Design Document - DD

January 2021

POLITECNICO
MILANO 1863

KONG XIANGYI
&
ZHANG YUEDONG

Contents

Chapter 1

INTRODUCTION

1.1 Purpose

The purpose of this document is to explain how we design and build CLup application. We demostrate the architecture design of the system, based on the rule of Top-down design approach, we describe the different design characteristics. We explain it from the view of component, deployment and runtime. To be more specific, we also show the user interface design to illustrate the CLup application step by step. The purpose of interface design is user-friendly and efficiently. Our aim is to let other stakeholders can easily understand and know the structure of CLup application.

1.2 Scope

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- Click Customer : The customer has the required technology to access the store. I.e a smartphone. They can use the customer terminal software.
- Brick Customer : The customer doesn't have the required technology to access the store, they have to hand out "tickets" on the spot.
- Store Manager : They have to manage the Store System, include the software and hardware.

- Ticket: The ticket is a document which contains three key information: QR Code, the estimated departure time, the queue number, and the Store Planned Roadmap. To the click customer, it's **E-ticket** but to the brick customer,it's **Paper Ticket**,and doesn't contain the estimated departure time, and just a General Store Map without the Planned Road.
- QR Code : When customer booked a visit, they will received a QR Code.
- QR Code Scanned Machine : A hardware, the Click Customer can use this machine scan their QR code.
- Tickets Hand-Out Machine : A hardware, the Brick Customer can use it retrieve their Ticket.
- Store Planned Roadmap: A store map that includes a finer way which is recommended form Store System.
- Digital Counterpart : A hardware, it with show the queue number.
- Store Back-End System : A software, as the back-end manages all stuffs.
- On-Time Store Data : A dataset that includes the store's on-time date.
  - The current queue
  - The customers in the store
  - The maximum number of people in the store.
- Long-Term Customers: The Click Customer who visited the store more than one time by the CLup mobile application.

1.3.2 Acronyms

- RASD - Requirement Analysis and Specification Document
- DD - Design Document
- CLup - Customers Line-up
- UI - User Interface
- IOS - iPhone OS

- PC - Personal Computer
- IaaS - Infrastructure as a Service
- CRM - Customer Relationship Management
- LAN - Local Area Network
- RAPS - Reliable Array of Partitioned Service
- RACS - Reliable Array of Cloned Services

1.3.3 Abbreviations

-

1.4 Revision history

1.5 Document Structure

Chapter 2

ARCHITECTURAL DESIGN

In this chapter, we will describe the architectural design of our system.

We will use the Top-down design approach, design the very high-level structure first, and then gradually work down to detailed decisions about low-level constructs. Finally, arrive at detailed decisions.[4] Let us start with the High-level components and their interaction.

2.1 Overview: High-level components and their interaction

We chose **3-Tiered architecture** with the **Thin Client** strategy for our system. As shown in Fig.2.1.[1]

Tier-1 is the **Presentation Layer**. This layer will deploy the Click Client's Mobile Application, the Store Manager's Management System, and even Digital Counterpart and Ticket Hand-Out Machine's presentation.

Tier-2 is the **Logic Application Layer**. This layer will deploy our Back-End System's components.

Tier-3 is the **Data Access Layer**. This layer includes our DBMS and the Data Base.

Finally, our high-level architecture is shown in Fig.2.2. The Store Manager's PC and other hardware will connect with the Ethernet Switch Hub. The Click Customer's Mobile App will communicate with our Back-End System via the Internet.

More detailed components will be introduced in the next sections. In particular, in Sec.2.3. There, we will introduce the DMZ, load balancer, and other Deployment plan selections.

