City Shared Parking Management System

Software Requirements Specification

2.0

2018.12.9

Chengcheng Lou

Requirements Analysis Engineer

Revision History

Date	Description	Author	Comments
2018.11.20	1.0	Chengcheng Lou	Create an overall document framework
2018.11.25	1.1	Chengcheng Lou	Complete the first and second parts
2018.11.29	1.2	Chengcheng Lou	Add multiple graphical descriptions of functional requirements
2018.11.30	1.3	Chengcheng Lou	Update the text description of the functional requirements
2018.12.1	1.4	Chengcheng Lou	Fulfill non-functional requirements
2018.12.2	1.5	Chengcheng Lou	Complete requirement change management
2018.12.9	2.0	Chengcheng Lou	Format, index, check and proofread

Table of Contents

1. Introduction

This document is a requirements analysis report of the City Shared parking management system (CS-PMS). In the description process, this document comprehensively uses the system environment diagram, data flow chart, functional structure diagram, entity relationship diagram, swimlane flow chart. This document attempts to illustrate the system requirements in detail through vivid graphics and necessary text descriptions. The third part establishes the models from multiple perspectives, including: scene modeling, object-oriented class modeling, data flow modeling, and behavior modeling.
This chapter first introduces the main design objectives of CS-PMS, and then introduces the scope of the system. The definitions, acronyms, and abbreviations used in this document are then listed. Finally, the overall organizational structure of this paper is summarized.

1.1 Purpose

The purpose of this document is to define the project requirements and describe the system structure. With its help, software development companies can develop projects with confidence

1.2 Scope

The CS-PMS is designed to help solve problems such as parking difficulties and uneven parking Spaces. We want to push for open parking Spaces on university campuses and residential areas. More specifically, the urban Shared parking management system mainly provides the following functions for different users:

For the administrators: Collecting and updating the information of idle parking Spaces in real time, and checking the parking records of users. At the same time, the administrator can also manage user authentication information.

For drivers: Checking the parking space information in real time, and reserving the free parking space in advance. There are also online parking confirmation and payment operations.

For guards: they can complete the operation including charge, parking space management, checking the vehicle access record and so on.

For employers: In every parking space, employers can easily manage their own employees.

1.3 Definitions, Acronyms, and Abbreviations

CS-PMS: the city shared parking management system

administrator: CS-PMS administrator
driver: CS-PMS driver, the primary user of the system
guard: CS-PMS guard, they are responsible for managing community parking lots and enclosed parking lots, which are also important sources of parking information.
employer: CS-PMS, can hire or fire guards
Parking Record: Parking Record Table
Parking Lot Info: Parking Lot Information Table
Payment Record: Payment Record Table
User Info: User Information Table
PM: project Manager
CCB: change control block
SRR: system requirements review
RTM: Requirement tracking matrix
RCR: Requirement change request
RMP: Requirements change control report

1.4 References

[1]王安生. 软件工程化[M].北京: 清华大学出版社, 2014
[2] IEEE Software Engineering Standards Committee. IEEE SA 830-1998, IEEE Recommended
Practice for Software Requirements Specifications. October 20, 1998.

1.5 Overview

The rest of the document is organized as follows:

General Description: This chapter includes product perspective, product functions, user characteristics, general constraints and assumptions and dependencies.

Functional Requirements: Several schematic modeling methods are used in this chapter. Vividly explain the functional requirements of the CS-PMS.

Non-Functional Requirements: Introduce the system requirements from another perspective.

Change Management Process: Use swimlane flowchart to illustrate the correct flow of requirements changes.

2. General Description

2.1 Product Perspective

This system is different from the common parking management system in the market. It expands the boundaries of parking management services. The

administrator combines all the parking places in the city to know the occupancy information of the parking space at the first time. Then, it provides drivers with real-time query and precise navigation services.

2.2 Product Functions

To the administrator: He can manage the local city parking space information, which belong to the residential area, school, public parking lots and street. In addition, the administrator should change the account information for the user if requested.

For drivers: functions include: querying free parking Spaces, reservation and navigation, parking and payment. Certainly, drivers don't have to pay any fees to park on the side streets or in public parking lots.

For the guard: He need to collect and upload parking space information. Another import function is to charge the driver.

For employers: functions include: managing owned employees, such as appointment or dismissal.

2.3 User Characteristics

Driver: The most important role, the whole system serves them.

Administrator: Key role, who is responsible for the collection, collation and update of system information, such as parking space information.

Guard: The basic role, he needs to manage a specific parking lot and upload parking information to the system.

Employer: A role that can manage a doorman.

2.4 General Constraints

This is a system that supports high concurrent access. It must run on a high performance server, and use a more stable Linux operating system. Besides, the development cycle of the first version of the system is within 2 months. Moreover, the final products include the HTML5 web manager, two mobile apps are based on the android and ios operating systems, respectively. And a product description.

2.5 Assumptions and Dependencies

1. The local government can provide information about available parking Spaces on highways and sidewalks.

2. Internet-connected hardware is embedded in the parking lot, which uploads parking space information to the cloud server

3. This system considers the parking process and does not involve the specific management of the parking lot.

4. A reserved parking space will not be occupied by other drivers.

5. The system allows drivers to reserve parking Spaces, but does not allow drivers to reserve parking Spaces on the sidewalk, but can provide navigation services for drivers.

6. Operators and users shall use the system in accordance with the regulations and shall not commit malicious damage.

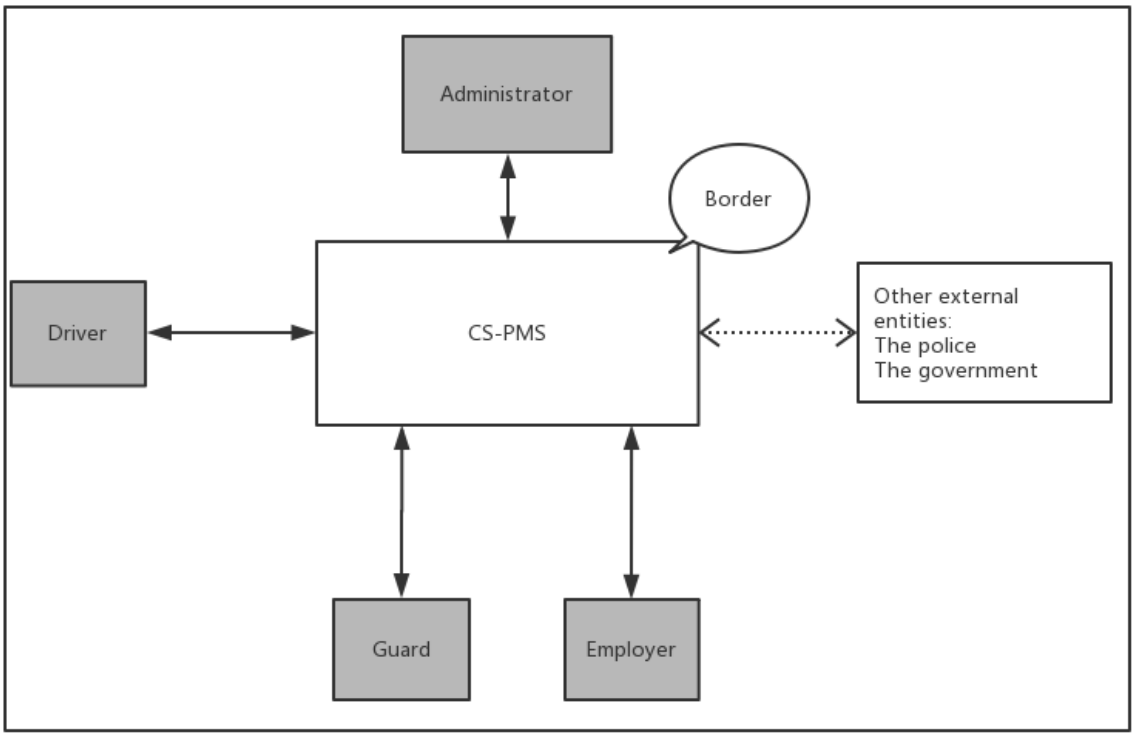
3. Functional Requirements

This chapter is the most important part of the whole document, which gives the functional requirements definition of the CS-PMS.

