Taxi Taking Management System

Architecture Design

Version 2.0

Dec.20, 2019

Haichao Zhang

School of Software Engineering

Student ID: 2017211904

Class Number: 2017211502

Introduction to Software Engineering

Fall 2019

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 10/12/2019 | Version 1.0 | Haichao Zhang | First Revision |
| 18/12/2019 | Version 2.0 | Haichao Zhang | Second Revision |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Content**

[**Revision History** ii](#_Toc27734080)

[**Document Approval** ii](#_Toc27734081)

[**1. Introduction** 1](#_Toc27734082)

[**1.1 Purpose** 1](#_Toc27734083)

[**1.2 Scope** 1](#_Toc27734084)

[**1.3 Definitions, Acronyms, and Abbreviations** 2](#_Toc27734085)

[**1.4 References** 3](#_Toc27734086)

[**1.5 Overview** 3](#_Toc27734087)

[**2. Architectural Representation** 3](#_Toc27734088)

[**3. Architecture Requirements** 4](#_Toc27734089)

[**3.1 Product Perspective** 4](#_Toc27734090)

[**3.2 Product Functions** 4](#_Toc27734091)

[**3.3 User Characteristics** 6](#_Toc27734092)

[**3.4 Architecture Perspective** 6](#_Toc27734093)

[**3.5 General Constraints** 8](#_Toc27734094)

[**3.6 Assumptions and Dependencies** 8](#_Toc27734095)

[**3.7 Nonfunctional Requirements** 8](#_Toc27734096)

[**3.7.1 Performance** 8](#_Toc27734097)

[**3.7.2 Reliability** 9](#_Toc27734098)

[**3.7.3 Availability** 9](#_Toc27734099)

[**3.7.4 Security** 9](#_Toc27734100)

[**3.7.5 Maintainability** 9](#_Toc27734101)

[**3.7.6 Portability** 9](#_Toc27734102)

[**3.8 Critical Standard** 9](#_Toc27734103)

[**4. Architecture goals and constraints** 10](#_Toc27734104)

[**4.1 Architecture goals** 10](#_Toc27734105)

[**4.2 Architecture Constraints** 10](#_Toc27734106)

[**4.2.1 Basic constraints** 10](#_Toc27734107)

[**4.2.2 System Constraints** 10](#_Toc27734108)

[**4.2.3 Engineering design constraints** 11](#_Toc27734109)

[**5. Solution** 11](#_Toc27734110)

[**5.1 Related architecture patterns** 11](#_Toc27734111)

[**5.1.1 C/S pattern** 11](#_Toc27734112)

[**5.1.2 B/S pattern** 12](#_Toc27734113)

[**5.2 Overview of Architecture** 13](#_Toc27734114)

[**5.3 Structural View** 13](#_Toc27734115)

[**5.3.1 Conceptual Architecture** 14](#_Toc27734116)

[**5.3.2 Modular Architecture** 15](#_Toc27734117)

[**5.3.3 Runtime Architecture** 16](#_Toc27734118)

[**6. Use-Case View** 17](#_Toc27734119)

[**6.1 Architecturally-Significant Use Cases** 18](#_Toc27734120)

[**6.2 User Role** 19](#_Toc27734121)

[**6.2.1 User Role #1** 19](#_Toc27734122)

[**6.2.2 User Role #2** 19](#_Toc27734123)

[**6.2.3 User Role #3** 19](#_Toc27734124)

[**6.2.4 User Role #4** 20](#_Toc27734125)

[**6.3 Use Case Description** 20](#_Toc27734126)

[**6.3.1 Send an order request** 20](#_Toc27734127)

[**6.3.2 Process the order** 20](#_Toc27734128)

[**6.3.3 Arrange the taxi** 21](#_Toc27734129)

[**6.3.4 Record the information** 21](#_Toc27734130)

[**6.3.5 Evaluate the service** 22](#_Toc27734131)

[**6.3.6 Arrange Archive** 22](#_Toc27734132)

[**6.3.7 Make payment** 23](#_Toc27734133)

[**7. Logical View** 23](#_Toc27734134)

[**7.1 Architecture Overview – Package and Subsystem Layering** 23](#_Toc27734135)