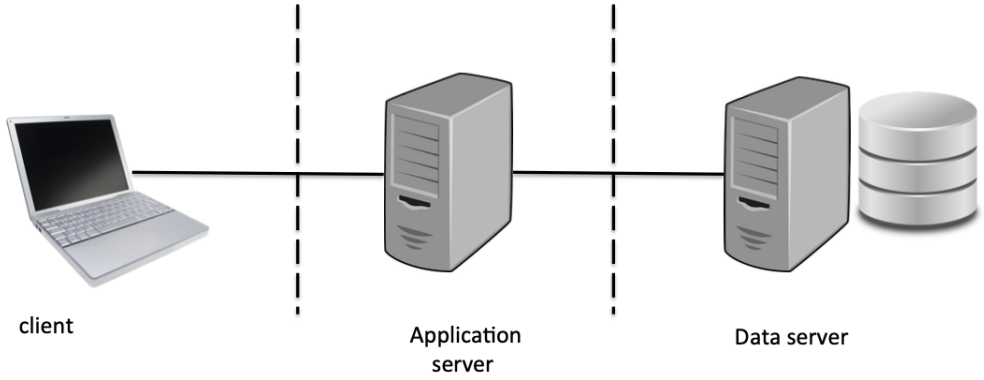

Figure 2.1: 3-Tiered architecture

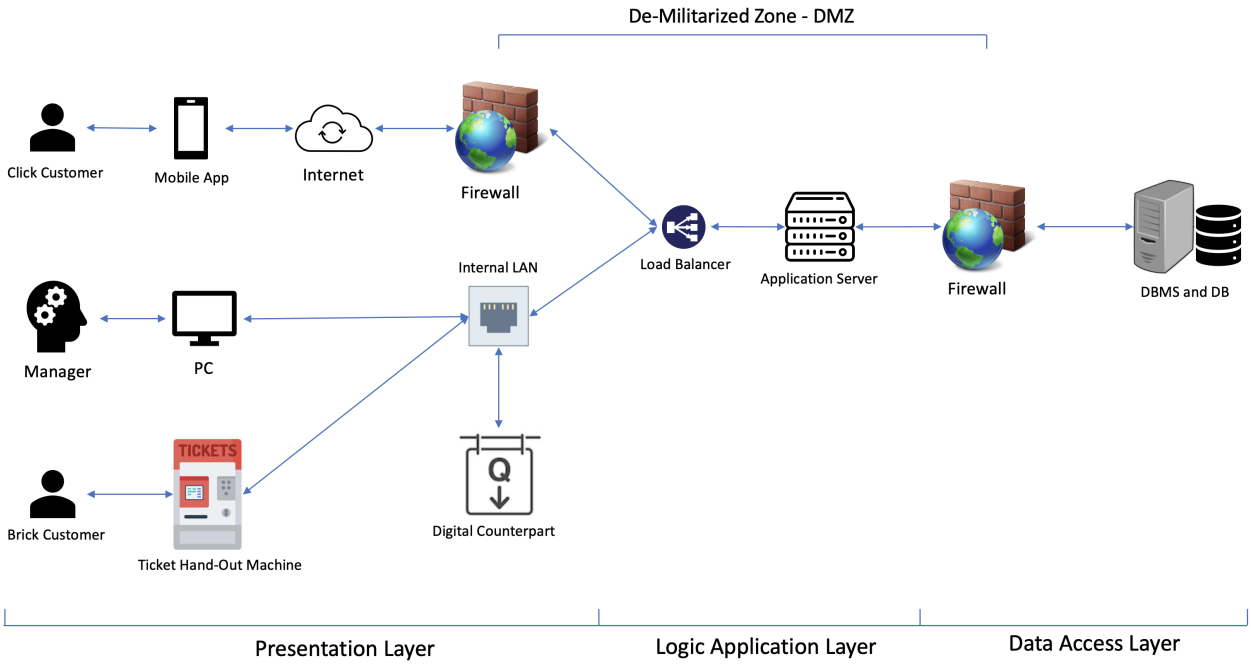

Figure 2.2: High level architecture

2.2 Component View

The following diagram Fig.2.3 is the **Component Diagram**, and it contains all components in our system. This diagram shows the 3-Tiered Architecture of our system. The Yellow components are in Presentation Layer. The blue components are in the Logic Application Layer, and the Green components DBMSServer is in the Data Access Layer. We will separately introduce all these components in this section, for the interfaces, please go to Sec.2.5.

The **Redirector** component is a bridge for communicating the Presentation Layer and the Logic Application Layer. It is transparent to User and back-end systems, plays the role of forwarding information on both sides, and connect interfaces. It has four subsystems, respectively, responsible for communicating with four types of devices, as shown in Fig.2.4.

The **ClickCustomerRedirector** will transfer the message between Mobile Application and the Application Server. The Register message will pass by the RegisterNewClickCustomer, and look out the Valid/History Booking will use the GetClickCustomerData Interface, this Interface will return all information of this Customer. For the most important operation - Booking a visit, use GetBookingSchedule to get the available Time/Date. Then, through the ModifyClickCustomerData Interface to submit the Booking, by the way, it also can modify other available data of this Customer. Moreover, the SendNotifyToClickCustomer Interface will handle Notify message. The Mobile Application can actively "pull" Notify message through this Interface from the Back-End System. To "push" Notify information, we will use the Observer Pattern. It will introduce in Sec.2.6

The **TicketMachineRedirector** will transfer the Ticket Hand-Out Machine's message. It needs two pieces of information, the allowed maximum number of people in the store and how many Customers in the Queue in time. So The GetQueueSchedule and GetMaxNumberInStore Interface can do these operations, and these two fields will show on the Ticket Machine's screen. Finally, the retrieve ticket operation will handle by the ModifyQueueSchedule Interface. It will put a BrickCustomer into the Queue.

The **DigitalCounterpartRedirector** will help the Digital Counterpart display the next queue number. The Customer can enter the store after found his number on the Digital Counterpart.

The **ManagementSystemRedirector** will function for the Management System of our Store Manager. This redirector will transmit the message between the Presentation layer's web page and the Back-End System. Through the Interface shown in Fig.2.5, the Manager can realize his operation like reschedule someone's Booking, adjust the Queue order, or lower the maximum number.

