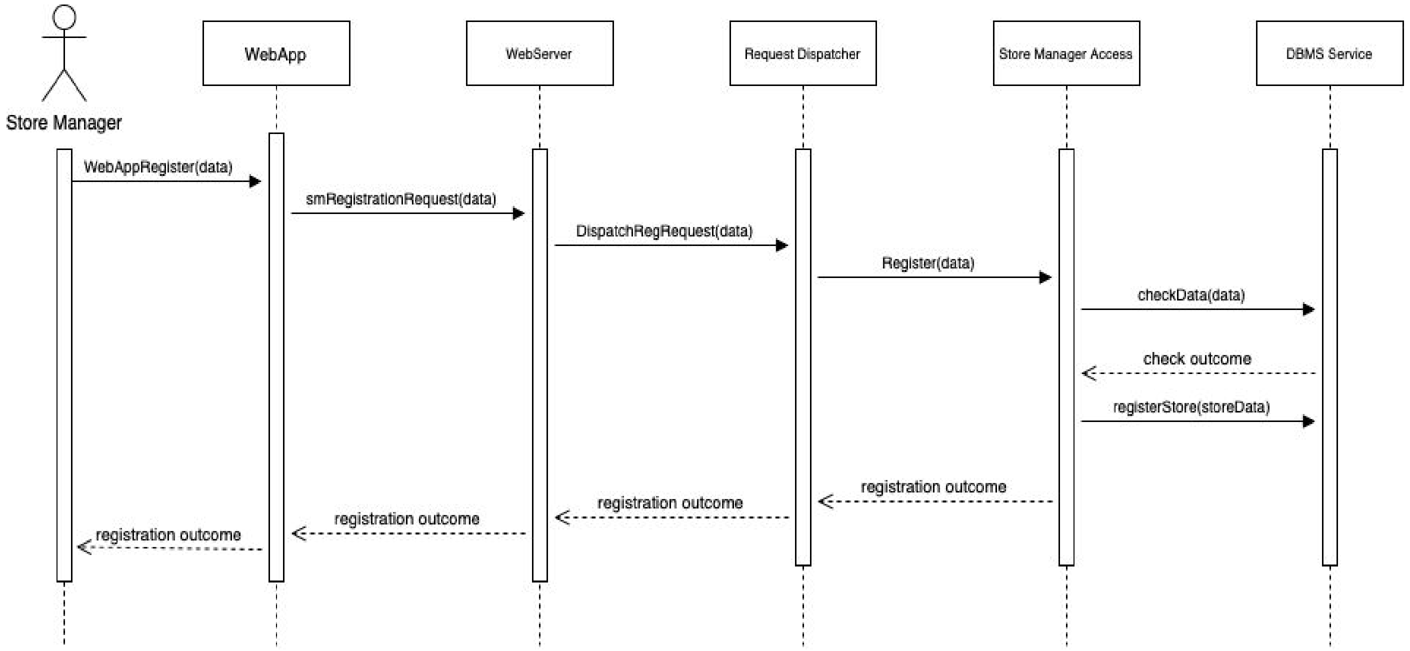
In the sequence diagram above, the user's registration is shown. Once the user has completed the registration form that requires his personal data, the mobile app sends a registration request for the user. Then the request dispatcher forwards the request to the User Access Manager by calling the register method of the provided interface. Then the access manager checks if registration data are valid (for instance checks whether the inserted email is already associated to another account or not), and eventually confirms the registration.

- **User Login**



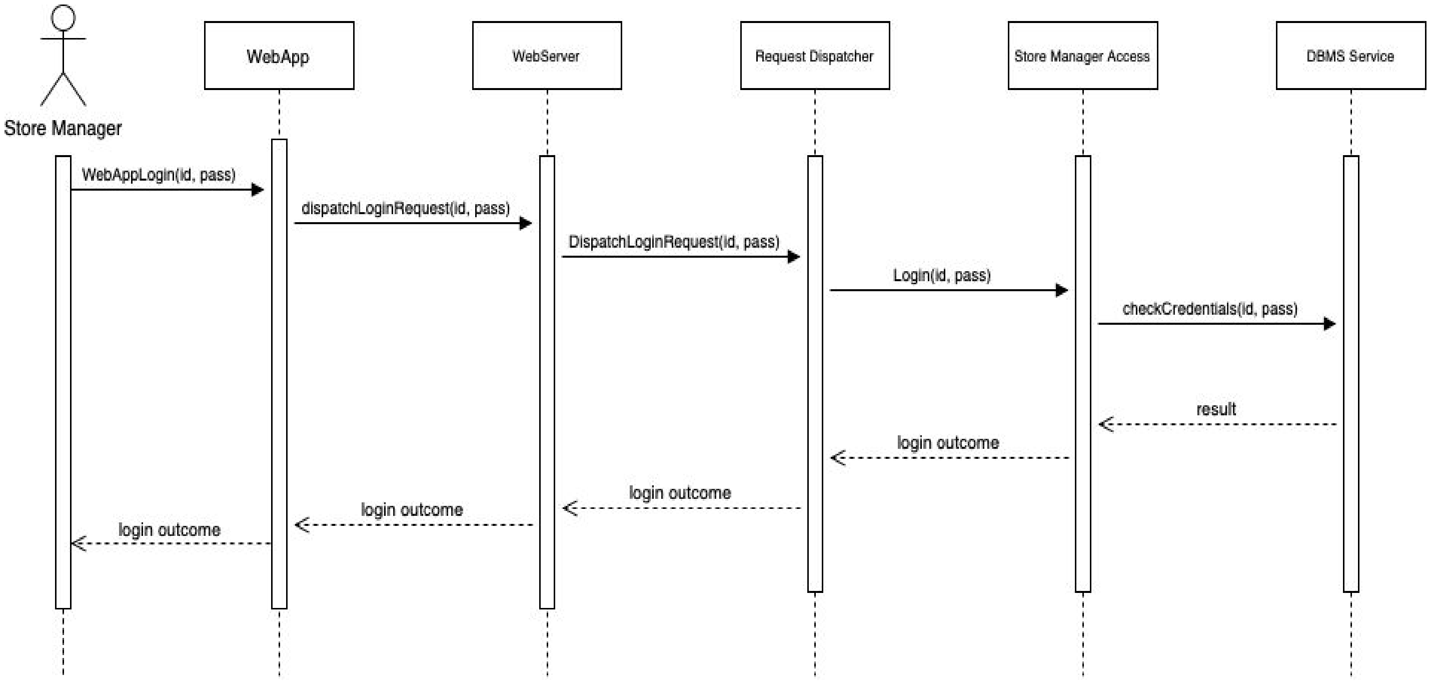
The user login sequence diagram is similar to the registration one. Firstly, the user inserts his username and password, then the mobile app sends a login request to the system. Afterwards, the user access manager checks the user's credentials by querying the database through the DBMS. Then the outcome of the identification is shown to the user.

- **Store Manager Registration**



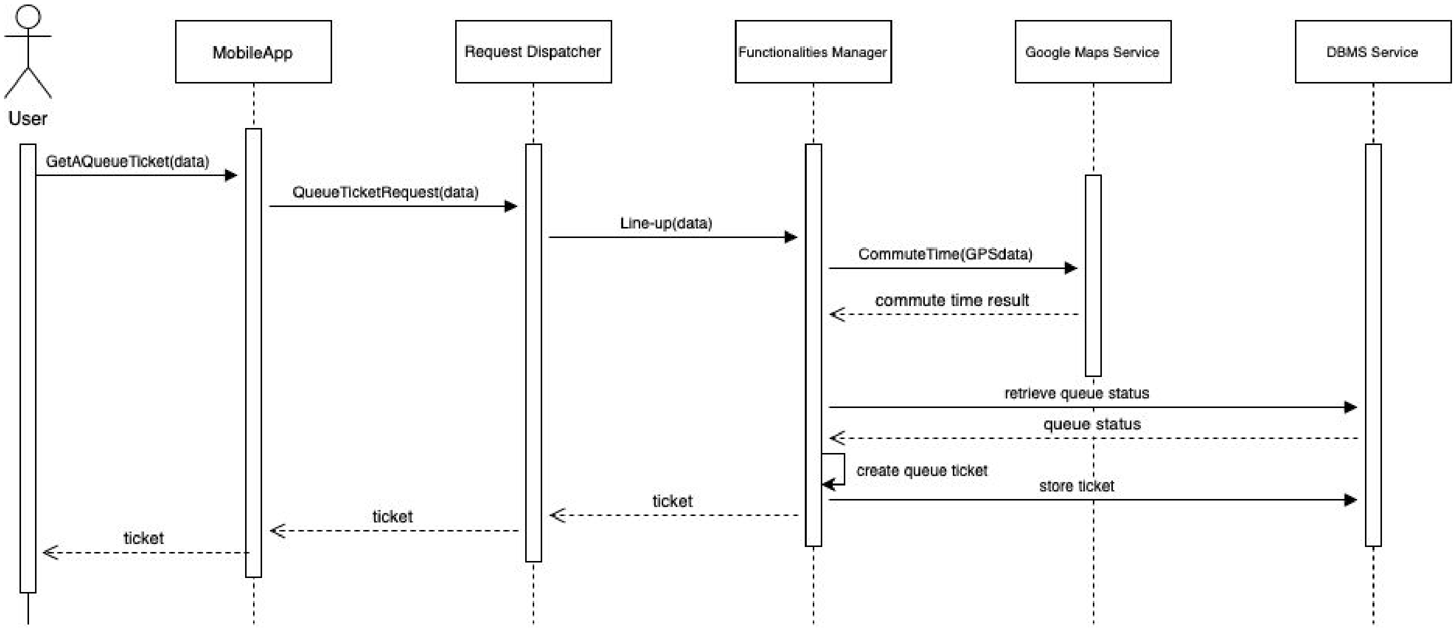
The store manager registration is similar to the user registration process but this time interaction is through a webapp and requests are firstly handled by a Web Server. Since this user is registering himself and the store he works for, he inserts his personal data, store data (such as departments, capacity, products, etc…) and the certificate that proves he is an employee of the store. Then these data are checked and verified and in case of positive evaluation, the employee is registered as store manager and the store will be accessible by the users of the app.