3.1 Context

3.1.1 Context Diagram

DIAGRAM_1



<CS-PMS>

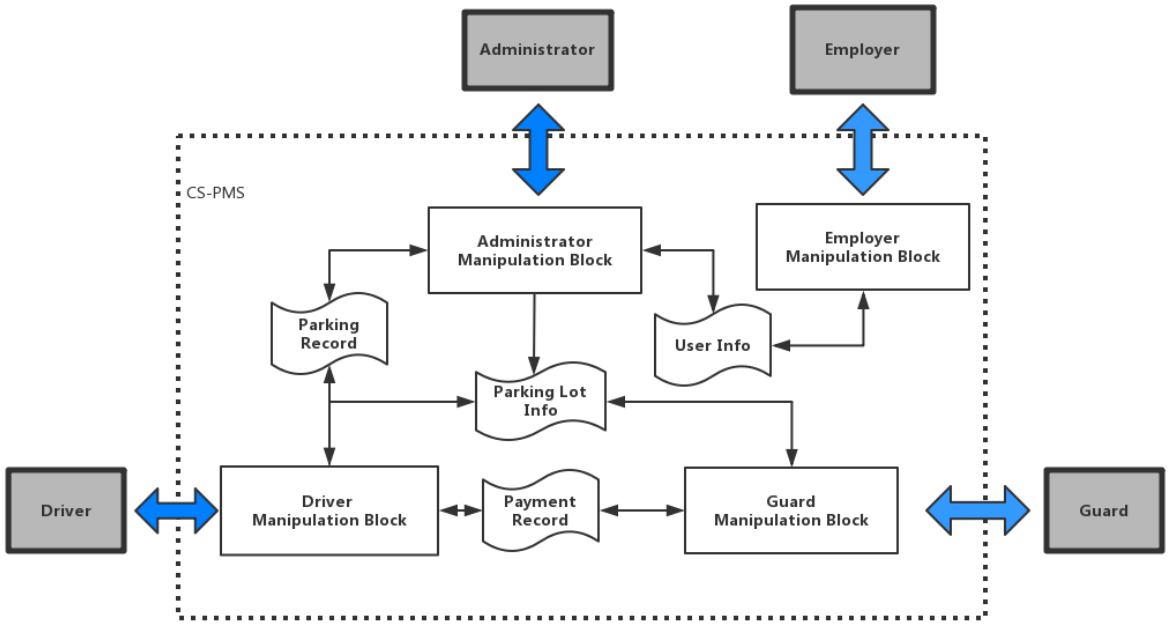
3.1.2 Context Analysis

The context diagram shows the relationship between CS-PMS and the external environment entities.
The middle rectangle represents the CS parking management system.
The surrounding rectangles respectively represent the external entities of the system: drivers, the guards of the duty room in the parking lots (residential area, university campus, institution courtyard, closed public parking lots), system administrators, and employers.
Arrows show the direction in which data flows, and each entity communicates with the system through a specific interface.

3.2 Data flow

3.2.1 Data Flow Diagram of Level1

DIAGRAM_2



3.2.2 Illustration

The data flow diagram breaks down the internal parts of the system into four functional blocks that show the flow of data between external entities and internal functional modules of the system. External entities invoke system functions to query or modify system internal data. The internal data of the system mainly includes: parking lot information, payment record, parking record and user information

Parking Lot Info: It contains the information of all parking Spaces in a city, including free parking Spaces, reserved parking Spaces and occupied parking

Spaces. Normally, the guards submit the parking lot information to the system, and the administrator is responsible for updating the parking space occupancy information. Therefore, drivers can query the parking space information near the destination in real time.

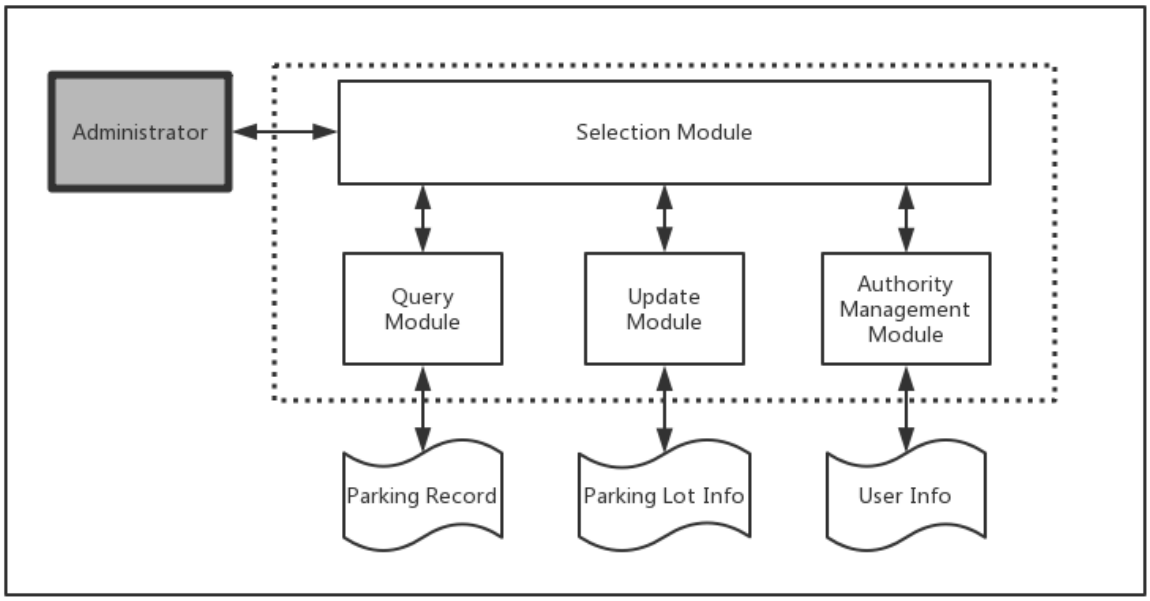
Payment Record: It contains the driver's payment record in a parking lot, which is checked by the guard when the driver leaves.

Parking Record: Open community parking spaces can cause safety problems. This record keeps vehicle access information in a parking lot for a week. Both the administrator and the driver can access this record, but the driver can only see his own parking record.

User Information: It includes a user's account, password and other information. Two sub tables are derived from it -- employee information table and driver information table. Considering the security mentioned above, the driver information is used to complete the visitor registration. With the guard's information table, an employer can appoint or fire a doorman. Beyond that, administrators can access all information at any time. But employers and guards can only access some information under certain circumstances.

3.2.3 Data Flow of Administration Manipulation Block

DIAGRAM_3



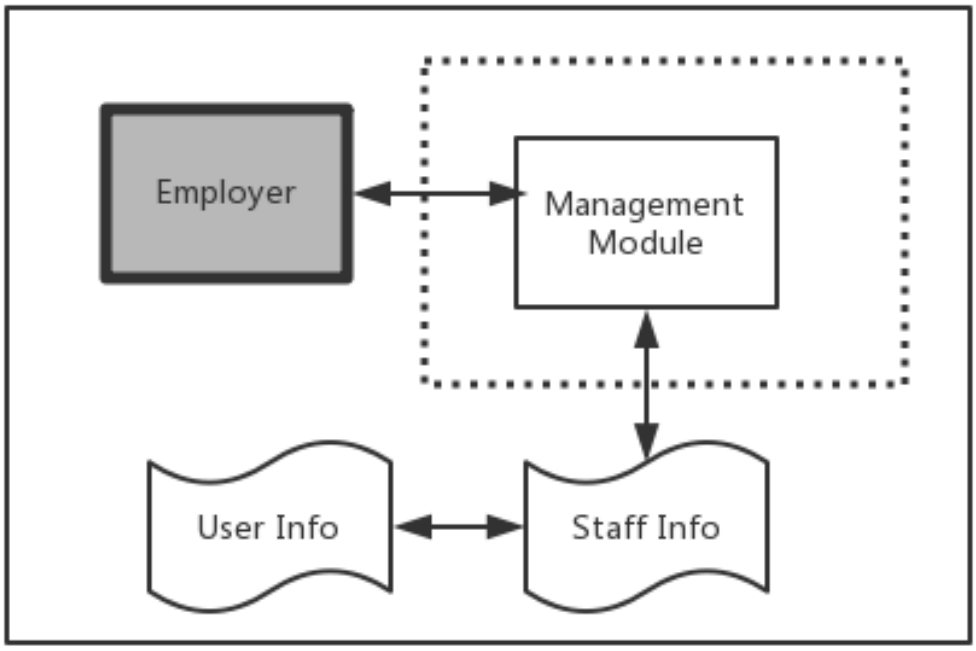
This diagram describes the direction in which data flows as the system administrator interacts with the system.