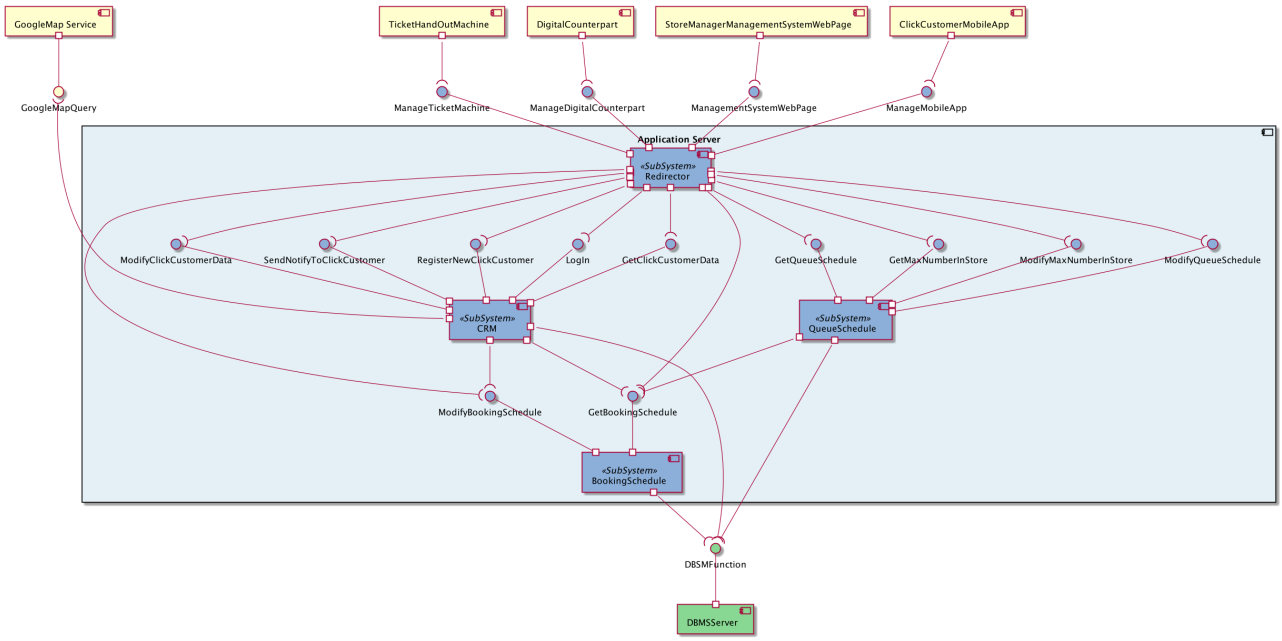

Figure 2.3: Component Diagram

The next three subsystems are the core subsystems in our application layer.

The **CRM** component is responsible for communicating with Click Customers' End, handle the register, Log-In, Booking, data-send, and notify-send operation. The Booking operation will communicate with the Booking Schedule component, send the Booking Information, general E-Ticket, and store all data in the database. In addition, it must also help the Customer calculate the estimated departure time from depart place to the store, so it will call the GoogleMap API to query it.

The **Booking Schedule** component is working for all Booking tasks. Receive the Booking from CRM, handle the Store Manager's modified Booking, send the Booking schedule to the Queue Schedule component, and store all Booking Schedule to the database.

The **Queue Schedule** subsystem is the main component to communicate with the Store Manager's Management System. It has to take the Booking schedule to calculate the Queue order and handle the Store Manager's modification for the Maximum value or Queue order.

At last, the **Database System**, it will manage all database, and handle the CRM, BookingSchedule, QueueSchedule's Query/Insert/Update.

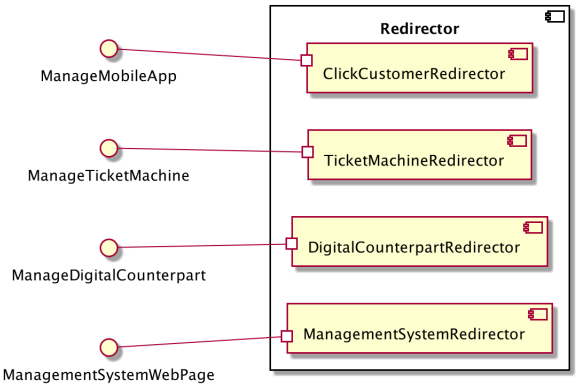

Figure 2.4: Redirector Component Diagram

2.3 Deployment View

Our deployment situation, as shown in Fig.2.6. Three colors respectively represent the three layers. The Green devices are in the Data Access Layer, the Blue devices are in the Logic Application Layer, and the Yellow devices are in the Presentation Layer.

For all our devices, to reduce the development/management/maintenance cost, we will use the same Operation System - Ubuntu 20.10, and run in the same Runtime Environment - Java SE Runtime Environment 8u271.

In the **Presentation Layer**, The Store Manager's PC, the Digital Counterpart, and the Ticket Machine are the Internal Devices, which means we must manage them ourselves. We built a LAN to connecting them with the Application Server via the Switch Hub. In our LAN, we used the Ethernet Protocol. Moreover, we have chosen the Thin Client strategy in the system architecture, so all our Internal Devices can run on the Raspberry Pi 4. It is a very low-cost deployment solution.

In contrast, the Mobile Application is the External Device that will run on the Click Customer's SmartPhone. It has to connect with Back-End System via the Internet. We will separately develop the Client End for the two kinds of systems - IOS and Andriod System.