- **Store Manager Login**



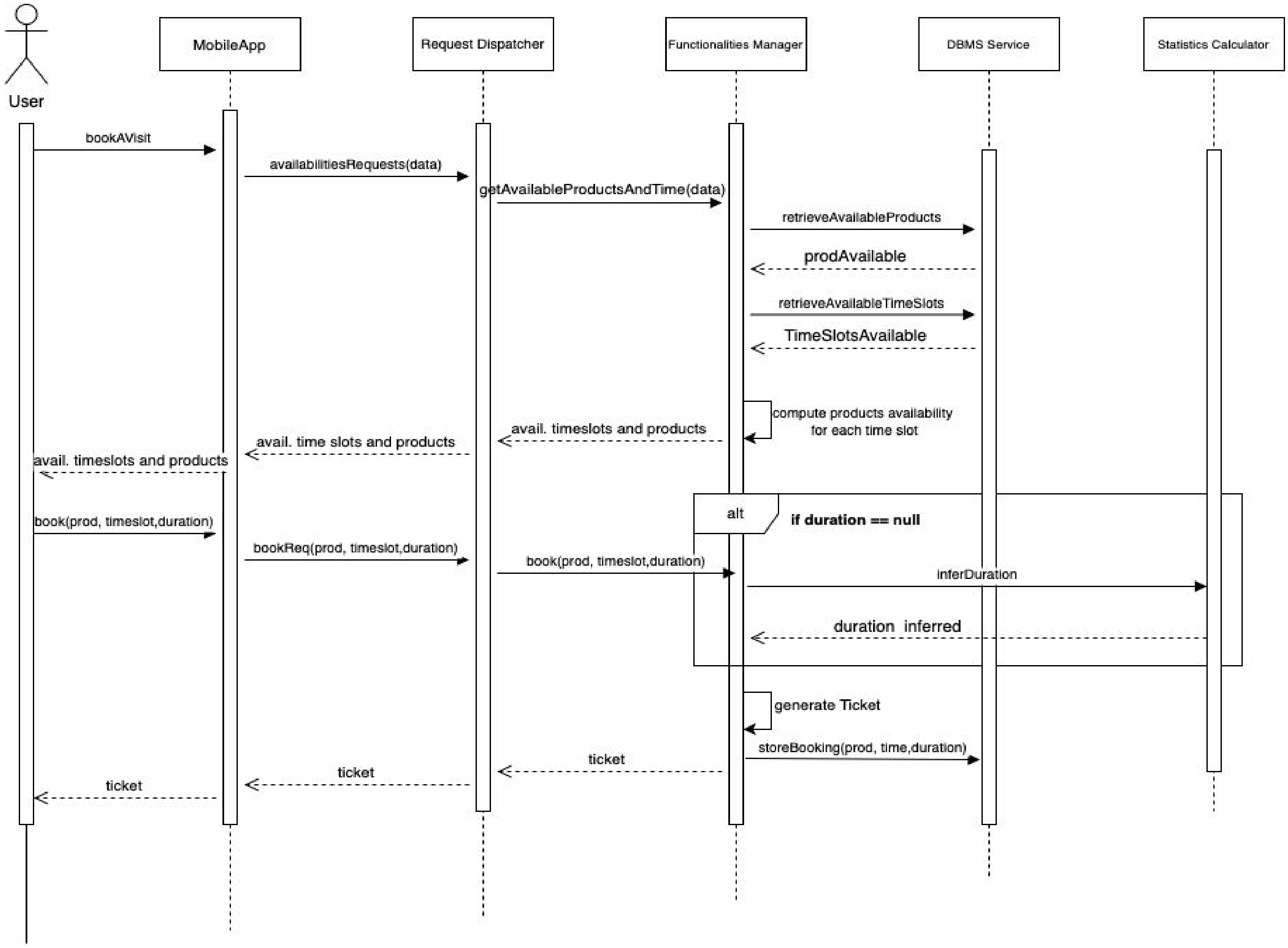
In this diagram the store manager connects to the login page of the webapp and after having inserted username and password, clicks on the login button. Then the webapp will interact with the web server that in turn will forward the request to the Store Manager Access component. This component, after having checked the validity of credentials, sends back the response to the store manager.

- **Get a Queue Ticket**



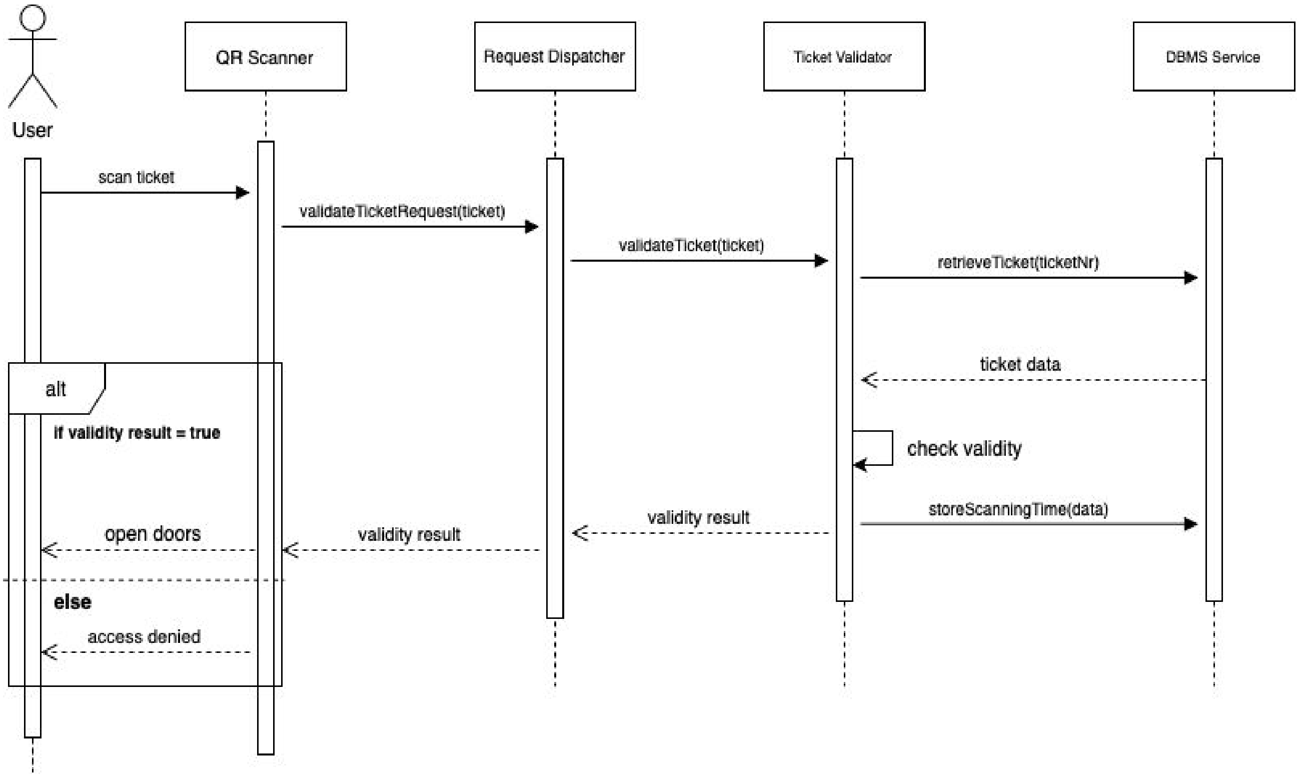
In the sequence diagram above is shown one of the two main functionalities of the CLup software. After having logged in and selected the store, the user inserts the means of transport he will use to reach the store and finally clicks on the “Line-Up” button, as shown in the mockups. Then the login request is sent to the request dispatcher and then forwarded to the Functionalities Manager, by calling a method of its interface. This component calls a method of the Google Maps Service API in order to get the time needed by the user to reach the store, given his GPS data. Then gets the queue status from the database in order to generate the new ticket number and its waiting time. When the ticket is generated, it is stored and sent to the user with the related QR code included.

- **Book a Visit**


User
MobileApp
Request Dispatcher
Functionalities Manager
DBMS Service
Statistics Calculator
bookAVisit
availabilitiesRequests(data)
getAvailableProductsAndTime(data)
retrieveAvailableProducts
prodAvailable
retrieveAvailableTimeSlots
TimeSlotsAvailable
compute products availability for each time slot
avail. timeslots and products
avail. time slots and products
avail. timeslots and products
alt
if duration == null
book(prod, timeslot,duration)
bookReq(prod, timeslot,duration)
book(prod, timeslot,duration)
inferDuration
duration inferred
generate Ticket
storeBooking(prod, time,duration)
ticket
ticket
ticket