Administrators can view all parking records (a feature that the police and local authorities would like) or update the occupancy information of a particular parking lot. Of course, there is a most basic function - user rights management.

Before starting his work, the administrator uses an intermediate module called the selection module to select functions.

3.2.4 Data Flow of Employer Manipulation Block

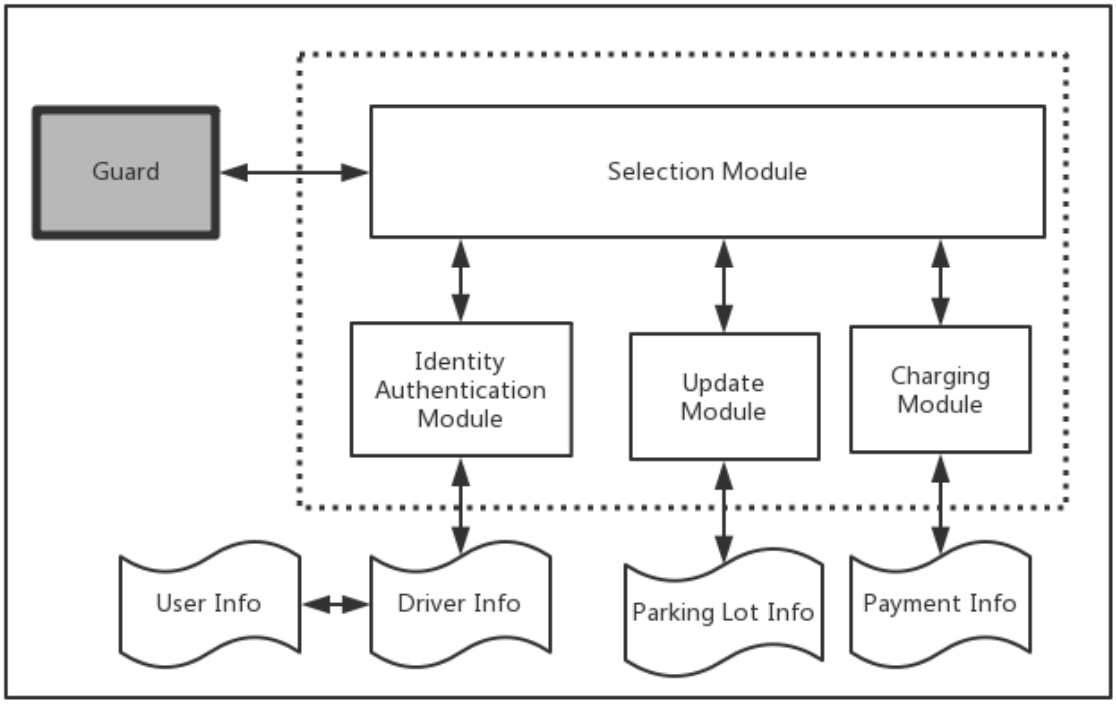
DIAGRAM_4



This function block contains only one submodule. It provides an interface for employers to manage their employees. The employee information table is a child table established on the user information table, which records the unit, employee number, position, salary and other information.

3.2.5 Data Flow of Guard Manipulation Block

DIAGRAM_5



In the figure, the gatekeeper has three types of operations.

The first is driver identification. For some special places, such as schools and residential areas, they need to register visitor information when the free parking space is opened to the public. Similar to the employee table mentioned earlier, driver information table is also a child table based on the user information, which records the driver's identity information such as id number and contact information.

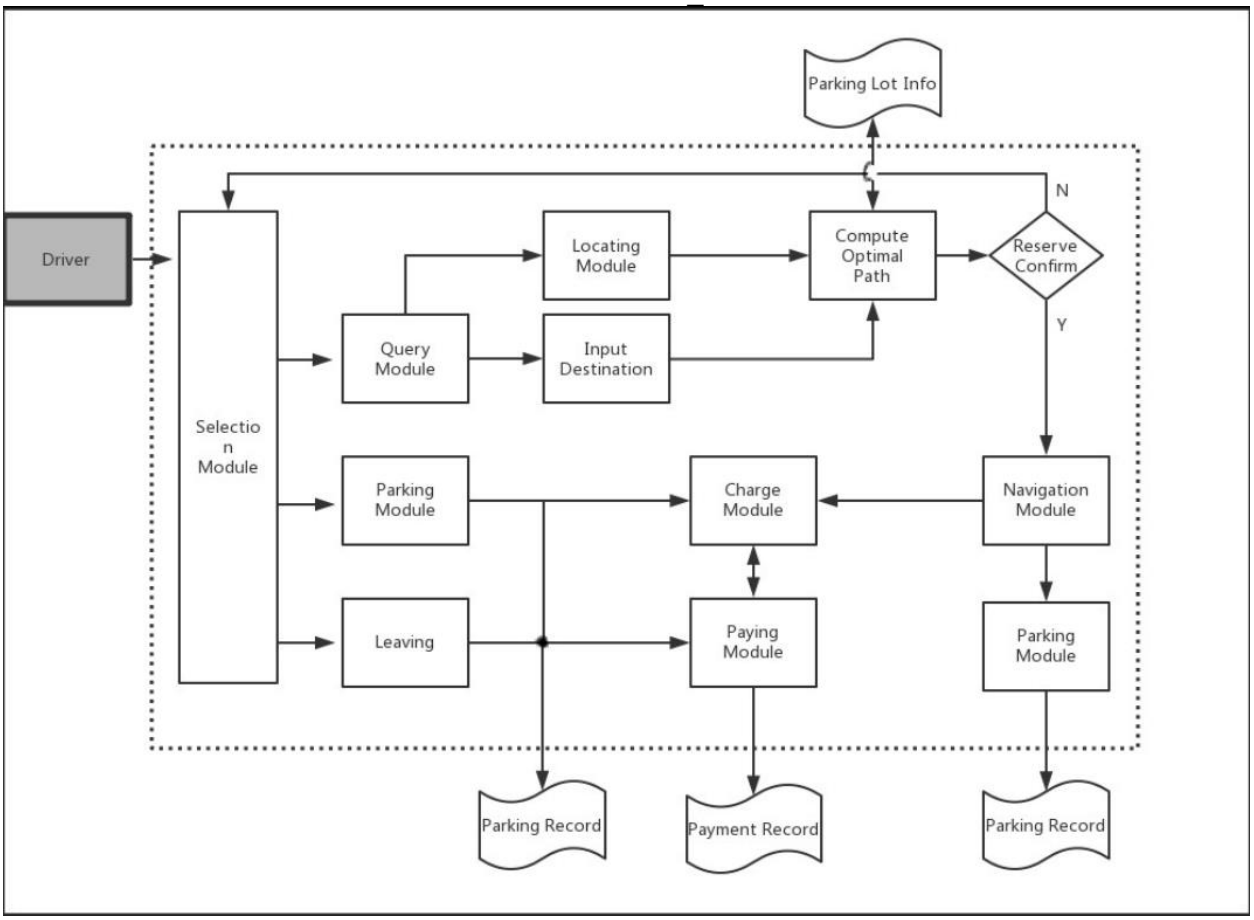
The second is to update records. Guard can collect information about the information of parking Spaces and then update data about parking lots.

Another is the charge. The guard checks the payment record to decide whether to let a car leave the parking lot.

<CS-PMS>

3.2.6 Data Flow of Driver Manipulation Block

DIAGRAM_6



This is the most complicated diagram, where an arc indicates that two arrows do not intersect. In contrast, dots indicate intersections. This picture describes the optional operations of drivers, major users of CS-PMS. The whole function block provides the driver with three operations: querying, parking and leaving.