[**7.1.1 User Interface Layer Package** 23](#_Toc27734136)

[**7.1.2 Domain Layer Package** 24](#_Toc27734137)

[**7.1.3 Technical Services Layer Package** 24](#_Toc27734138)

[**8. Process View** 25](#_Toc27734139)

[**8.1 Interaction and Configuration** 26](#_Toc27734140)

[**8.2 Classes** 26](#_Toc27734141)

[**8.2.2 Class#1: Taxi Taker** 26](#_Toc27734142)

[**8.2.3 Class#2: Driver** 26](#_Toc27734143)

[**8.2.4 Class#3: System Administrator** 26](#_Toc27734144)

[**8.2.5 Class#4: Business monitor** 27](#_Toc27734145)

[**8.2.6 Class#5: Database** 27](#_Toc27734146)

[**8.2.7 Class#6: Order** 27](#_Toc27734147)

[**9. Deployment View** 27](#_Toc27734148)

[**10. Development View** 29](#_Toc27734149)

[**11. Quality** 29](#_Toc27734150)

[**11.1 Quality Requirements** 29](#_Toc27734151)

[**11.2 Scenario Analysis** 30](#_Toc27734152)

[**11.2.1 Use-case Scenarios** 30](#_Toc27734153)

[**11.2.2 Growth Scenarios** 30](#_Toc27734154)

[**11.2.3 Exploratory Scenarios** 30](#_Toc27734155)

[**11.3 Prototype Analysis** 30](#_Toc27734156)

[**11.4 Risk** 31](#_Toc27734157)

[**12. Extensibility** 32](#_Toc27734158)

[**13. Modifiability** 32](#_Toc27734159)

**1. Introduction**

This is an architecture design document of the Taxi Taking Management System (TTMS). In the following sections, this document will mainly show you the architectural of representation, architectural of goals and constraints and views from different angles (use-case view, logical view, process view and deployment view). Size, performance and quality of the system will also be discussed at the end of the document. Several structural diagrams are used in the report to illustrate the architecture and organization of TTMS clearly.

**1.1 Purpose**

This document is a document written after a comprehensive analysis of the functional and non-functional requirements of the online taxi software system and the design of the software architecture in various aspects. The content of this document is comprehensive. The use case view and logic show readers the overall structure of the system, including logical design, physical structure, and analysis of the system's architectural requirements, including constraints, system size, performance indicators, quality requirements, etc., and use Multiple views describe the solution of architecture design and analysis modeling, and finally perform the quality analysis of the architecture.  
This document serves as a reference document for product establishment and product development. It specifies the detailed functional requirements of each user, the composition and contact of system functional blocks, deployment and hardware requirements, etc., which is helpful to improve the visibility during the software development process and facilitate the software development process. Control and management. This architecture design document is the basis for software project design and development, and the main basis for writing test cases and system testing. It plays a guiding role in the subsequent stages of development. At the same time, this document can also be used as a basis for software project communication between software users, software customers, developers and other parties.

This software architecture document is written for the Taxi Taking Management System (TTMS). It provides a comprehensive architectural overview of TTMS. Different architectural views are utilized in the following sections to illustrate the system from various aspects. It is intended to capture and convey the significant architectural decisions which have been made on TTMS. The purpose of the document is to give a comprehensive and accurate introduction to architecture of TTMS and enhance related professionals’ efficiency.

**1.2 Scope**

The name of the software product is Taxi Taking Management Software (TTMS). This software architecture document provides an architectural overview of TTMS.

The system provides following functions for different users:

* The system enables taxi takers to send an order request to the available nearby taxi driver with the mode and destination those takers set. The taking mode includes assigned mode and competitive mode. When there are no available taxis nearby, takers are allowed to increase their taxi fare in a special form called “red packet” to schedule the distant taxi to serve them. Takers will make their payment based the distance they travel and get their receipt after payment. The taxi takers are also expected to evaluate drivers’ services.
* For the orders in assigned mode, the taxi driver is able to assigned to the nearest takers automatically; for the orders in competitive mode, the taxi drivers within a certain range are allowed to give their response to an order request, the nearest responder who responds within one minute will get the order. When they finish their service, they will gain their money and evaluation.
* Business monitor can gain the evaluations of drivers and inform system administrator to process the database.
* System administrator has the right to check the drivers’ archive and modify the database. They are also in charge of processing the request and arrange the cars.
* When the driver accepts the order, the state of taxi will be recorded in the database.
* The evaluation from takers will be recorded in the database.
* Product design background: In some large cities, people's demand for taxis is increasing, and in most cases, it is difficult for passengers to find an empty taxi at the intersection. At the same time, it is difficult for taxis to find passengers who need to take a car. Therefore, it is also difficult for passengers to take a taxi and the driver ’s empty rate is high.
* Document content: This architecture design document outlines the main functions of the online taxi software system, elaborates the overall structure of the system, explains the overall design strategy of the system, gives a solution for architecture design, and analyzes and models Finally, the quality analysis and evaluation of the architecture are performed.
* Application: This software architecture design document is suitable for the overall application structure of the city's shared parking management system. The purpose is to meet the system's quality and reliability requirements, as well as future maintenance, operation, and upgrade requirements for online taxi software systems .

**1.3 Definitions, Acronyms, and Abbreviations**

This section shows all the definitions, Acronyms and abbreviations in TTMS.

|  |  |
| --- | --- |
| Term | Definition |
| TTMS | Taxi Taking Management System |
| State of Taxi | Represent the taxi is idle or working and the details of certain order the taxi serving. |
| Taxi nearby | Those taxis which are within 3 km to taker. |
| Distant taxi | Those taxis whose distance to taker are greater than 3km. |
| C/S pattern | Client/Server pattern. B / S structure (Browser / Server, browser / server mode) is a network structure mode after the rise of the WEB. The WEB browser is the main application software of the client. This mode unifies the client, centralizes the core parts of the system's function implementation on the server, and simplifies system development, maintenance, and use. |
| B/S pattern | Browser/Server pattern. The three-layer B / S architecture is based on the B / S structure. A layer of structure is added to the data management layer (Server) and user interface layer (Client), which is called middleware. Three floors. |

**1.4 References**

[1] 王安生. 软件工程化. 北京: 清华大学出版社, 2014

**1.5 Overview**

The rest of the document is organized as follows:

* Architectural Representation

This chapter tells you the how the architecture is presented.

* Architecture Requirements

This chapter give a brief introduction of the TTMS system, including product perspective, product functions, user characteristics, general constraints, dependency, nonfunctional requirements and critical standard.

* Solution

Introduce some related and frequently-used architecture and describe the overview of architecture with structural view.

* Describe the architecture from “4+1” views.
* Quality analysis and prototype analysis.
* The discussion about Extensibility and Modifiability.

.

**2. Architectural Representation**

This document describes the architecture in the form of text and Unified Modeling Language (UML) model developed using Process On. The architecture is going to be presented as a series of views based on the 4+1 view model which is published by Philippe Kruchten: use case view, logical view, process view, deployment view and Development View.

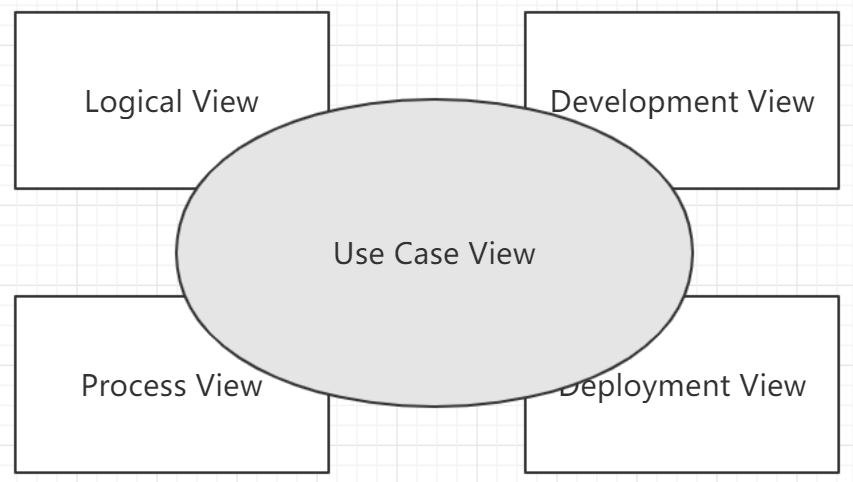


Figure 4+1 view model

**3. Architecture Requirements**

**3.1 Product Perspective**

The purpose of TTMS is to make people take taxi more quickly and conveniently and improve drivers’ efficiency. Different with other management system, TTMS provides several taking mode and evaluation subsystem to make sure the efficiency and quality of drivers’ services.