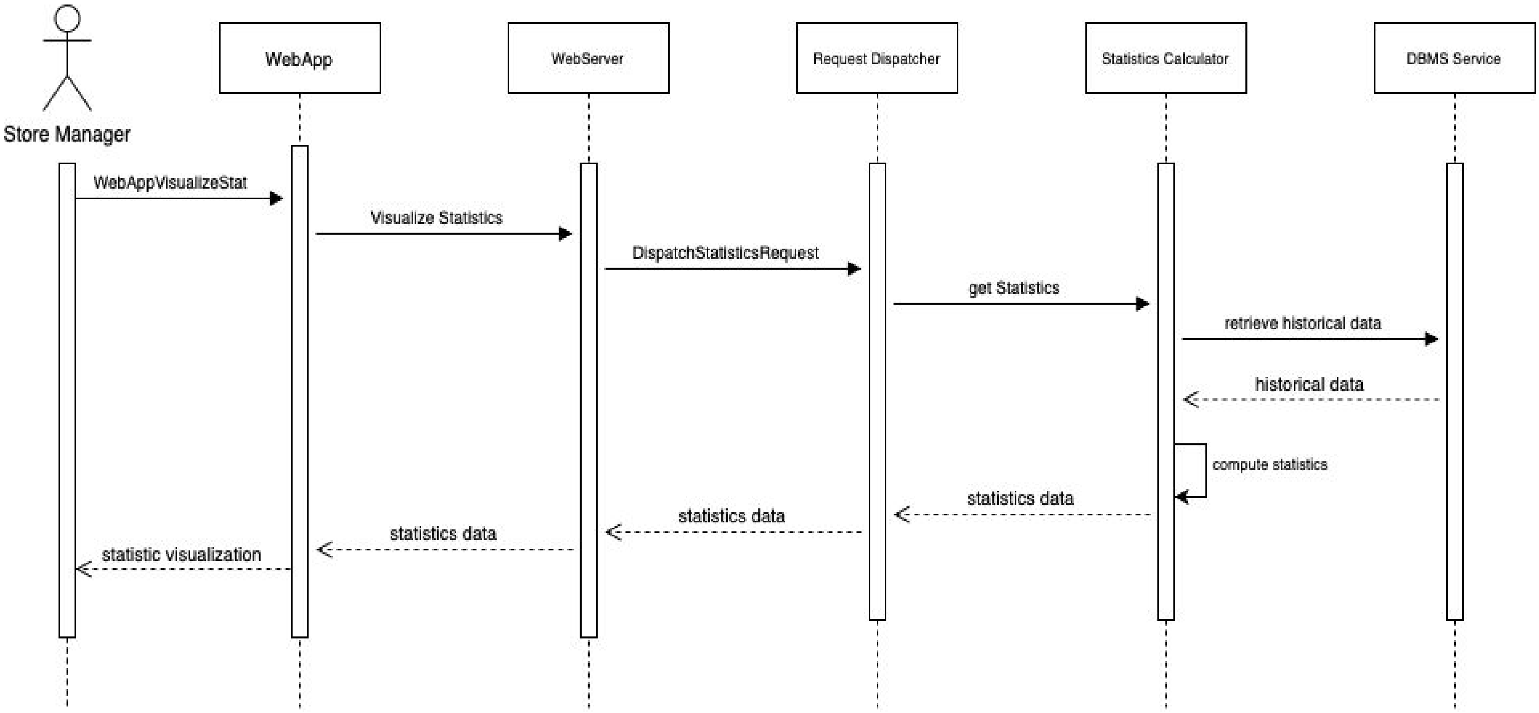
This sequence diagram shows the second main functionality of the CLup software. After having logged in and selected the store where to do shopping, the user selects the “Book a Visit” button. The mobile app makes a request to get the availability of time slots and products’ categories. This request is forwarded from the Request Dispatcher to the Functionalities Manager that in turn queries the database to get data about capacities of departments for each timeslot. Then these data are sent back to the mobile app that will show the available time slots for shopping, and for each timeslot the products that can be bought, considering their department’ s available capacity. Then the user selects the timeslot and the products’ categories he is interested in and may specify the duration of the visit. Then confirms the booking. When the system receives the request, if the user did not insert the duration, it is inferred by a method call of the Statistic Calculator component interface. Afterwards, the ticket is generated, stored into the database, and sent to the user with its QR code.

- **Ticket Validation**



In the sequence diagram above is shown the validation process of the ticket provided by the user at the entrance of the store. Once the customer has scanned the QR code, the QR scanner's software sends a validation request to the Request Dispatcher. Then it forwards the request to the Ticket Validator by calling a method provided by the validator's interface . This component checks through the DBMS service whether the provided ticket is valid or not for entering or exiting the store. Then the validity result is sent to the QR Scanner and if the ticket is valid the system opens the doors, otherwise denies the access to the customer.

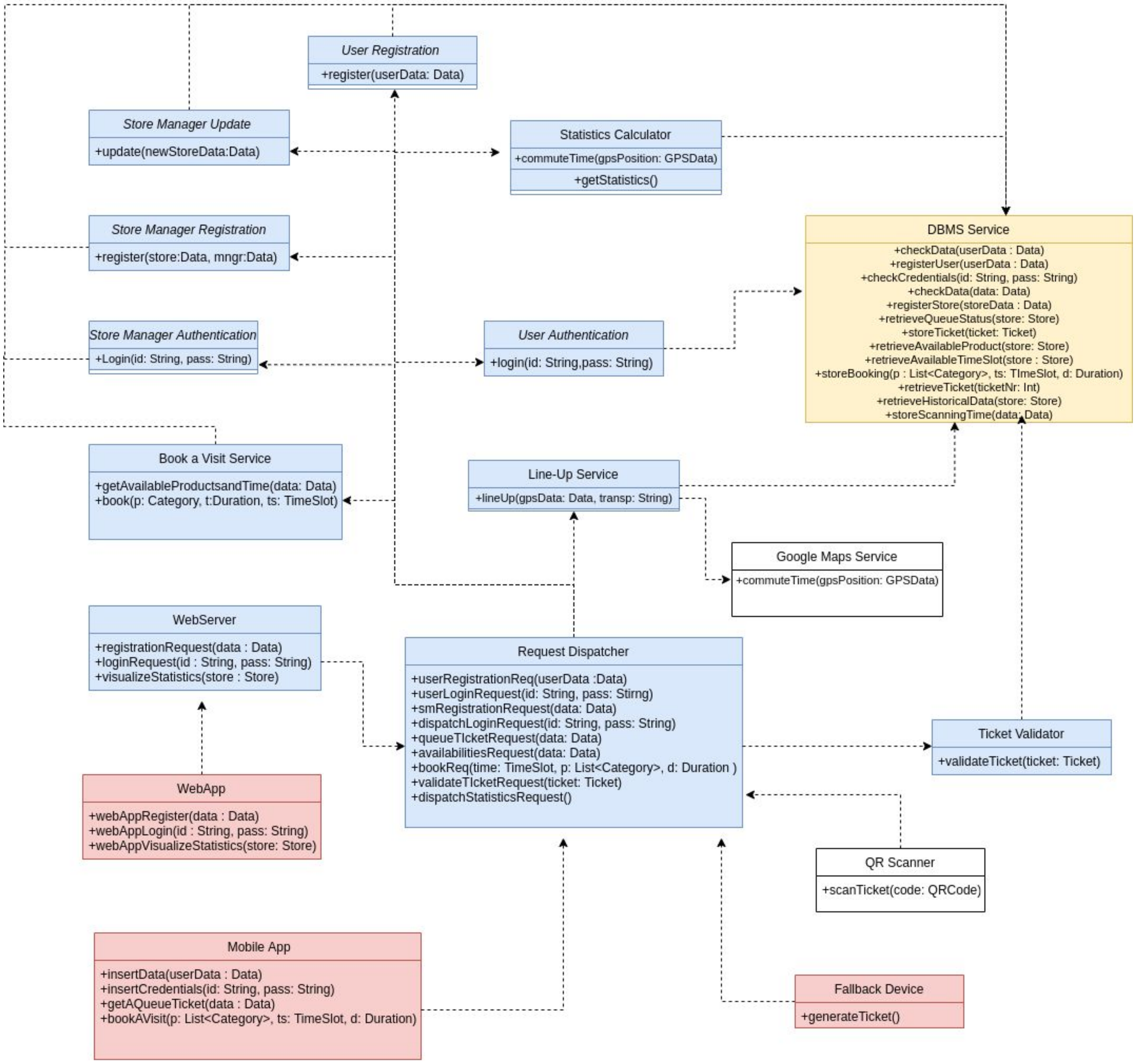
- **Statistics Visualization**



The sequence diagram above illustrates the statistics visualization process. Once the store manager has logged into the web app, selects the statistics visualization option. Then the web app makes a request to the web server and it is forwarded to the Statistics Calculator by the Request Dispatcher. The Statistics Calculator retrieves historical data from the database and uses them to compute statistical data that are subsequently sent back to the store manager. Then the browser shows and plots the data received.

2.5. Component Interfaces

In this section are described the main methods of the main components' interfaces.


User Registration
+register(userData: Data)
Store Manager Update
+update(newStoreData:Data)
Statistics Calculator
+commuteTime(gpsPosition: GPSData)
+getStatistics()
Store Manager Registration
+register(store:Data, mngr:Data)
DBMS Service
+checkData(userData : Data)
+registerUser(userData : Data)
+checkCredentials(id: String, pass: String)
+checkData(data: Data)
+registerStore(storeData : Data)
+retrieveQueueStatus(store: Store)
+storeTicket(ticket: Ticket)
+retrieveAvailableProduct(store: Store)
+retrieveAvailableTimeSlot(store : Store)
+storeBooking(p : List<Category>, ts: TImeSlot, d: Duration)
+retrieveTicket(ticketNr: Int)
+retrieveHistoricalData(store: Store)
+storeScanningTime(data: Data)
Store Manager Authentication
+Login(id: String, pass: String)
User Authentication
+login(id: String,pass: String)
Book a Visit Service
+getAvailableProductsandTime(data: Data)
+book(p: Category, t:Duration, ts: TimeSlot)
Line-Up Service
+lineUp(gpsData: Data, transp: String)
Google Maps Service
+commuteTime(gpsPosition: GPSData)
WebServer
+registrationRequest(data : Data)
+loginRequest(id : String, pass: String)
+visualizeStatistics(store : Store)
Request Dispatcher
+userRegistrationReq(userData :Data)
+userLoginRequest(id: String, pass: Stirng)
+smRegistrationRequest(data: Data)
+dispatchLoginRequest(id: String, pass: String)
+queueTIcketRequest(data: Data)
+availabilitiesRequest(data: Data)
+bookReq(time: TimeSlot, p: List<Category>, d: Duration)
+validateTIcketRequest(ticket: Ticket)
+dispatchStatisticsRequest()
Ticket Validator
+validateTicket(ticket: Ticket)
WebApp
+webAppRegister(data : Data)
+webAppLogin(id : String, pass: String)
+webAppVisualizeStatistics(store: Store)
QR Scanner
+scanTicket(code: QRCode)
Mobile App
+insertData(userData : Data)
+insertCredentials(id: String, pass: String)
+getAQueueTicket(data : Data)
+bookAVisit(p: List<Category>, ts: TimeSlot, d: Duration)
Fallback Device
+generateTicket()

2.6. Selected architectural styles and patterns

2.6.1. Model View Controller

The distributed Model View Controller is the main architectural pattern upon which the application is built.