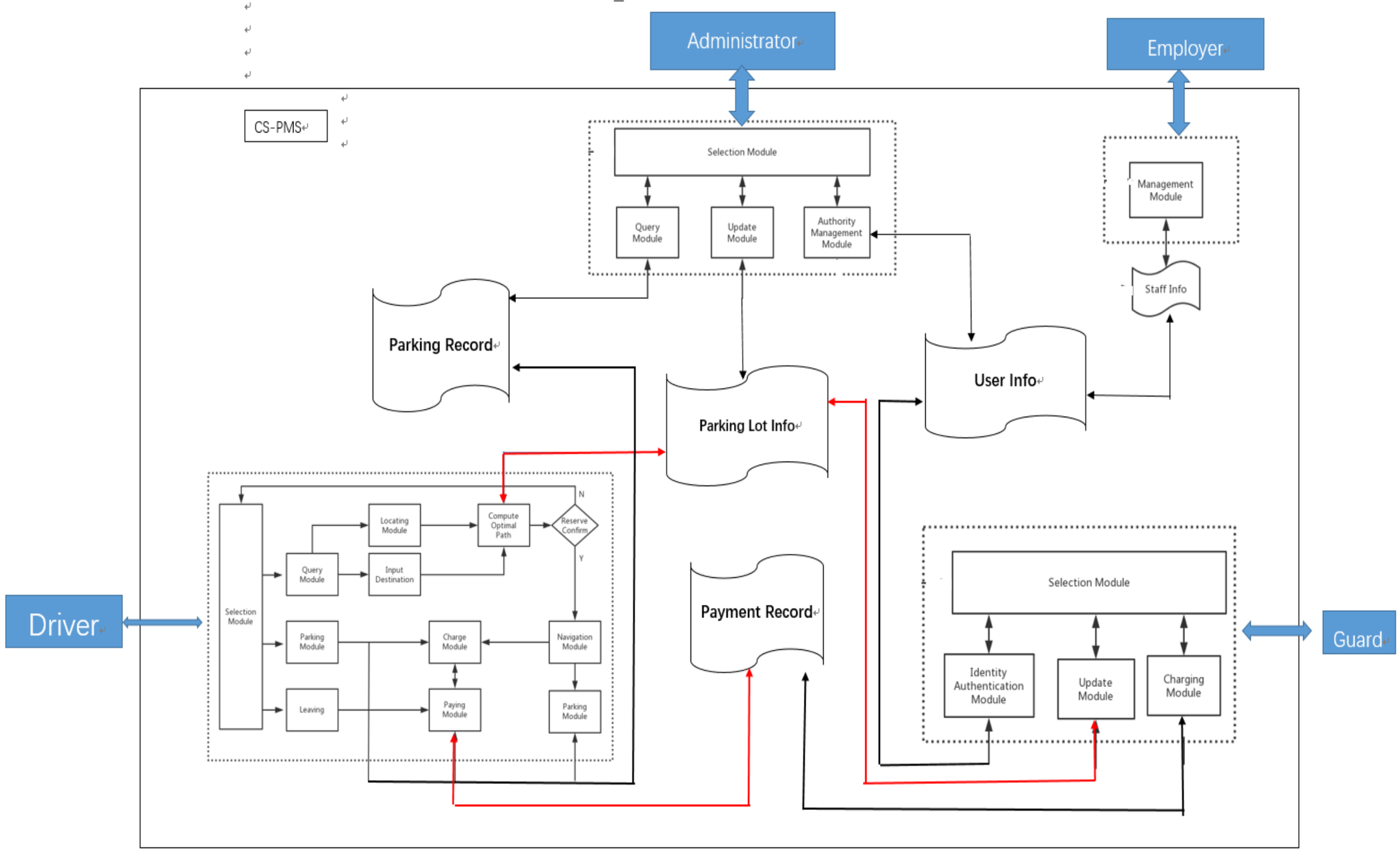
Querying: The query branch first turns on the location function to get the user's current location. After the user selects the destination address, it accesses the data in the system and selects the nearest free parking space from the destination address. It then calculates the optimal path to the parking space. At this point, the system asks the user whether to reserve this parking space. If the user confirms the reservation, the navigation function will be started and the charging will start at the same time. Finally, the user is guided to the target location by the navigation system. After that, the user's parking record is updated.

Parking: Parking Spaces can also be chosen by the user. CS-PMS only needs to start charging function and update parking record after the user confirms the location.

Leaving: Before the user leaves, he has to pay a parking fee. CS-PMS will automatically update the payment record

3.2.7 Summery: Data Flow Diagram of Level 2

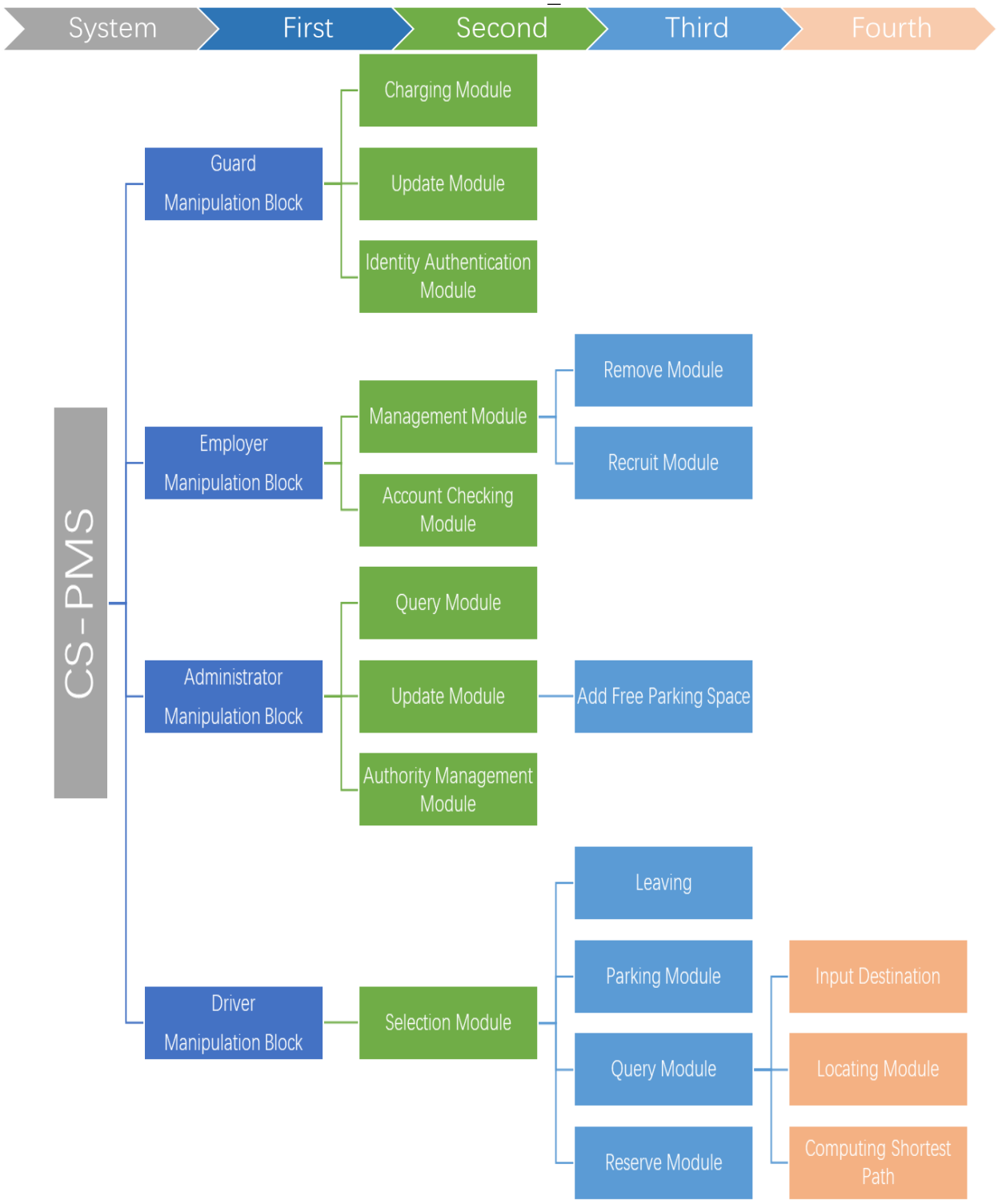
DIAGRAM_7



3.3 Functional Requirements

3.3.1 Functional Structure Diagram

DIAGRAM_8



In the previous section, the data transfer process in the system was described.