In the **Application Layer**, We refer to the network security part of the information system book P.243. We used two Firewalls to build a **DeMilitarized Zone - DMZ**. The external network can only access the resources exposed in the DMZ (Our Application API), and the rest of the network is behind the second firewall. The DMZ functions as an isolated subnet between the Internet and the private network, So that our database can be better protected.[1]

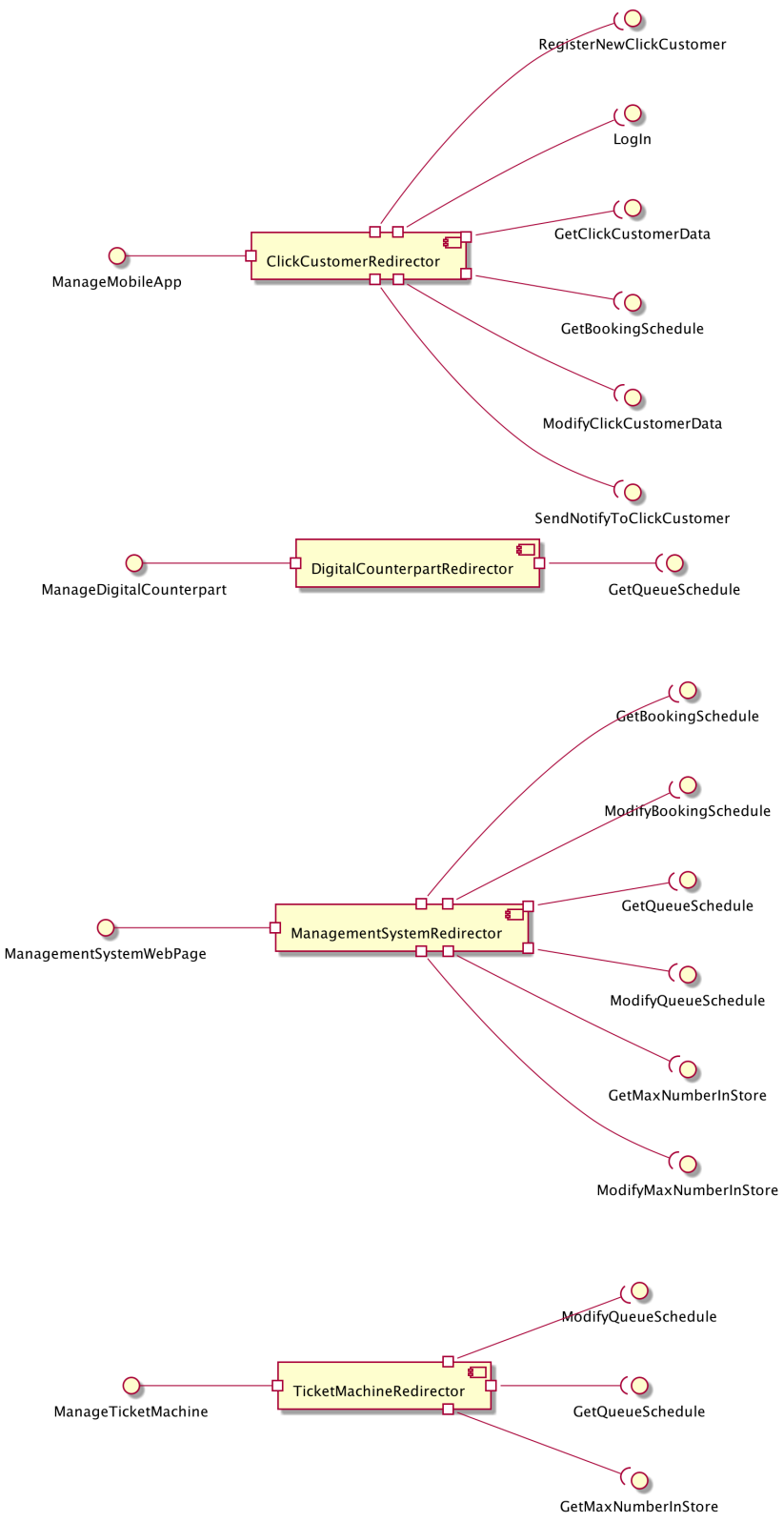

Figure 2.5: Redirector's Subsystem Component Diagram

For the **Load Balancer**, we must first ensure that our Internal Devices can allocate enough Server resources and then ensure the Click Customer's resources, so we considered two plans.

1. Allocate dedicated Application Servers for our Internal Device, and other Application Servers connect with Load Balancer.

2. Do not allocate server resources separately for internal devices, but set higher weights for Internal Devices in the load balancer.

Plan 1 can guarantee that our system's most important part can work well, whether the balancer is working or how crowded the network is. Plan 2 ensures that when some servers are not working, the well-function servers can always allocate to Internal Devices. After weighing, we finally chose plan 2. Because we think Plan 2 has higher system availability/maintainability.