The MVC pattern divides the application into three parts:

- Model: holds the data and the most fundamental components of the application and communicates exclusively with the controller
- View: represents the user-facing part of the application
- Controller: receives user events and user inputs, performs the necessary computations and interacts with the model. The controller receives the input, optionally validates it and then passes the input to the model

It was chosen to decouple the Data, Business Logic and Presentation layers as we discussed in section 2.1. This enables a faster development cycle, since developers can work on separate parts of the application, in addition to facilitating the debugging and update processes.

2.6.2. Facade

The facade pattern has been used in this application to abstract away the complexity of the application, in fact there is only one component that interacts with the client as a "facade", the Request Dispatcher. This component shows an interface that hides all the complexity behind the business logic and data layers to the presentation layer.

2.6.3. Thin Client

The thin client paradigm is implemented for the interaction between the frontend and the backend. A thin client needs to perform very few computations, but it has to handle the communication and display data to the user.

It’s convenient as it requires less effort to create new client implementation because of the lack of logic. A thin client does not rely on local storage since all the business logic is on the application servers, which have enough computing power.

This applies to both the webapp and the mobile app.

2.6.4. REST Architecture

We chose to implement the REST architecture mainly for the sake of scalability and flexibility. Data is not tied to resources or methods, so REST can handle multiple types of calls, return different data formats and even change structurally with the correct implementation of hypermedia. This flexibility allows developers to build an API that gives much more independence to the development of different components belonging to the presentation layer, such as the mobile app, the web app, the QR Scanner and the Fallback Device.

Data exchange is also made through the HTTPS Protocol using SSL to provide encryption of sensible data. Using HTTPS and complying with REST guidelines allows us to provide clear, complete and secure communication with the Clients.

2.7. Other design decisions

2.7.1. Database

The application uses a relational database, which is the most appropriate choice given that it deals with structured data. Even though the computations of the statistics might evolve big amounts of data, the vast majority of the operations only involve small amounts of data. These are repetitive operations that are computed very fast by a relational database, therefore the cons of using a non-relation model would far outweigh the pros. Also, a relational database can be queried using SQL, which is a standard across the industry.

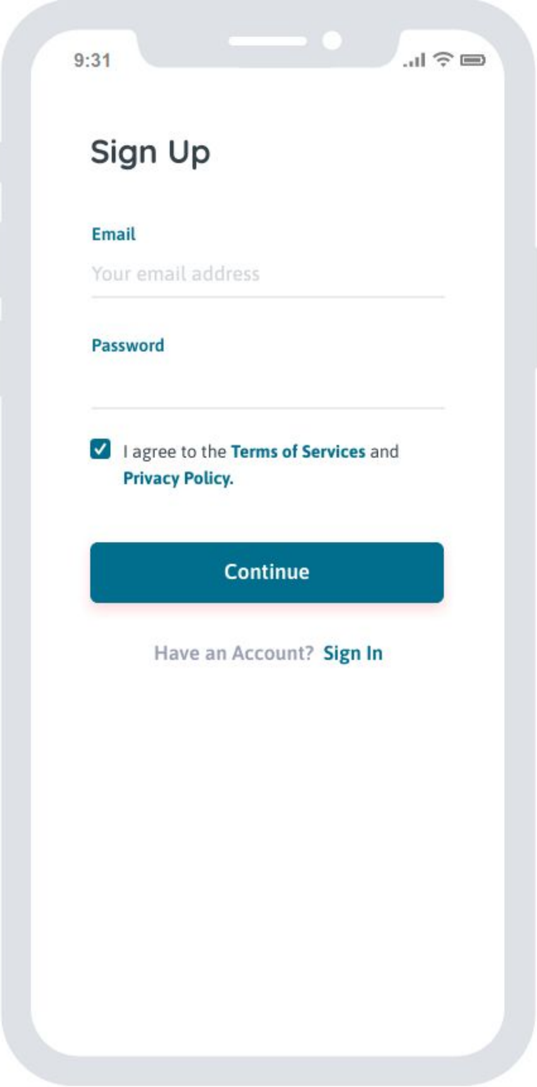
2.7.2. Firewall

There are two firewalls that shield the application server from the outside security threads. One between the web server and the web application view, and another between the Load Balancer and the mobile application view.

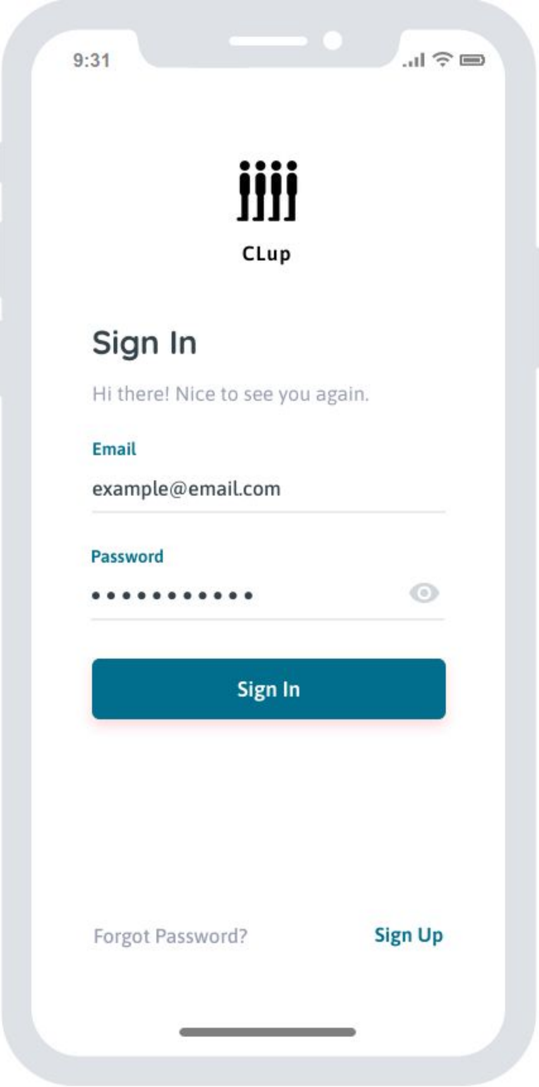
3. User Interface Design

Here we provide the mockups for the mobile app application, which will be used by users, and the web app interface for store managers.

3.1. Sign up & Sign in



☑ I agree to the **Terms of Services** and **Privacy Policy.**

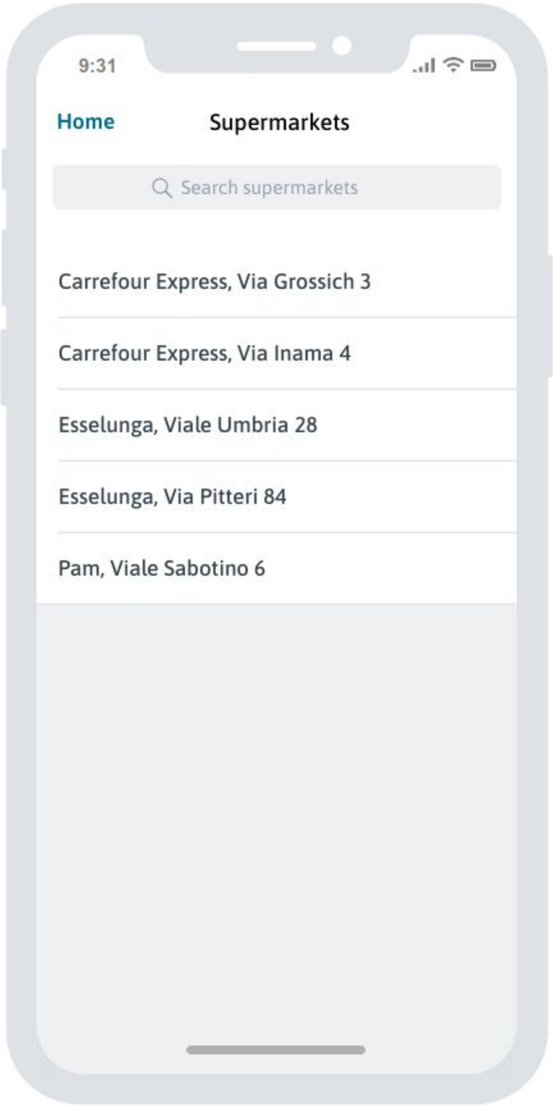


Hi there! Nice to see you again.