In this section, the functions of the system were summarized with the function structure diagram, and each function point was aggregated together.

3.3.2 Collecting and updating free parking space information

3.3.2.1 Introduction

The parking information of each parking lot is updated dynamically.

3.3.2.2 Inputs

Operation of guards and administrators.

3.3.2.3 Processing

Each guard has a parking lot. They collect parking information directly and upload it. The administrator is responsible for sorting out the data, removing useless information, and then updating the parking information table.

3.3.2.4 Outputs

Parking Lot Info Table.

3.3.2.5 Error Handling

When the data uploaded by the guard conflicts, it is verified and passed by the administrator.

3.3.3 Reserving parking space

3.3.3.1 Introduction

The driver queries the parking space near the target location, then he follows the system navigation to reach the destination, and finally, confirms the parking

3.3.3.2 Inputs

The system automatically locates the user's current position.
The driver enters the destination address.
The driver clicks the ok button.

3.3.3.3 Processing

The system opens the positioning function, queries the database according to the destination address, and generates the optimal path.
The system starts the navigation system to guide users to the parking space.

3.3.3.4 Outputs

Parking information of the target address attachment.
The optimal path from the current location to the target address.
Navigation information.

3.3.3.5 Error Handing

Prompt the user to change the information when the user enters an illegal destination address.
The system alerts the user when he deviates from the correct path.
When the user deviates significantly from the correct path, the system regenerates the optimal path.

3.3.4 Parking and payment

3.3.4.1 Introduction

After arriving at the destination address and confirming, the driver will pay according to the billing result.

3.3.4.2 Inputs

Driver’s operation

3.3.4.3 Processing

When the driver confirms to arrive at the destination, the system automatically updates the parking record table.
Before the driver leaves, pay according to the system prompt.

3.3.4.4 Outputs

Parking time and cost.

3.3.4.5 Error Handing

Indication that the operation failed.

3.3.5 Access control

3.3.5.1 Introduction

Administrators can view or modify user account information.
Employers appoint or remove doormen.

3.3.5.2 Inputs

The operation of administrators and employers.

3.3.5.3 Processing

The administrator views or modifies the user information table.
The employer adds or deletes a record in the employee table.

3.3.5.4 Outputs

Successful tip.

3.3.5.5 Error Handing

Failure prompted.

3.4 Use Cases

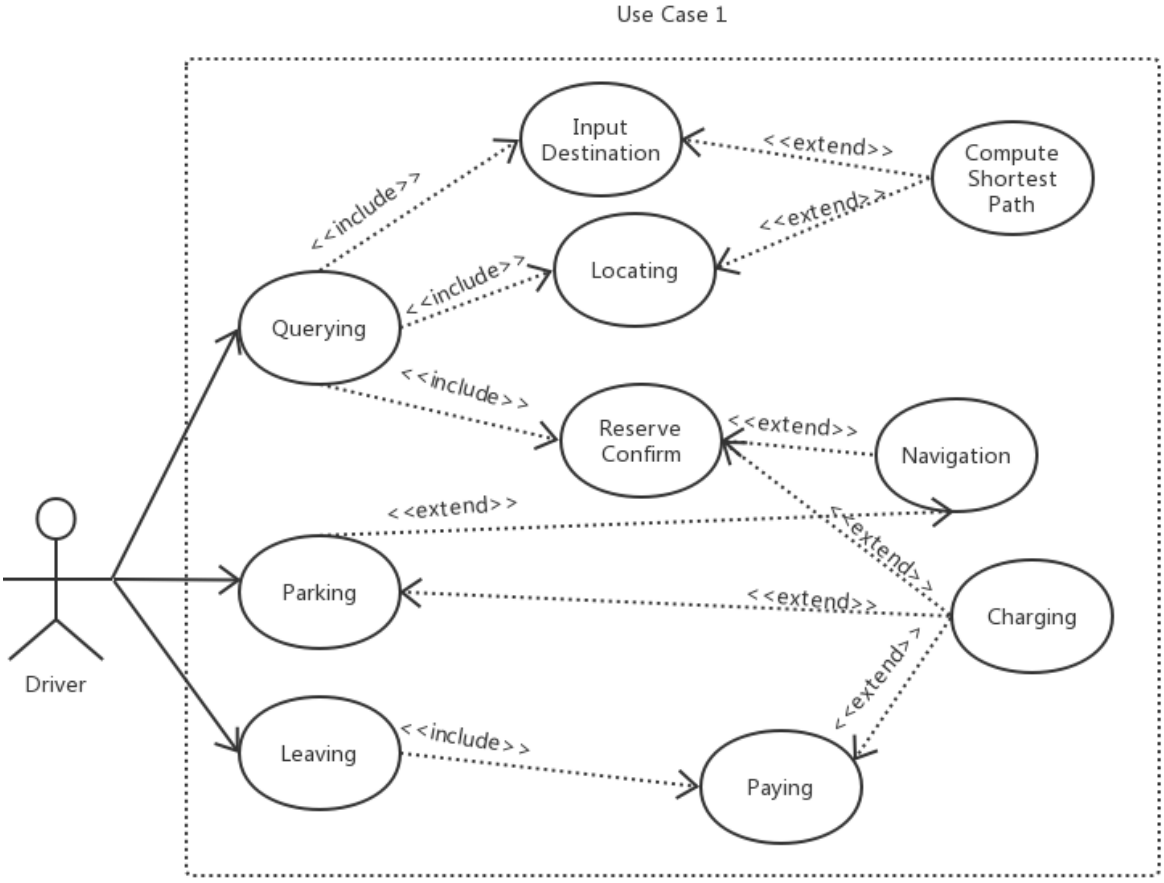
3.4.1 User Role

System administrator, driver, guard, employer

3.4.2 Use Case 1

Driver: As a participant in the system, drivers can query available parking Spaces in the city. He can also park or pay a fee.

DIAGRAM_9

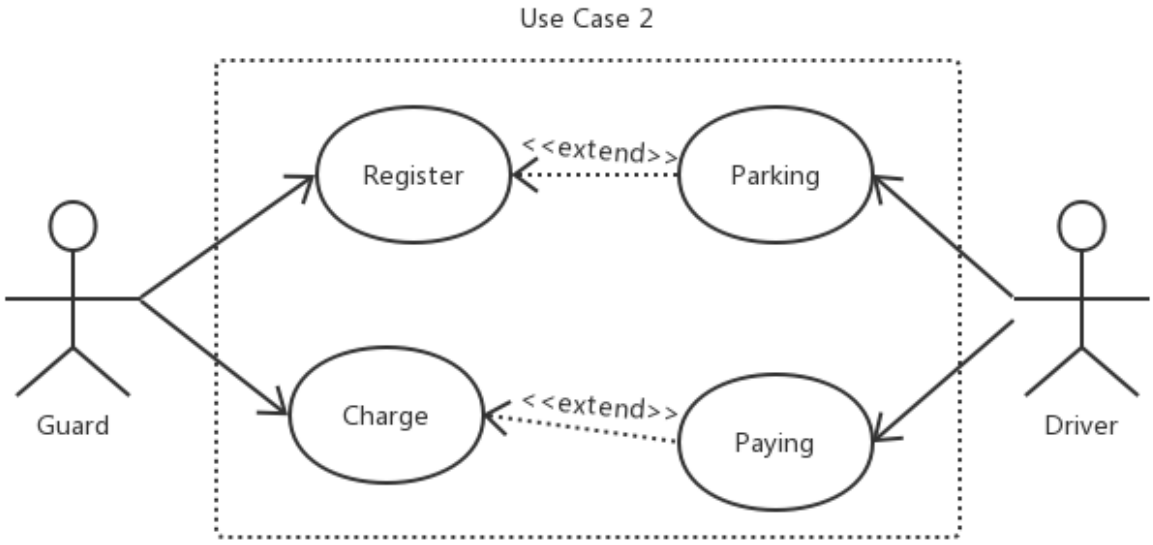


The human form corresponds to the system actor, and the ellipse represents a functional use case.
The arrow indicates that there is a relationship between the two.
Footnotes indicate the type of relationship.

3.4.3 Use Case 2

Guard: The role mainly interacts with drivers and is responsible for visitor registration and fees.