As mentioned above, for the **Application Server**, we adopted the Reliable Array of Cloned Services - RACS, like the figure in the Information System Book P.202[1], Fig.2.7. The load balancer must allocate the higher weights to Internal Devices, and all our Application Server will run the same application. There is a different point in our case from the figure: we have separated the database and the Application Server. Nevertheless, cause our business is "write-intensive", so we used the shared-disk configuration in essence.

Finally, the **Database system**, we selected PostgreSQL as the Database Management System cause it is Open Sources and stable. Then we will use the relational database to implement it. For Security, as the DMZ mentioned above, only our private network can access this system, so that the external network will difficultly attack our database.

2.4 Runtime View

2.5 Component Interfaces

2.6 Selected Architectural Styles and Patterns

2.7 Other Design Decisions

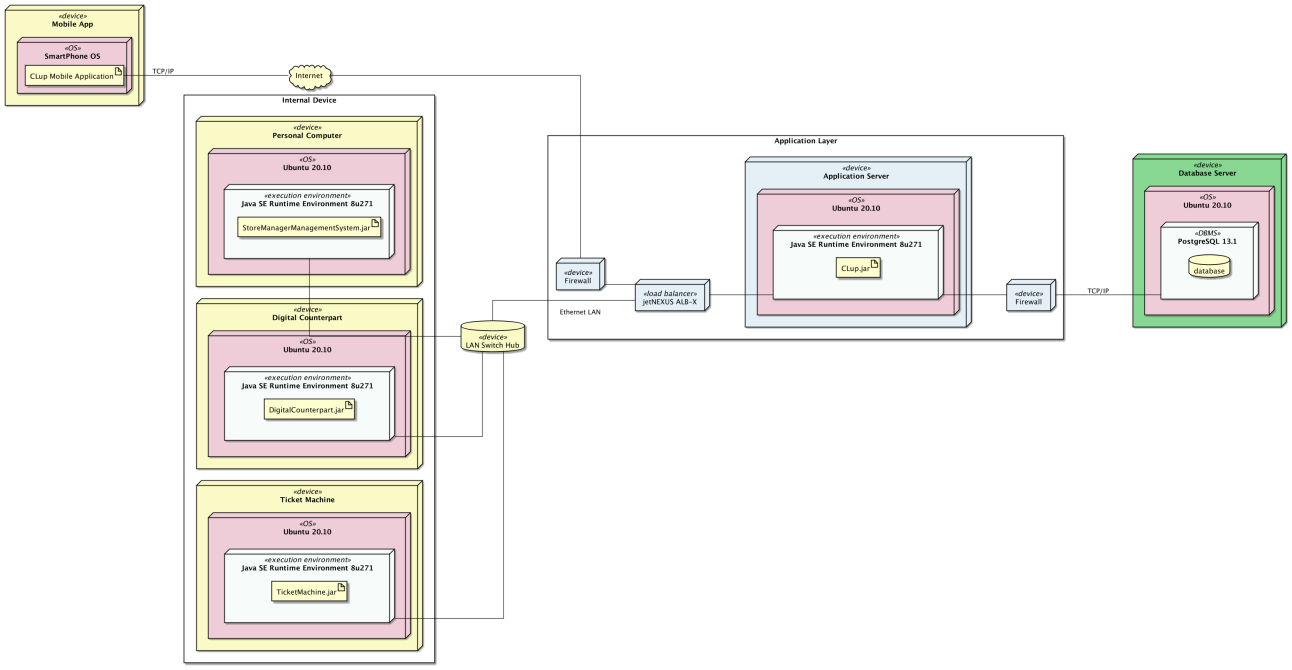

Figure 2.6: Deployment Diagram

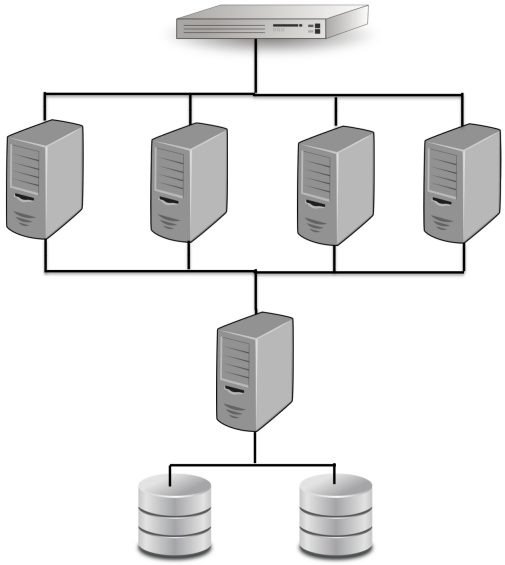

Figure 2.7: RACS: Shared-disk o cluster Configuration

Chapter 3

USER INTERFACE DESIGN

3.1 UI design

We illustrated some main UI designs in RASD document, here, we describe more details for the specific UI button.

UI-Date Time After enter the booking button, there is an option of date and time, Fig.3.1, the CLup will give the available booking date, which wiith the deep dark color. And the same method for selecting available booking time.

UI-Duration Time It needs the user to estimate his shooping time, Fig.3.2, for long-term users, they can press the "Analysis" button, the system will approximate a duration time depending on his shopping history.