Email: example@email.com

Password: ●●●●●●●●●●●●

3.2. Home & Supermarket



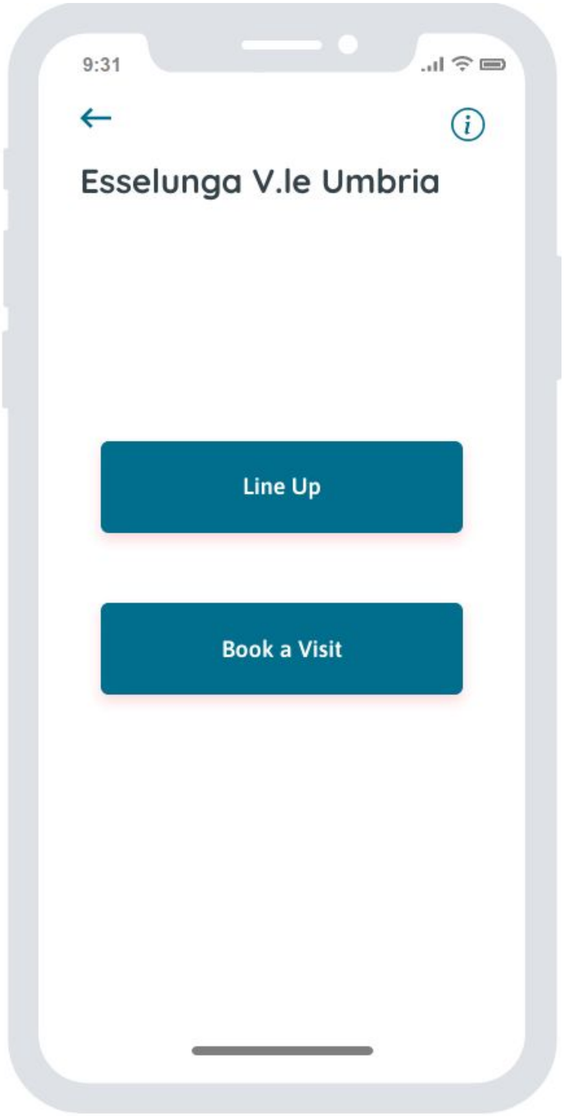
Carrefour Express, Via Grossich 3

Carrefour Express, Via Inama 4

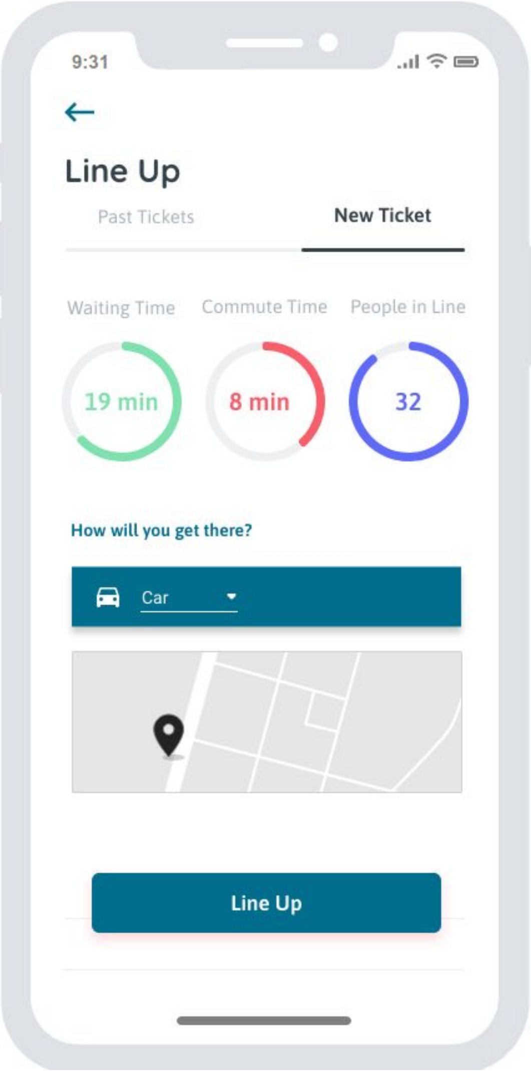
Esselunga, Viale Umbria 28

Esselunga, Via Pitteri 84

Pam, Viale Sabotino 6



3.3. Line-Up & Book a Visit

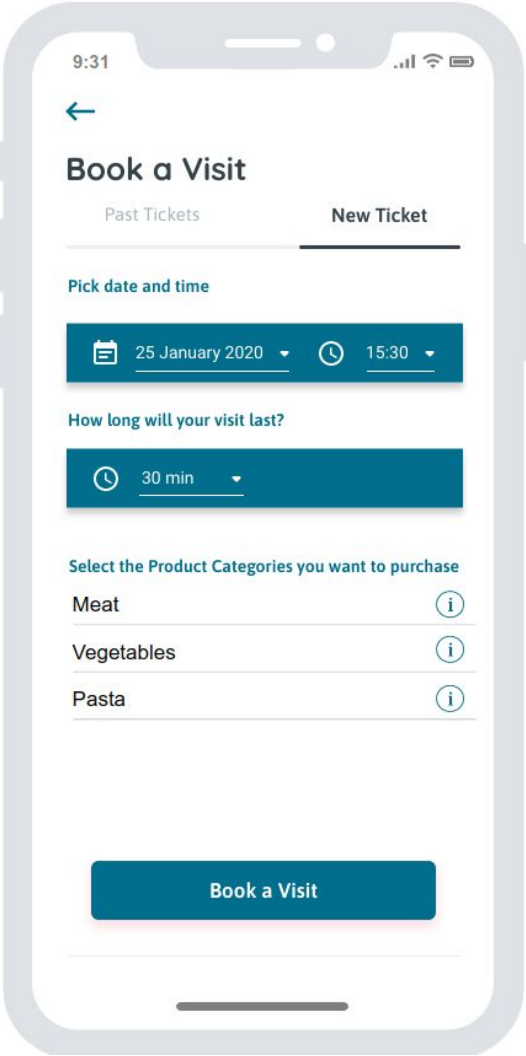


Waiting Time: 19 min

Commute Time: 8 min

People in Line: 32

How will you get there? Car



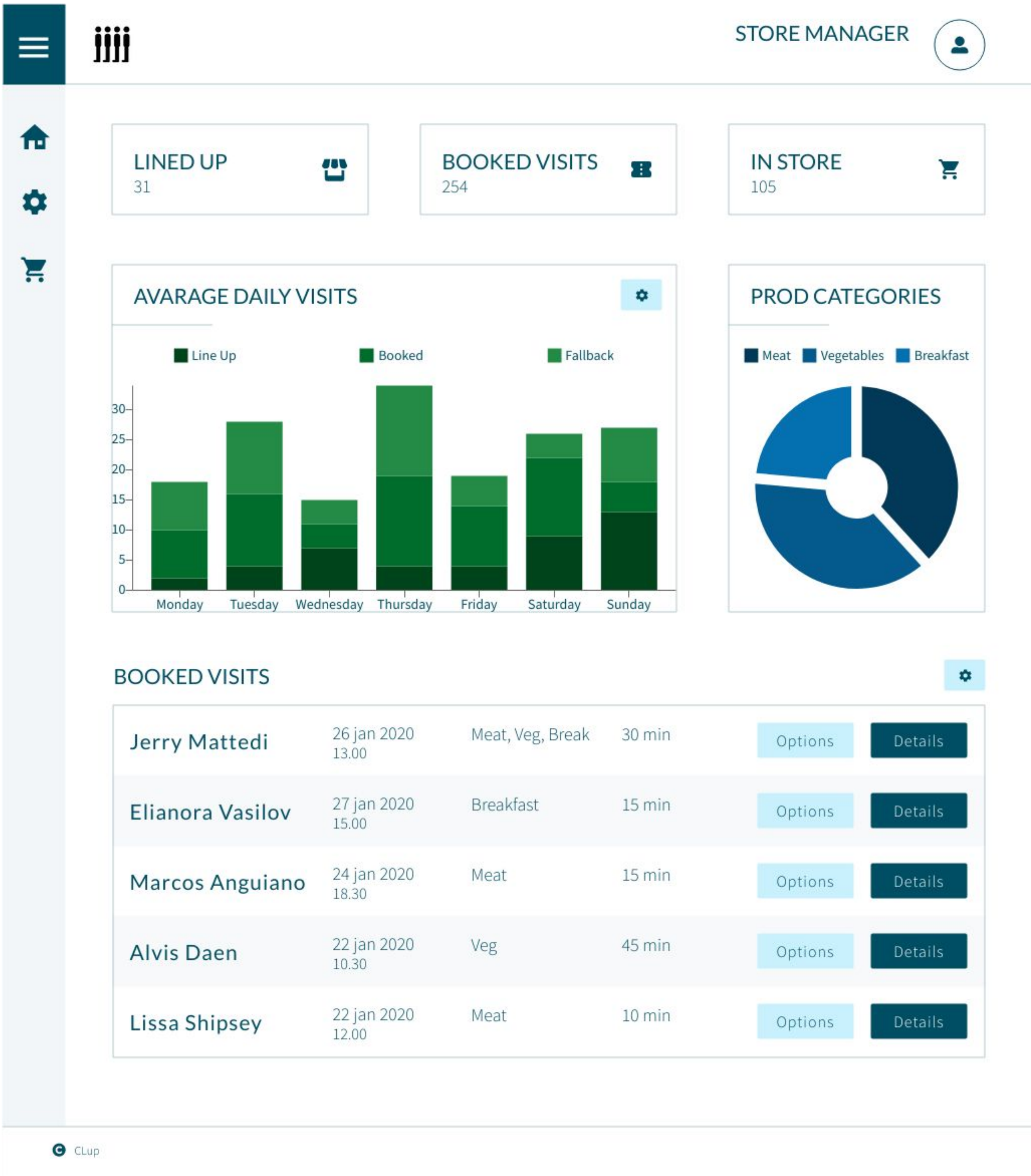
Pick date and time: 25 January 2020, 15:30