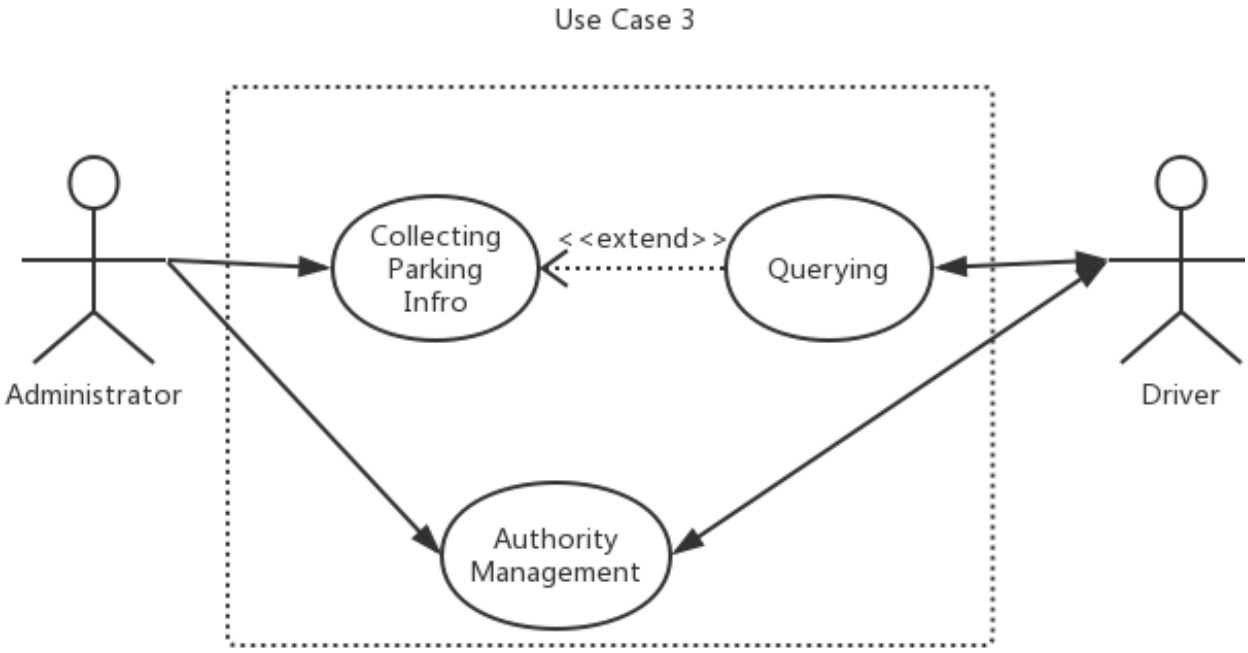
DIAGRAM_10



3.4.4 Use Case 3

Administrator: This role includes permission management and parking information collection