UI-Ticket Fig.3.3 shows the users current ticket and history ticket. The current ticket with the red point, to inform the user it is the valid ticket.

UI-Ticket PDF it present the PDF ticket for the user, Fig.3.4 the QR-code using for scan on the entry. And it also give the suggestion of departing time from current position to the store.

UI-Notification Fig.3.5 which display the notification of rescheduled ticket, the red point means the unread message. And it will also give the new ticket to the user.

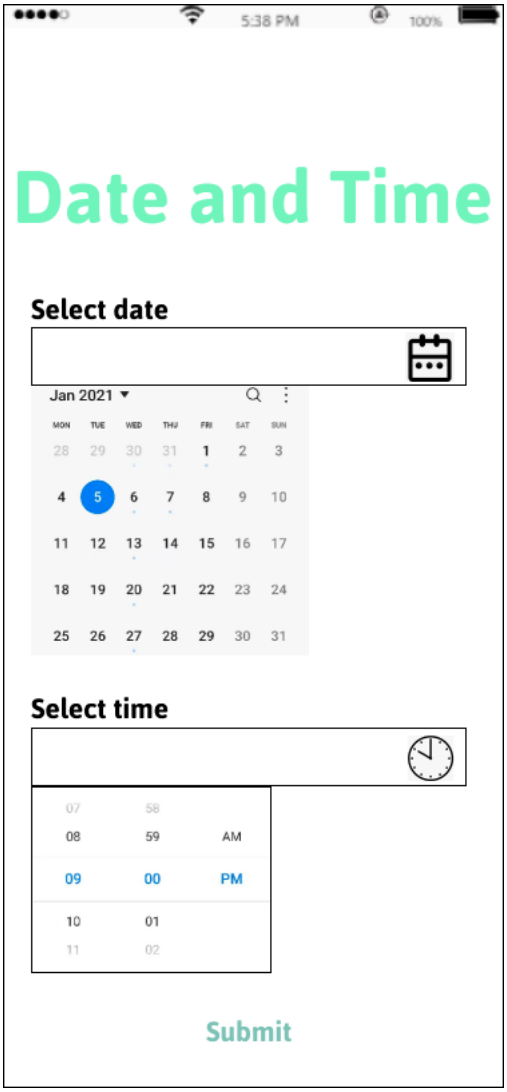