How long will your visit last? 30 min

Select the Product Categories you want to purchase

- Meat
- Vegetables
- Pasta

3.4. Store dashboard



STORE MANAGER

LINED UP
31

BOOKED VISITS
254

IN STORE
105

AVARAGE DAILY VISITS

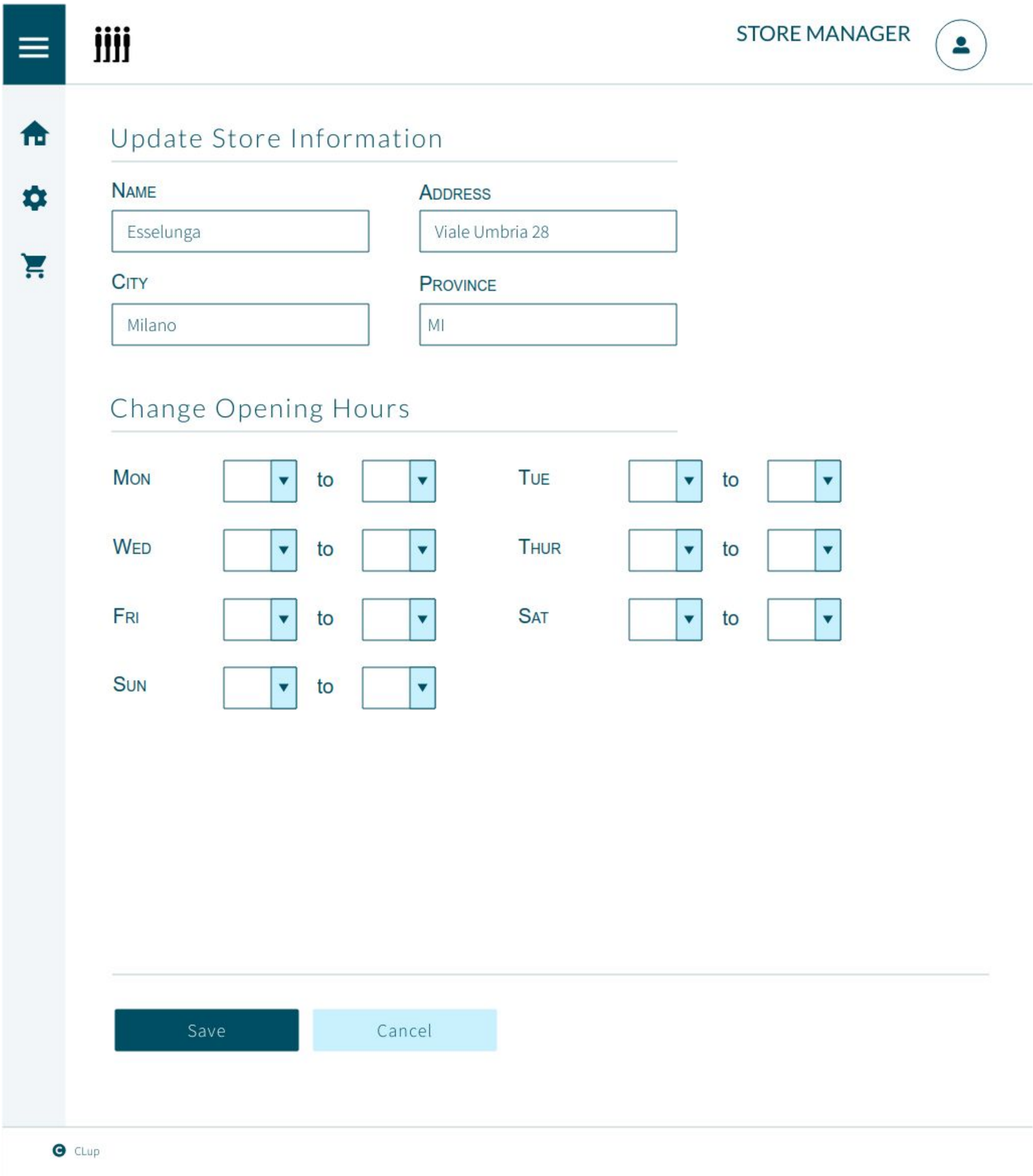
PROD CATEGORIES

BOOKED VISITS

Jerry Mattedi	26 jan 2020 13.00	Meat, Veg, Break	30 min	Options	Details
Elianora Vasilov	27 jan 2020 15.00	Breakfast	15 min	Options	Details
Marcos Anguiano	24 jan 2020 18.30	Meat	15 min	Options	Details
Alvis Daen	22 jan 2020 10.30	Veg	45 min	Options	Details
Lissa Shipsey	22 jan 2020 12.00	Meat	10 min	Options	Details