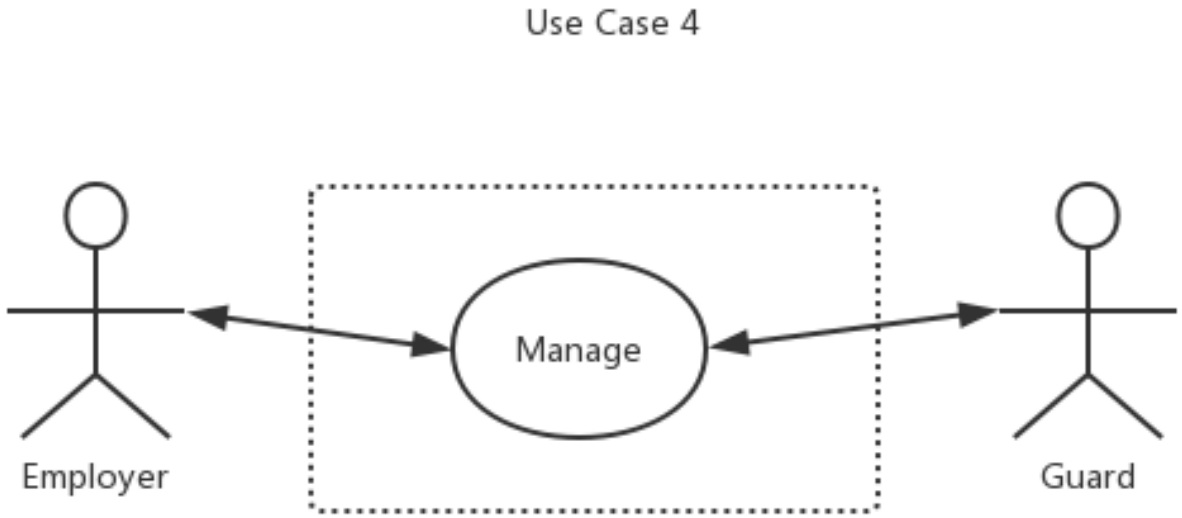
DIAGRAM_11



3.4.5 Use Case 4

Employer: His main job is to manage the doorman

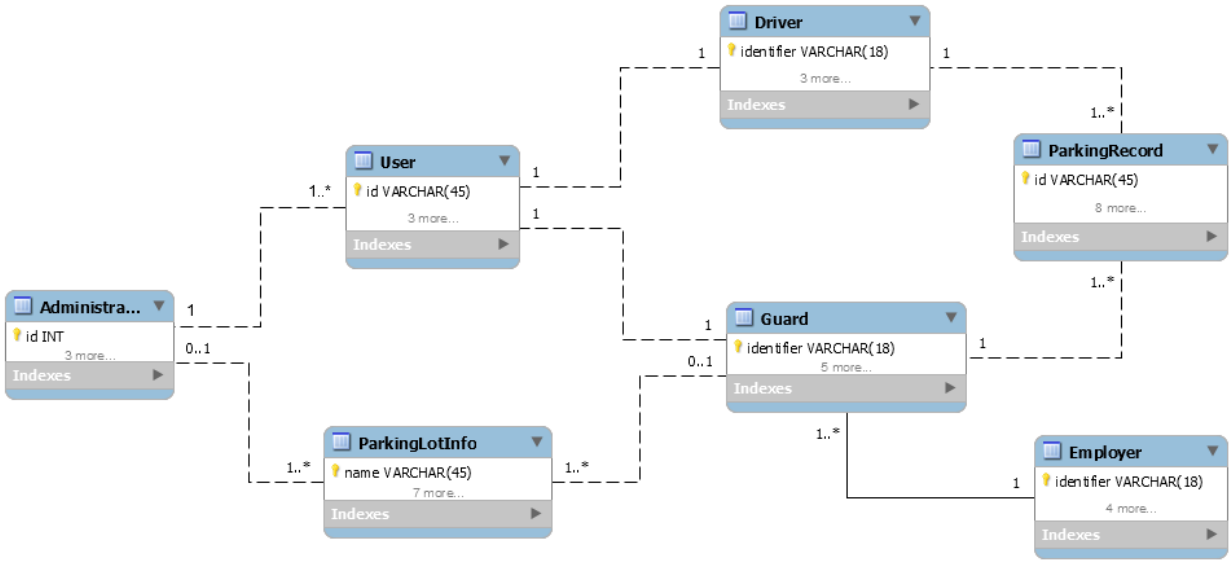
DIAGRAM_12



3.5 Classes / Objects

3.5.1 Overview

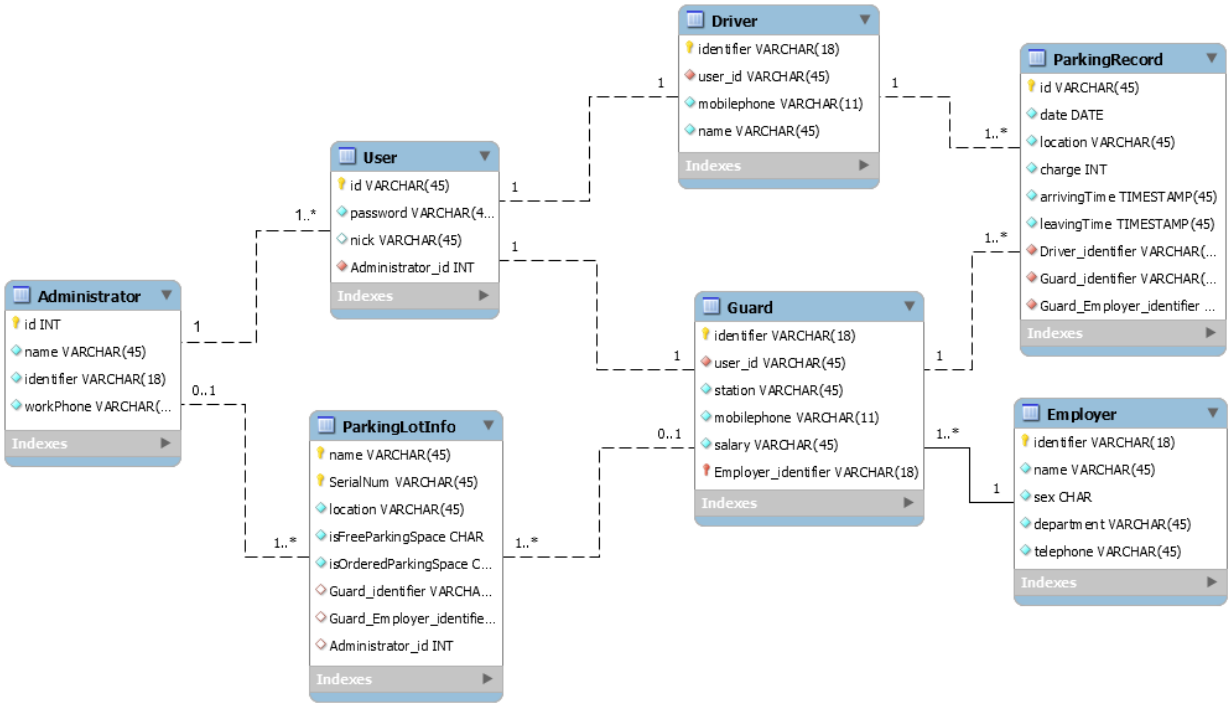
DIAGRAM_13



3.5.2 Detail

<CS-PMS>

DIAGRAM_14



4. Non-Functional Requirements

4.1 Quality Requirements

4.1.1 Performance

1. Under normal circumstances, the CS-PMS can guarantee 2 million users online at the same time. At peak times, at least 3 million users will be online at the same time.

2. When 500,000 users query at the same time, the response time does not exceed 1s. When one million users query at the same time, the delay is no more than 2s. When 2 million users simultaneously query, corresponding within 7s.

3. There is a 99 percent probability that the system supports 1 million users to use navigation function at the same time.

4. The probability of system data error during transmission is less than 1%. The system has a 100% chance of finding and fixing errors.

5. Backup system data once a week.

6. The system runtime memory shall not be less than 512GB.

7. Using a gigabit switch.

4.1.2 Reliability

1. Select the Linux operating system that does not need to be restarted for update.

2. Backup important data of the system in multiple places

3. After the normal error occurs, the system can immediately detect and recover by itself.

4. After serious errors occur, the system can stop losses in time. Then the maintenance team can solve the errors quickly.

5. The system gives warning when the administrator submits important changes.

6. The legitimacy of user input should be verified, and users' critical modifications should be warned.

7. The system should undergo multiple tests before deployment.

4.1.3 Availability

1. For the administrator and guard: providing a manual. For users: online help.

2. Users can turn on the main functions with only 3 steps.

3. The user interface should be simple and beautiful, avoiding complicated colors and redundant function keys.

4. There should be no more than 10 jumps in the management interface. A lightweight command-line control program needs to be provided to the administrator.

5. After successful or unsuccessful operation, the system can provide a clear prompt message

4.1.4 Security

1. Reliable identification system

2. Perform thorough vulnerability detection before the release of the new version

3. The source code is not visible to the client

4. According to the ALAR principle, the possible risks of the project shall be classified according to the severity of the risks.

4.1.5 Maintainability

1. Complete system development documents

2. Clear system architecture

3. Complete system iteration record

4. Detailed notes

5. There is no tight coupling between functional modules

4.1.6 Portability

1. Both android and apple clients should be provided

2. The system can be adapted to all popular Linux system versions

3. Support online update

4. No conflict with other software in the system

4.2 Engineering Requirements

4.2.1 Inverse Requirements

Details of each release iteration should be documented

Retain historical source code

Fully test and verify possible errors

Define the boundary of system behavior, that is, define what should not be done by the system

4.2.2 Design Constraints

TABLE_1

Design Constraints	Operating System	Linux/OSX
	CPU	3.60 GHz Intel Core i9
	Memory	16GB
	Switch	1000Mbps
	DataBase	MySQL/SQLite
	Language	Python/C++
	Builder	PyCharm/GCC
	Frame	Djange/Bootstrap
	Server	Nginx
	Client	Android/IOS/HTML5
	Version Control and Collaborative development	Git
	Code quality inspection	SonarQube

4.2.3 Logical Database Requirements

TABLE_2

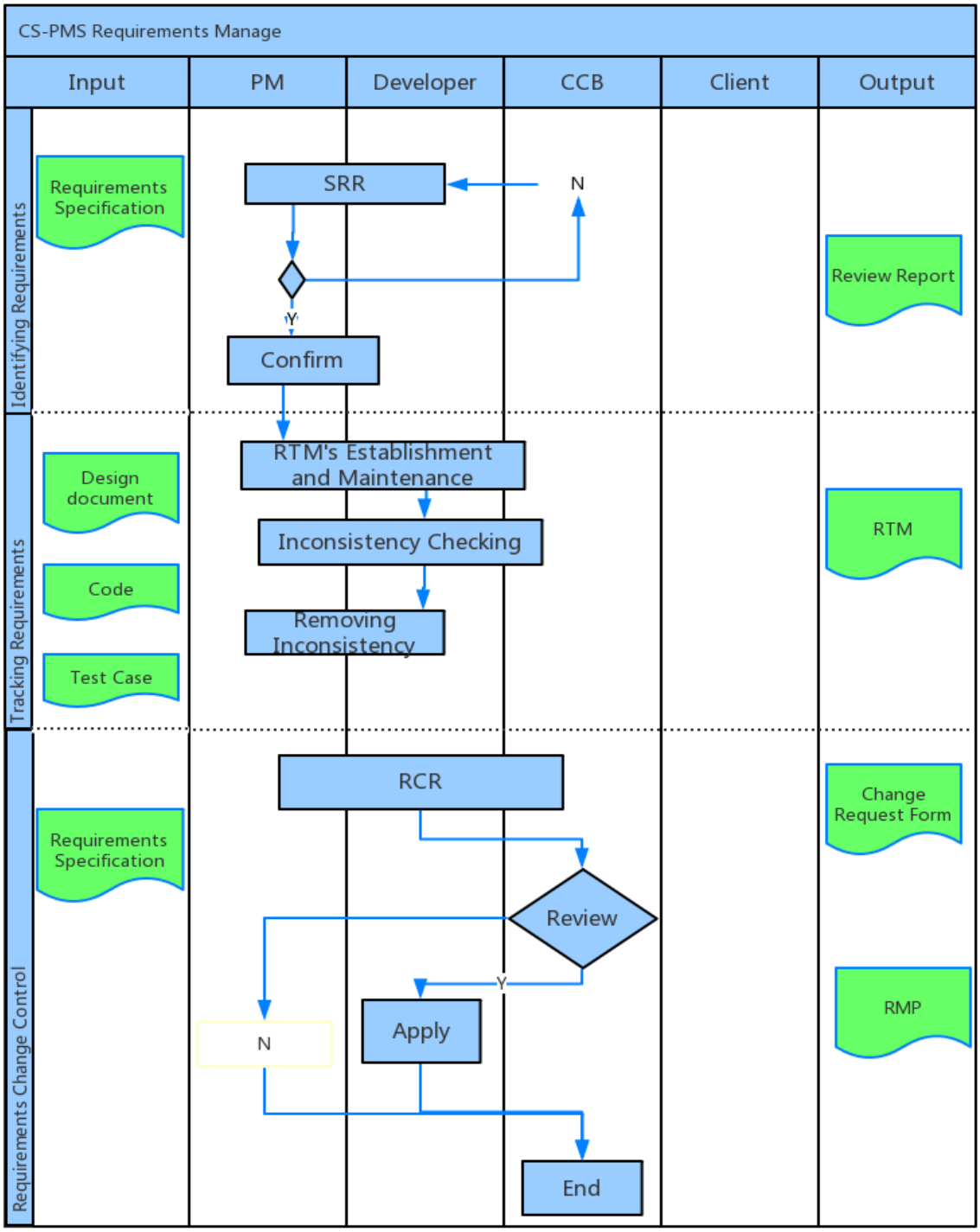
MySQL DBMS		
Normal Data Format	Id	Varchar,45
	Password	Varchar,45
	Name	Varchar,45
	Mobilephone	Varchar,11
	Identity Card	Varchar,18
	Sex	Char,1
	Parking Lot Location	Varchar,45
	Arriving Time and Leaving Time	Date
	Bill	Float,4,3
Storage Capacity	100GB, Updating Once a Year	
Data Retention	Account information: if the user has not been online for one year, the data will be cleared	
	Key data: such as administrator data, system iteration data is always retained	
	Runtime data: parking space information is always kept, but updated in real time	
	Logging: updated once a month	
Data Integrity	Following DBMS rules	