The motivation of the project is: in some large cities, people's demand for taxis is increasing, and in most cases, it is difficult for passengers to find an empty taxi at the intersection. At the same time, it is difficult for taxis to find passengers who need to take a car. Therefore, it is also difficult for passengers to take a taxi and the driver ’s empty rate is high. The rapid development of mobile Internet and smart terminals provides opportunities to solve this problem by using taxi software.

By using this system, passengers can instantly find an unloaded driver online and reach the destination in time. Drivers can use this system to find all nearby customers who need to ride and provide accurate route navigation. The system can simultaneously solve the problem of difficult taxi rides and high empty rate for drivers, so that the issue of taxis is no longer a problem.

**3.2 Product Functions**

TTMS allows taxi takers, drivers, system administrator, business monitor and system interact with each other. Taxi takers send an order request. Then system administrator will arrange the cars based on takers’ request. Drivers’ shall serve their customers as soon as they accept the order. After the services, takers are supposed to pay for the driver and give their evaluation. Business monitor is in charge for check the services of drivers. Once there is something wrong with driver, monitor can inform to system administrator to modify the database.

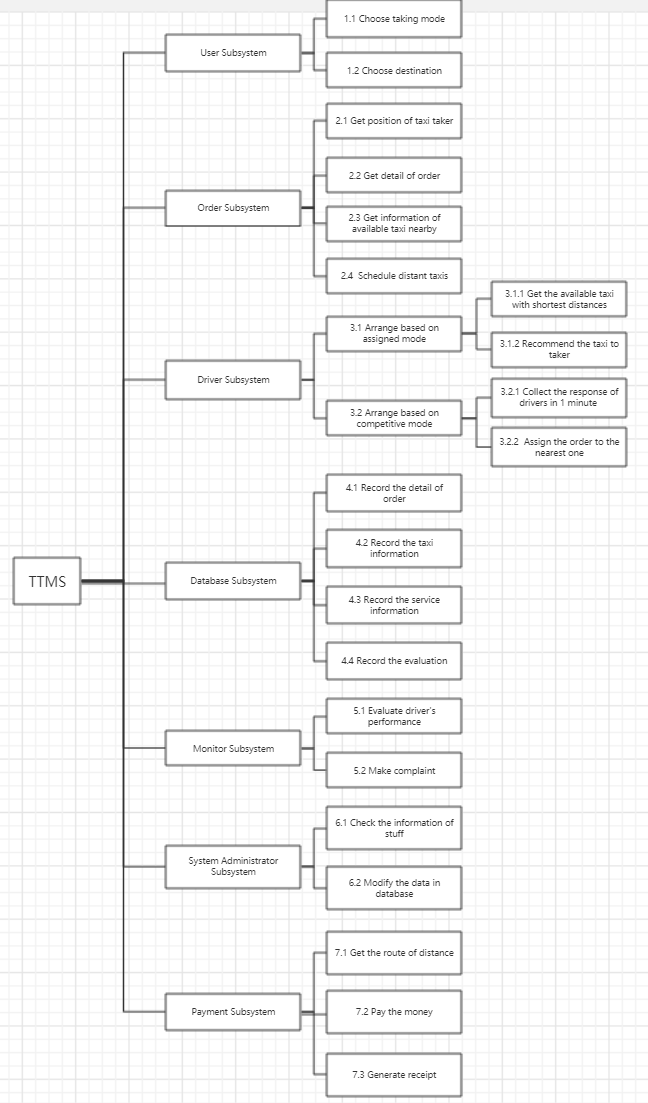


Figure System Function Diagram

**3.3 User Characteristics**

* Taxi taker: the applicant of service. Use the software to send request and set the mode and destination.
* Passengers can view and modify their personal information.