© CLup

3.5. Store Settings



Update Store Information

NAME: Esselunga

ADDRESS: Viale Umbria 28

CITY: Milano

PROVINCE: MI

Change Opening Hours

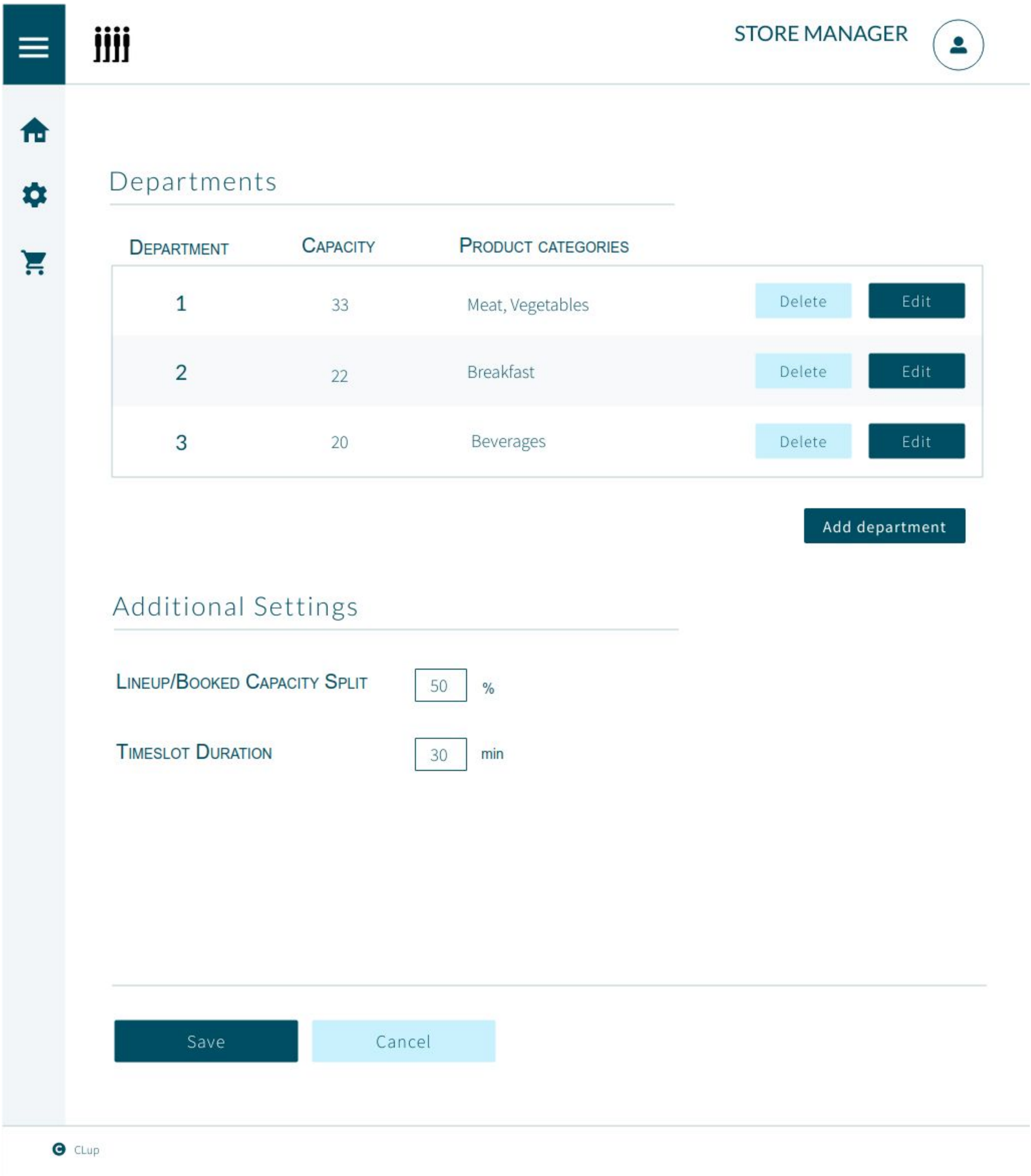
MON ▾ to ▾ TUE ▾ to ▾

WED ▾ to ▾ THUR ▾ to ▾

FRI ▾ to ▾ SAT ▾ to ▾

SUN ▾ to ▾

3.6. Department Settings



Departments

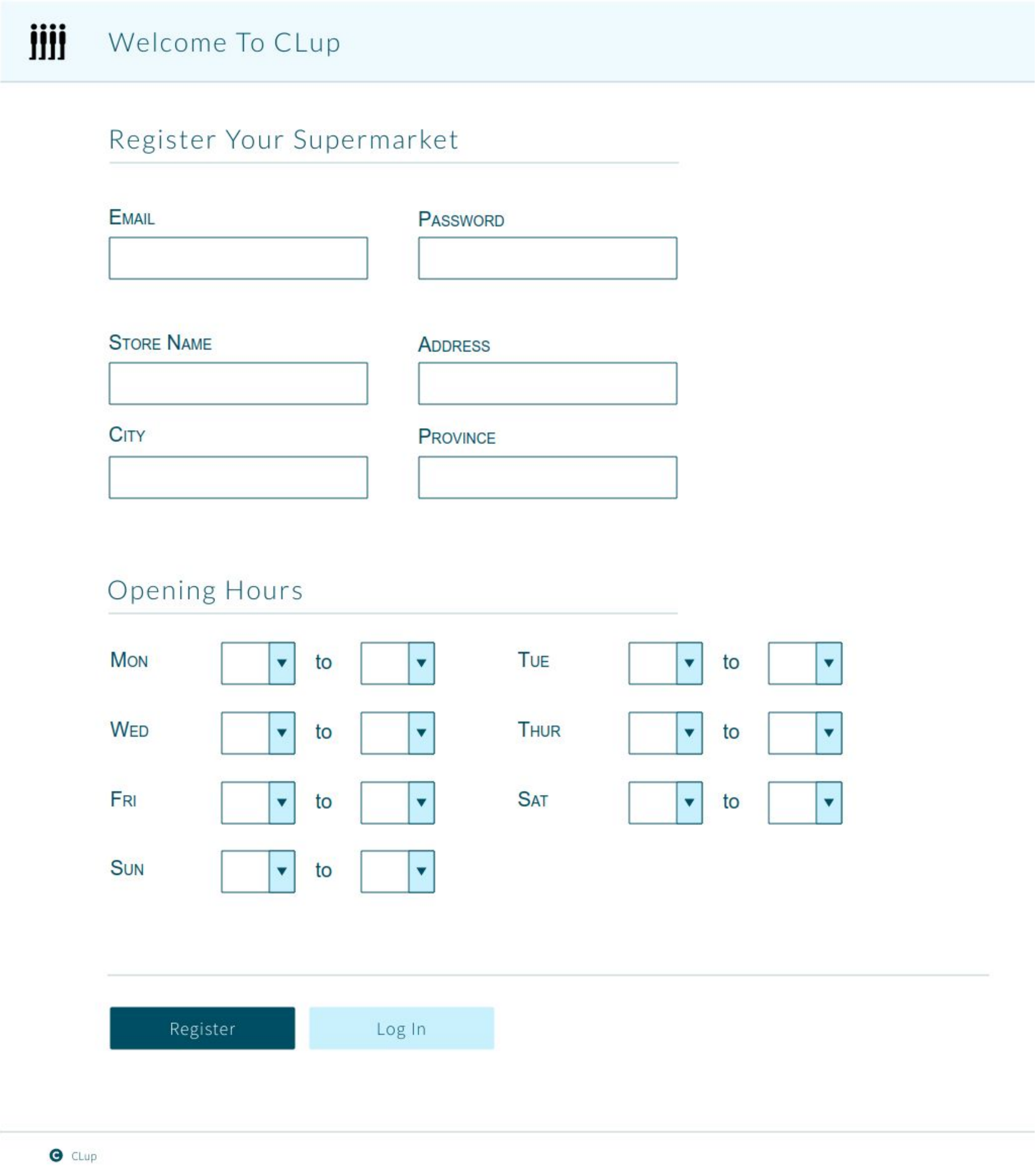
Department	Capacity	Product categories
1	33	Meat, Vegetables
2	22	Breakfast
3	20	Beverages

Additional Settings

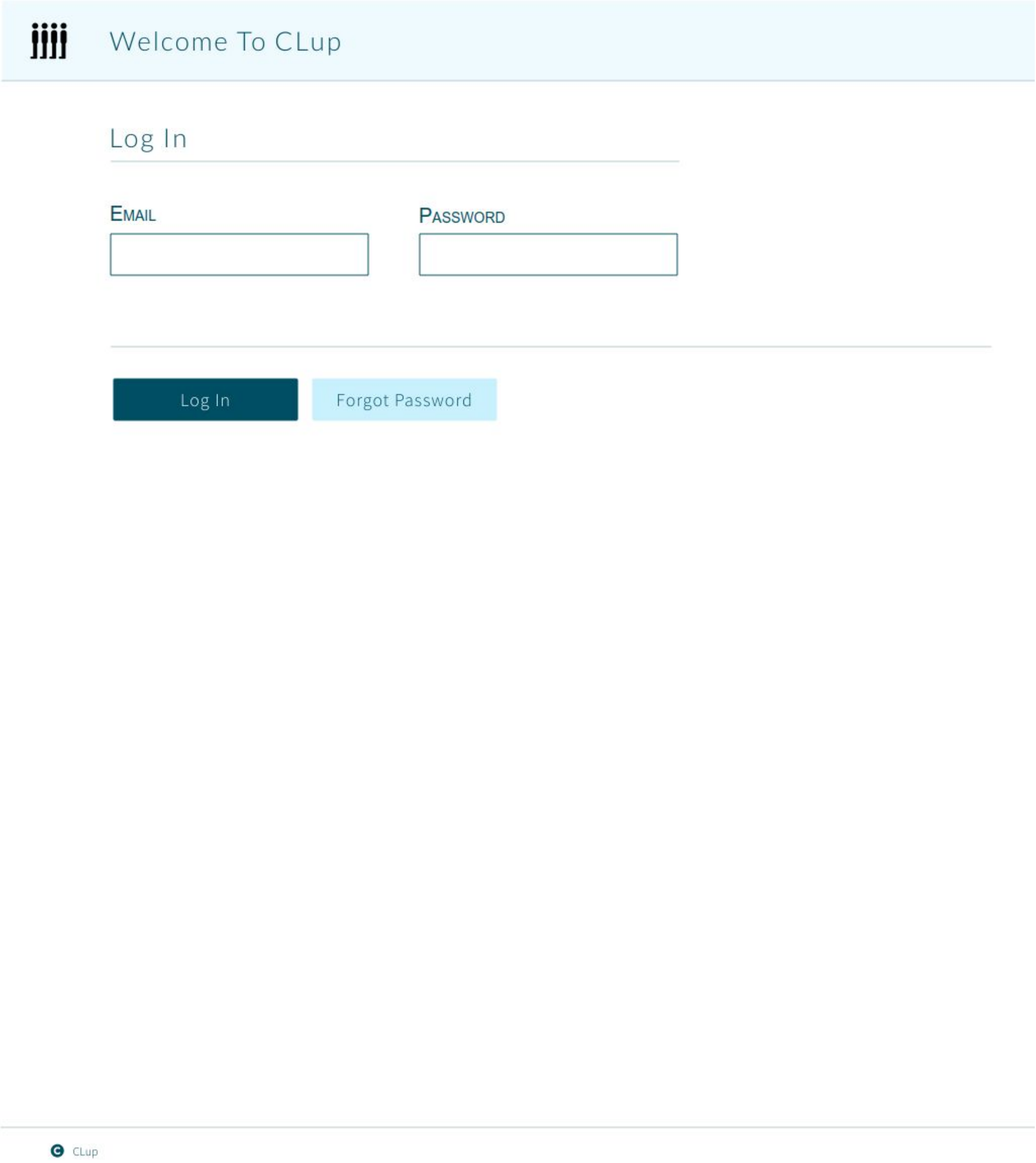
Lineup/Booked Capacity Split 50 %

Timeslot Duration 30 min

3.7. Store Register



3.8. Store Log In



4. Requirements Traceability

This section maps the goals and requirements defined in the RASD to the design components defined in section 2.2

G1	Allow store managers to regulate the influx of people in the building
	Requirements: R1, R2, R4, R5, R7, R8, R10, R12, R13, R15, R16, R17, R18
	Statistics Calculator, Web App, Web Server, DBMS Service, SM Access Manager
G2	Allow people to virtually line up instead of standing outside of stores
	Requirements: R1, R2, R4, R6, R7, R9, R14, R15, R16, R17, R18
	Line Up Service, Mobile App, Google Maps Service, DBMS Service, Ticket Validator, Fallback Device, QR Scanner, User Access Manager
G3	Allow people to plan a visit to the supermarket
	Requirements: R1, R2, R4, R6, R10, R11, R12, R13, R15, R16
	Book A Visit Service, Mobile App, DBMS Service, Ticket Validator, QR Scanner, Statistics Calculator, User Access Manager
G4	Allow to keep safety distance among people inside the store
	Requirements: R1, R2, R4, R5, R8,R10, R12, R13, R15, R16, R18
	Line Up Service, Book A Visit Service, DBMS Service

R1	Users must be able to register to the app, inserting personal information
R2	The system allows Store Managers to register their supermarket inserting the required information.
R3	Upon registering the supermarket, Store Managers must register themselves, providing a document that certifies their role
R4	The system allows users to log in to access the functionalities of the app
R5	The system allows store managers to log in to monitor influx of people

R6	The system allows users to select the supermarket they want to line up or book a visit for
R7	Users can get a ticket with a QR code to virtually line up
R8	Users who already have a valid ticket may not get another one or book a visit
R9	When getting a ticket to line up, the system must compute the time it takes for the user to get to the supermarket based on their position and chosen mean of transportation
R10	The system allows users to book a visit if capacity constraints are satisfied
R11	The system allows users to choose a day and a time for their visit among availables
R12	Upon booking a visit, users must insert product categories they intend to buy
R13	Upon booking a visit, users must insert how long they intend to stay at the supermarket. Long term customers may not insert such information and the system will infer it from previous visits
R14	The system should send a reminder to the user, at a time based on the estimated commute time
R15	The system must be able to let customers in the store by scanning the ticket's QR code
R16	The system allows users to exit the store by scanning the same QR code provided at the entrance
R17	The system allows guests to get a ticket to line up from the proxy devicE
R18	The system allows one person to enter for each ticket

5. Implementation, Integration and Test Plan

5.1. Overview

This section will cover the implementation, integration and testing of the CLup application. It aims to mitigate and possibly prevent any setbacks entailing changes on the project once started its development and do multiple tasks twice. It serves a critical purpose, given that no software is error free and a sound plan for implementation and testing is a crucial part of the project.

5.2. Implementation Plan

We will divide the project into different smaller components and will develop and integrate them adopting a bottom-up approach, i.e. we will start from the components that implement the most basic features and then go on to develop the most specific and complex ones. This method will be adopted for both the client and the server side, and for testing as well. The reason for the use of the bottom-up approach is to ease the bug tracking process, and facilitate the developers' workflow by dividing preemptively the development into smaller units which then interact with each other.

The System can be divided into four main subsystems: Data, Business Logic, View, External devices.

1. **Data**
   DBMS Service

2. **Business Logic**
   a. SM Access Manager
      User Access Manager
   b. Functionalities Manager + Google Maps Service
   c. Statistics Calculator
      Ticket Validator
   d. Request Dispatcher

   View (in parallel)
   Mobile App
   Webapp

   External Devices (in parallel)
   QR Scanner Software
   Fallback Device

The implementation will start from the most fundamental components, as others may rely on them, hence they are given priority. As a result, the DBMS Service, the component that allows access to the data used throughout the application, will be the first to be implemented as it is the most critical module and carries the highest risk.