<CS-PMS>

5. Change Management Process

5.1 Swimlane diagram

DIAGRAM_15



5.2 Illustration

The figure shows the requirements change process of CSPMS with the help of swimlane diagram. Each swimlane is a role. As shown in the figure, the dotted line separates the three phases of requirements management. The six lanes represent six important roles. Rectangular boxes are used to represent activities. A diamond box represents a selection. The arrows indicate the process order. In a swimlane called input and output, the relevant files are arranged in the order in which they are generated.

-PM: Project Manager. Throughout the development process, the main tasks of the project manager include requirements tracking, change and management. It is to prevent project failures due to requirements changes

-CCB: It can be a team or multiple different groups that decide which of the proposed requirements changes or new product features to apply. A typical change control board would also decide which versions to correct for which errors.

-SRR system requirements review. It's done by CCB and PM.

-RTM Requirement tracking matrix.

-RCR Requirement change request

-RMP Requirements change control report

A. Appendices

A.1 Quality Analysis

The following are 21 indicators of quality evaluation. Meanwhile, the quality index of this document is calculated. They're represented by Q1 through Q21. Using the data normalization method to ensure that their values range from 1 to 10.

1.Unambiguous:

$$Q_1 = N_u / N_r \qquad 1$$

Where Nu represents the number of requirements clauses with ambiguity and Nr represents the total number of requirements clauses.

Q1=10

2. complete

Within the scope of this project

• a) contains all realizable scenarios and responses to their input data;

• b) all page Numbers, all charts are numbered, all terms are defined, all units of measurement are provided, and all referenced materials are referenced.

Q2=10

3.correct

$$Q_3 = N_c/(N_c + N_{NV}) = N_c/N_r \quad (2)$$

Among them, Nc is the exact number of terms confirmed, Nnv is the unconfirmed requirement term, Nc is the total number of terms required.

Q3=10

4. understandable

Intelligibility is a relative term that increases the intelligibility of developers and testers while reducing the unintelligibility of customers and users.

Therefore, this document uses a formal language to describe the SRS for ease of understanding by developers and testers.

At the same time, each graphical model has a natural language description at the bottom.

A person with no development experience can understand 80% of the requirements of the system just by looking at the diagram.

If he reads the auxiliary instructions, he can understand 99% of the requirements of the system.

$$Q_4 = N_m/N_r \quad (3)$$

Q4=8

5. Verifiable

5.1 Analysis

99% of the requirements in this document are verifiable.

Unverifiable requirements are judged on the basis of human subjectivity. Including: beautiful software interface, timely response.

5.2 A measurement model——SRS-RV01

The verifiability of requirements clauses is mainly related to the cost and cycle. The higher the cost and the longer the test cycle, the more difficult it is for requirements to be verified.

Based on this idea, a measurement model was designed to measure the verifiable percentage of requirements clauses in SRS: srs-rv01It

The following is the definition of the model parameters,

Nu	Number of verifiable requirements clauses
Nr	Total number of requirements clauses
C(Ni)	The cost of verifying the requirement clause Ni, 0< i <=Nr
T(Ni)	The time required to verify requirement clause Ni, 0< i <=Nr
Sum	Verify the total time and total cost of all requirements terms
a	Weight of expenses
b	Weight of time
p	A threshold for verifiability

5.2.1 Sum Formula

At first, the model uses the formula 4 to calculate the total time and total cost of all requirements terms.

$$\text{sum} = \text{a} * \sum_{i=1}^{N_r} C(N_i) + \text{b} * \sum_{i=1}^{N_r} T(N_i) \quad 4$$

5.2.2 Calculate the number of verifiable requirements,

Then the algorithm 5.2 is applied to calculate the number of verifiable requirements.

Algorithm 5.2

```
i=1
while i <= Nr:
  pi = (a*C(Ni)+b*T(Ni)) / Sum
  if pi < p:
    Nu=Nu+1
  i = i+1
```

5.2.3 verifiable

Finally, using the formula 5 to calculate the percentage of requirements that are verifiable.

$$Q_5 = N_u / N_r \quad 5$$

5.2.4 calculate Q5

using the model SRS-RV01, taking the a and b are both 1/2, calculating the Q5.

Q5 = 8

6.Internally Consistent

After verification in 5 section, all functions of the system are consistent, and there is no redundancy.

Q6=10

7.Externally Consistent

In current stage, this system only completes SRS documents, and there is no requirement clause inconsistent with other documents of the system. Therefore, the subsequent development of the system should follow the definition and requirements of this document. Use the methods described in section 1 of this document for requirements management and change.

Q7=10

8.Achievable

A good way to evaluate the feasibility of a system is to first construct a prototype of the system, then estimate the cost and duration of the project, and evaluate whether it can be implemented within the budget and duration.

Q8=10

9. Concise

This document avoids redundant descriptions and complex terminology. Using graphical representations, tables, and formulas to add simplicity to documents.

$$Q_9 = 1/(Size + 1) * 100 \quad 6$$

Q9=4

10. Design-Independent

The external demand of this system is that the government needs to provide parking information of highways and sidewalks. There is a weak correlation between a solution that implements external requirements and a solution that implements internal requirements of the system. That is, the system has the design independence.

Q10=10

11.Traceable

Give a list of requirements as defined in this document, and all requirements can be tracked.

Q11 = 10

12.Modifiable

This document has a detailed directory and chart index.

Q12=10

13.Annotated by Relative Importance

The most important part is the functional requirements of the system, which is arranged in the third part.

The non-functional requirements and requirements change management sections of the system are less important and are arranged respectively in sections 4 and 5.

Q13 = 10

14.Annotated by Relative Stability

All functional requirements are key to the implementation of the system functions. They cannot be changed. In a way, all non-functional requirements can be changed.
Q14 = 10

15.Annotated by Version

The version change information is on the second page of the SRS document
Q15 = 10

16.Not Redundant

To improve the readability of the document, this document describes the functional requirements of the system in four ways. Including: data flow diagram, functional structure diagram, use case diagram, class diagram.
Q16 = 10

17.Precise

The requirements in this document have been quantified. For example, when 500,000 users query at the same time, the response time does not exceed 1s.
Q17 = 10

18.Reusable

This document uses formal models to express requirements that can be reused by subsequent requirements.
Q18 = 10

19.Traced

Sources of requirements for this document include but are not limited to: a parking lot management app already on the market, the Baidu maps application, the IEEE Software Engineering Standards Committee, the book named by software engineering.
Q19 = 10

20.Organized

The structure of this document is very clear, in order of general description, functional requirements, non-functional requirements, and requirements change management. Each chapter is divided into several sections according to the actual situation. Readers can locate the information they need based on the directory.
Q20 = 10

21.Cross-Referenced

Chapter 1 shows the organization of the document. The second chapter gives a brief introduction of the remaining chapters. The content of chapter 4 depends on the content of chapter 3.
Q21=10

A.2 Conclusion

According to the corresponding evaluation criteria. The score ranges from 1 to 10.
Q1=10
Q2=10
Q3=10
Q4=8
Q5 = 8
Q6=10
Q7=10
Q8=10
Q9=4
Q10=10
Q11= 10
Q12=10
Q13=10
Q14=10
Q15=10
Q16=10
Q17=10
Q18=10
Q19=10
Q20=10
Q21=10
The total score is 210
The total score of this document is 200