Figure 3.1: UI-Date Time

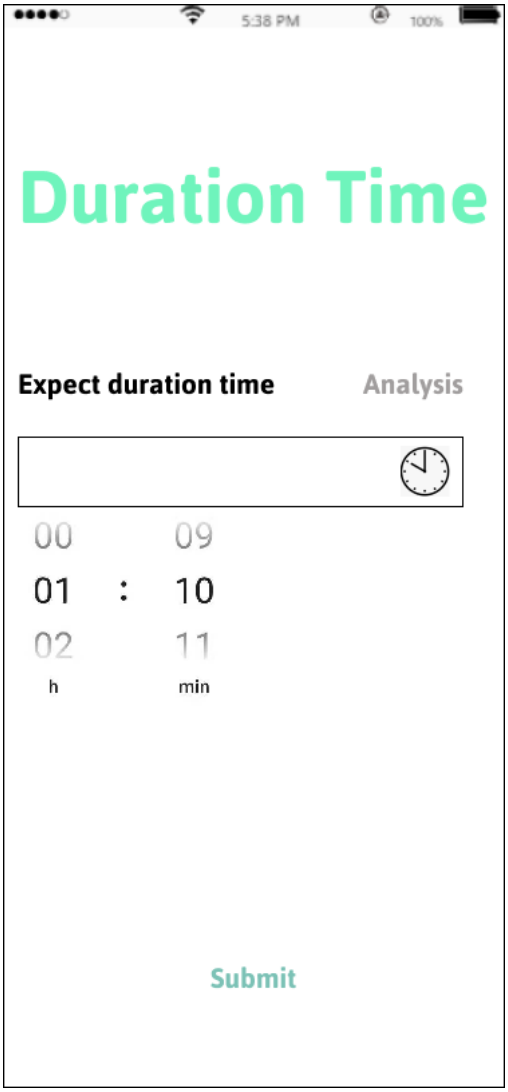

Figure 3.2: UI-Duration Time

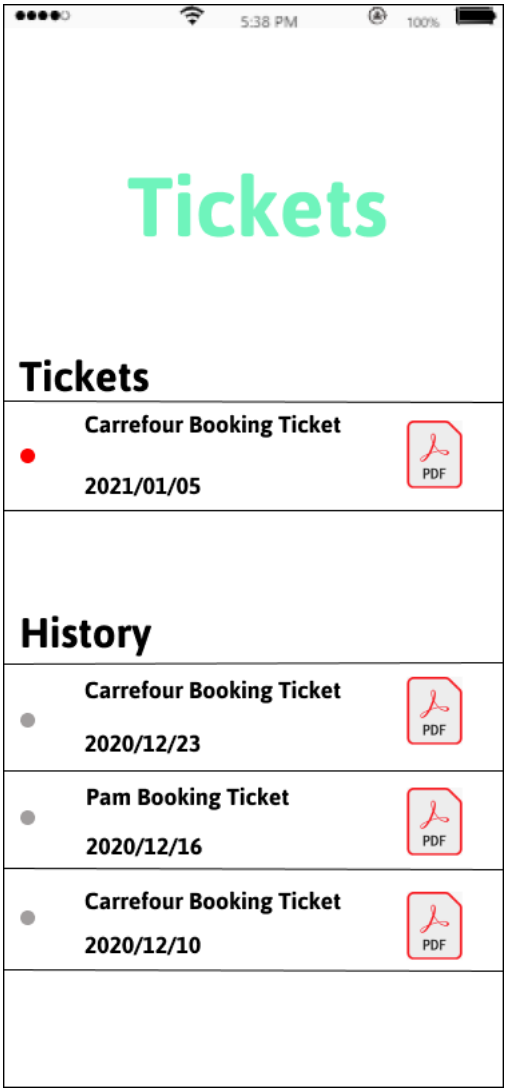

Figure 3.3: UI-Ticket

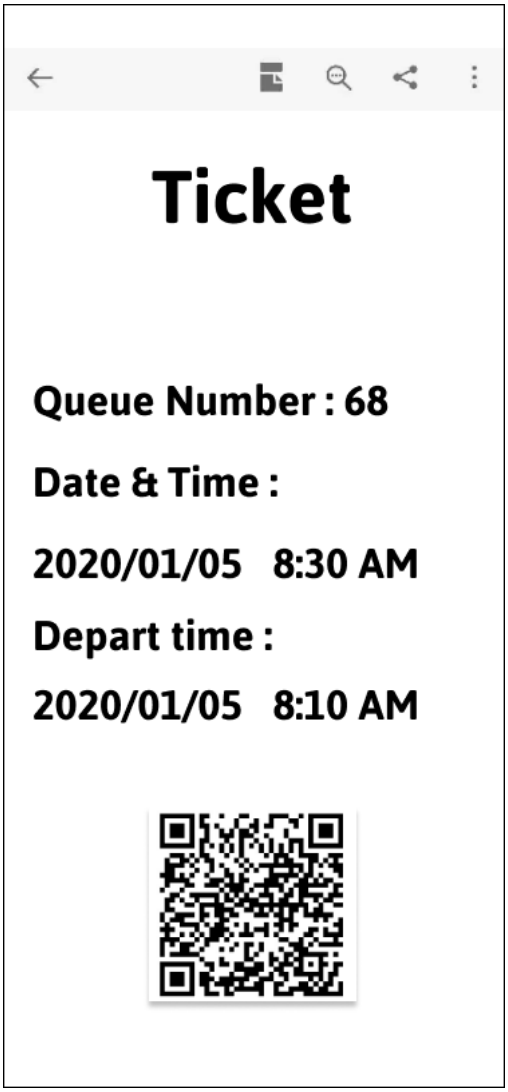

Figure 3.4: UI-Ticket PDF

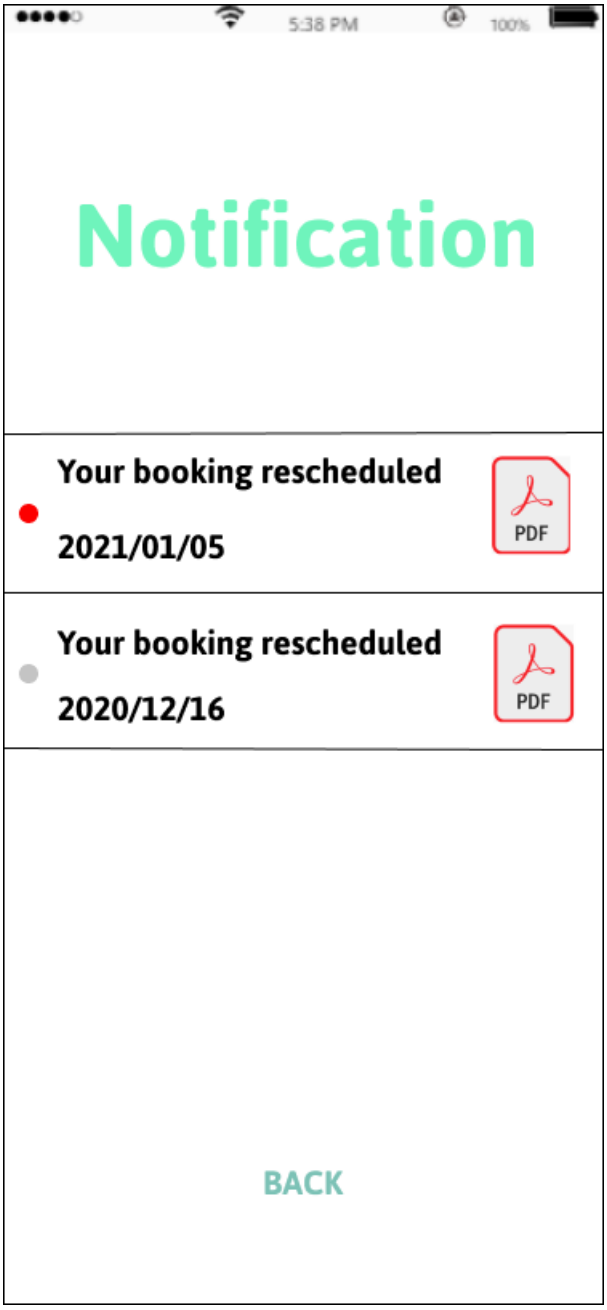

Figure 3.5: UI-Notification

3.2 UX design

We present the UX diagram of user, Fig.3.6 the diagram show the proceed of how the user from login to get the valid ticket. Corresponding to the UI design, it show the total user view of the CLup application.