We will then proceed to develop in parallel all components closely related to the DBMS, such as the User and SM Access Managers, that allow respectively Users and Store Managers to register and log into the application. We can then develop the Functionalities Manager, which comprises the Line-Up and Book A Visit services. In fact they rely upon the Access Managers, which we must have already implemented and tested. The former will also integrate the Google Maps API, which allows the correct functioning of the service and won't be implemented nor unit tested, as it is considered reliable, and will only sustain integration tests.
At the same time we can develop the Statistics Calculator, which talks to the DBMS Service to compute relevant statistics which will be served to the Store Manager, as well as used to infer the expected time of the visit of repeat customers.
Finally, we should develop the Request Dispatcher, which will act as an interface with all external devices such as the Fallback Device, the QR Scanner and regular View components such as the mobile app and the webapp.

Another block of components which will be developed in parallel is the software of the External Devices, i.e. the QRScanner and the Fallback Device. Our software will sit upon the existing firmware provided by the hardware manufacturer and will talk to the Request Dispatcher.

The View, consists of the mobile app for the users and the webapp for store managers, both of which can be built in parallel. In fact their integration with the backend will consist of an API call.

To sum up, the following are the blocks of components. One specialized team should handle the Data Tier and the DBMS Service as it requires specific knowledge of databases and it is the most complex and critical part of the project. Another team will work on the Business Logic block, and if possible, the View and External devices may be built by two different teams in parallel, to speed up development time.

5.3. Integration Strategy

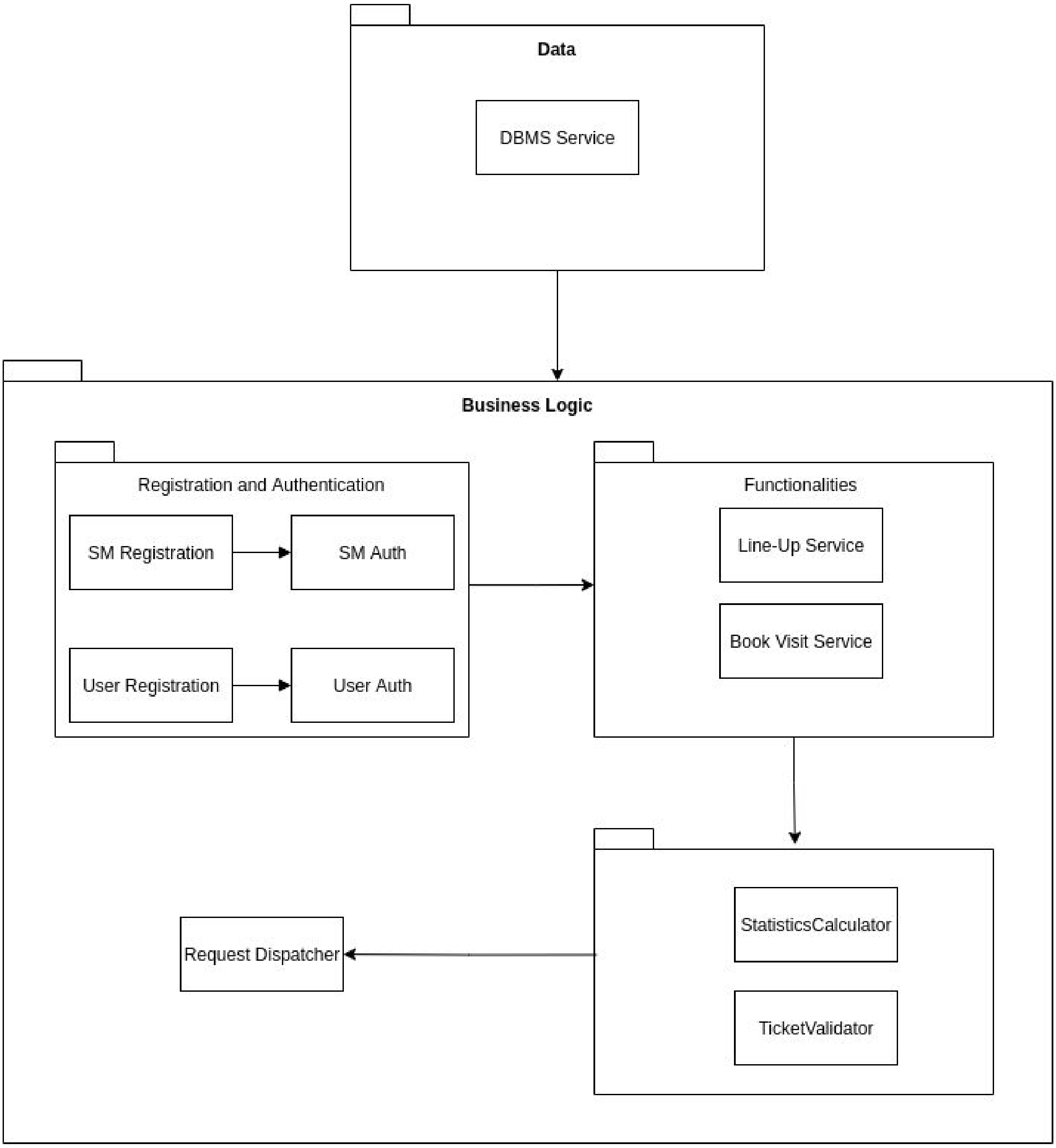
In addition to following a bottom-up approach, it is possible to implement components in parallel by using an incremental approach for the integration and testing processes. Once two components (within the same subsystem) are implemented, the integration and testing of those can be performed. Keep in mind that before starting with the integration process, all the unit tests for each component implemented must already pass. Once the unit test phase has been completed, the integration process can start.
The integration tests will be black box tests, aimed to verify the proper working of all functions in the same way as unit tests, but substituting stubs and drivers with the real corresponding modules.
Below we describe how the integration and testing are performed.

First the DBMS Service will be implemented and united tested, as it is the base upon which the whole Business Logic Tier relies.

1. **Data**
   DBMS Service

Then we will go ahead and implement the Business Logic Tier.

2. **Business Logic**
   a. SM Access Manager
      User Access Manager
   b. Functionalities Manager + Google Maps Service
   c. Statistics Calculator
      Ticket Validator
   d. Request Dispatcher



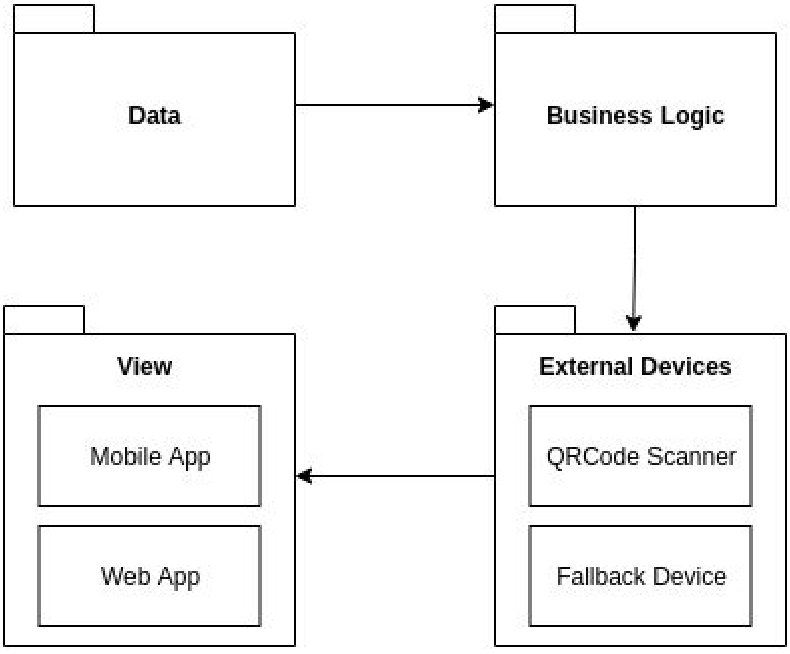
In parallel, independently from the other components, we will develop and test the mobile app and web app.

View
Mobile App
Webapp

The External Devices, consisting of the QR Scanner and the Fallback Device will be also implemented in parallel, independently from the other blocks.

External Devices
QR Scanner Software
Fallback Device

All the above mentioned subsystems will then come together, first by integrating the Data and Business Logic subsystems as soon as the subsystem itself has been integrated.
We will then proceed to integrate the External Devices subsystem first, and finally the View components.



5.4. System Testing

After the integration testing the system must be subjected to system tests.
First of all the system will be subjected to an **alpha test**, executed by the CLup developers in their production environment, checking all functions with a good number of test cases which should cover at least 80% of the code.
After the check of the functional requirements this test will also verify the non-functional requirements, in particular the performance under a heavy load: the database will be stressed with large amounts of data and all functions will be tested, checking their elaboration and response times.
The second and most important system test is the **beta test**, which will be executed in a real production environment, with the support of a supermarket: a group of voluntary citizens will install and try the mobile app, while the supermarket store manager will use the web app. This is particularly important to verify another time the correct behavior of the system (i.e. the absence of bugs), in a real environment with all its unpredictable events, but withstand an acceptance test, i.e. also to verify the usability and the correctness of the system for different types of users.

6. Effort Spent

Description	Tot
Initial discussion	4
Index + Introduction	2
2.1 Architectural Design - overview	3
2.2 Component View	8.5
2.3 Deployment View	7
2.4 Runtime View	8
2.5 Component Interfaces	4.5
2.6 Patterns + 2.7 others	4.5
3.1 User Interface Design - Mockups	10
4. Requirements Traceability	2.5
5.1 overview + 5.2 implementation Plan	5.5
5.3 Integration Strategy	6
5.4 System Testing	4
Final Revision	4

7. References

- Software Engineering 2 lecture slides
- IEEE Standard for Information Technology - Systems Design - Software Design Descriptions