When taxiing, the system can display map information around the rider, as well as the current location of the rider and the location information of the surrounding driver.

* Passengers can choose the boarding position, service type, destination information, etc.
* After entering the itinerary information, the system can display the amount of travel expenses expected to be paid.
* When a driver grabs a ticket, the system can display the driver's basic information and location information.
* After the driver arrives, the passenger can see the vehicle navigation information.
* Passengers can ask customer service if they have any questions.
* Driver: the provider of service. Serve the taker when they accept the order.
* Drivers can view and modify their personal information in the system
* The system can display map information around the driver.
* The system can display nearby order information. Drivers can use this system to select the order they want to pick up and perform the order grab operation.
* After receiving passengers, the system can provide navigation information for drivers and provide route planning services.
* After sending the rider to the destination, the system can calculate the amount generated by this order and require the user to complete the payment within 24 hours.
* Drivers can ask customer service if they have any difficult questions.
* System administrator: When the customers’ requests arrive, schedule the available cars to serve customer as soon as possible. System administrator is also responsible for the arrangement of database.
* When there is a transaction error in an order, you can view the order and manage it.
* Manage user authentication information.
* Business monitor: monitor the service condition of drivers. For the abnormal condition, inform system administrator to modify the database.

**3.4 Architecture Perspective**

The architecture of the system is demonstrated from multiple perspectives (use case perspective, logical perspective, process perspective, deployment perspective). Analyze the context and business goals of the system. In Figure 2-1, the relationship between several perspectives is more vividly described. There is no separate implementation of each perspective in this document, only the corresponding model description is given through the UML modeling language

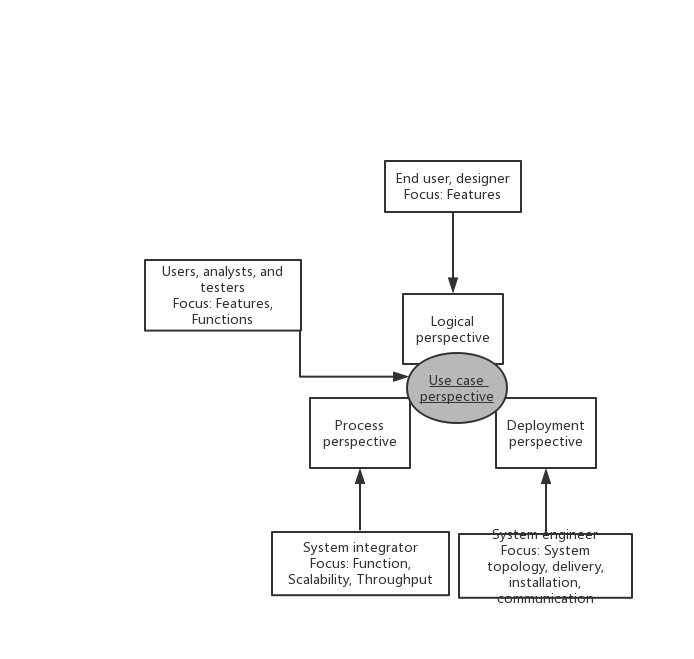
. 

Figure System architecture overview

In this model, the respective requirements of stakeholders are as follows:

The end users and designers of the system require that the system can implement the basic functions of the requirements analysis phase. For OSTM, the system is required to meet all the requirements defined by users in the requirements analysis phase for passengers, drivers, system administrators, etc., including their functional requirements. And non-functional requirements.

Users, analysts, and testers require the system to meet the system characteristics envisaged in the requirements analysis phase. The system is required to operate correctly and reliably, with complete data, strong system operability, and a good user experience.

The system integrator requires the system to meet the performance indicators proposed in the requirements analysis phase, such as the system's time performance and space performance requirements. It is required that the various modules and levels of the system are as loosely coupled as possible to facilitate the integration between various components.

The programmer requires that each part of the system software has a reasonable structure, which is convenient for modification, upgrade, testing, and maintenance. It also requires that the system interface is designed reasonably, which facilitates the reuse of certain module function codes, and is used as a template framework for the secondary development of other systems .