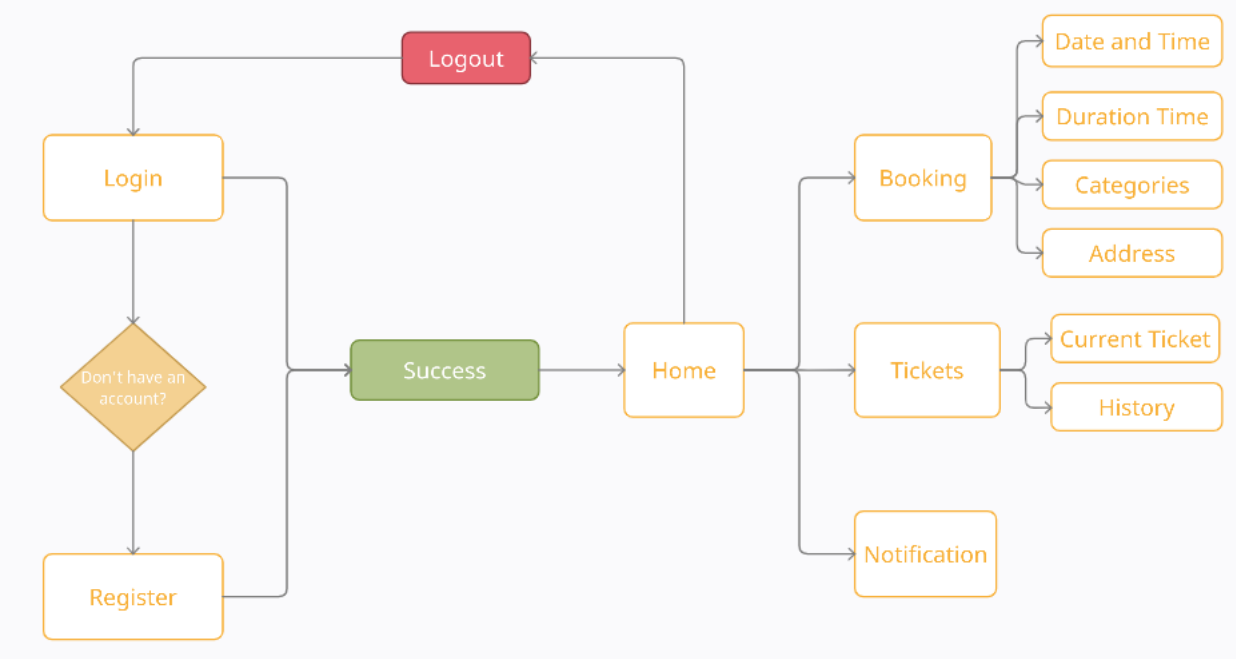

Figure 3.6: UX-User

Chapter 4

REQUIREMENTS TRACEABILITY

Chapter 5

IMPLEMENTATION, INTEGRATION AND TEST PLAN

Chapter 6

Effort Spent

- **Kong XiangYi**

Date	Task	Hours
2021/01/02	Group discussion and task assignment	2h
2021/01/04	UI design	2h
2021/01/05	UX design diagram in chapter 3	2h

- **Zhang YueDong**

Date	Task	Hours
2021/01/01	Launch DD	2h
2021/01/02	Group discussion and task assignment	2h
2021/01/03	Did S.2.2.1 and S.2.2.2 Component Diagram	3h
2021/01/04	Did S.2.2. Component Diagram describe.	2h
2021/01/05	Did S.2.3. Deployment Diagram and describe.	5h

Bibliography

[1] Cinzia Cappiello, Mariagrazia Fugini, Paul Grefen, Barbara Pernici, Pierluigi Plebani, Monica Vitali. (7 settembre 2018) Fondamenti di Sistemi informativi: per il Settore dell'Informazione.

[2] "IEEE Standard for Information Technology–Systems Design–Software Design Descriptions," in IEEE STD 1016-2009 , vol., no., pp.1-35, 20 July 2009, doi: 10.1109/IEEESTD.2009.5167255.

[3] "ISO/IEC/IEEE Systems and software engineering – Architecture description," in ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000) , vol., no., pp.1-46, 1 Dec. 2011, doi: 10.1109/IEEESTD.2011.6129467.

[4] Elisabetta Di Nitto, Matteo Rossi. (A.Y.2020/2021) The slides of the Software Engineering 2 course.