The system engineer requires the system to be completed and delivered within the specified time, and the project leader should be trained accordingly when the system is delivered so that it can use the system correctly.

**3.5 General Constraints**

* System uses the interface of database
* System transmits data in an encrypted way
* System supports multithreading concurrency
* Web server is deployed in Linux operating system to maintain stability.
* When developing this software, it must be implemented in strict accordance with relevant laws and policies.
* System has access to the interfaces of QQ, Wechat, and GaoDe map.
* Once the driver is working, he cannot compete for the new order or receive the assignment from system administrator.

**3.6 Assumptions and Dependencies**

* The estimate of amounts of customers is about 1 billion and the amounts of drivers may be 1million.
* It is assumed that the most concurrency is no more than 10000 per server; that is to say, no more than 10000 taxi takers can send their order request simultaneously.
* The drivers are well-trained and well-educated.
* The system is designed based on this document strictly.
* The system has been fully tested in all the perspective before formal release.

**3.7 Nonfunctional Requirements**

**3.7.1 Performance**

* The taxi taker’s order request shall be able to be accepted by server within 1 second.
* In assigned mode, the system administrator shall be able to select the proper driver and assign it to the current taxi taker within 2 seconds.
* The receipt should be generated within 2 seconds.
* The system shall have memories which supports ten-year storage of information and three-year of log.
* With a probability of higher than 97%, the TTMS can support 10000 users online or more.
* With a probability of 100%, the system can support more than 5000 users’ simultaneous processing.
* With a probability of higher than 99%, the system can support more than 8000 users’ simultaneous processing requests. And the delay shall not exceed three seconds.

**3.7.2 Reliability**

* 98% of transaction should be handled correctly within 5 seconds.
* The complaints made by users should be responded within 3 days.
* When system errors occur, the system shall be able to recover within 15 minutes.
* The system can check its work state once per day, and report the consequence to the system administrator.
* At 98% of the time, the TTMS shall be able to run without problems.

**3.7.3 Availability**

* System should be able to set the recover points. Recover points can help the system recover from the exceptional state.
* The software system should be compatible to multiple hardware devices.
* The software system should have user-friendly user interface.

**3.7.4 Security**

* The personal information of our users should be transmitted through an encrypted way.
* All the users’ account should be and only be created by system administrator.
* Periodic check should be supported.

**3.7.5 Maintainability**

* System database shall be updated twice a year.
* Each maintenance should be done within 3 days.
* The components in the TTMS should be integrated with high cohesion and low coupling.

**3.7.6 Portability**

* System shall be able to run in the environment below jdk 1.8.0.
* System shall be able to support online update.
* The software system should be compatible to multiple platform.

**3.8 Critical Standard**

1. Response time:

* When 30000 users send their requests at the same time, the response time of the system shall not be greater than 9s;
* When 10000 users send their requests at the same time, the response time of the system shall not be greater than 7s;
* When 5000 users send their requests at the same time, the response time of the system shall not be greater than 5s.

2. Error rate:

* In the case of 100000 service requests, the number of errors shall not be more than 3.
* In the case of 1000000 service requests, the number of errors shall not be more than 60.

3. Stability:

* In the case of 10000 concurrent users and 5 consecutive hours of work, the system will not stop service.

**4. Architecture goals and constraints**

**4.1 Architecture goals**

By designing the software architecture described in this document and implementing it during the system implementation phase, the system can implement all the requirements defined during the requirements analysis phase and meet the requirements of non-functional requirements, such as reliability, portability, Performance and other aspects.

**4.2 Architecture Constraints**

By designing the software architecture described in this document and implementing it during the system implementation phase, the system can implement all the requirements defined during the requirements analysis phase and meet the requirements of non-functional requirements, such as reliability, portability, Performance and other aspects.

**4.2.1 Basic constraints**

|  |  |
| --- | --- |
| Basic elements | 主要约束 |
| Legal and policy constraints | During the software development process, laws and regulations must be strictly observed. |
| Equipment constraints | Software must be able to run on Android and IOS |
| Basic scope of the project | Complete project development and testing |
| Project development time | 6 months |
| Project development cost | 10000~20000元 |

Table Basic constraints

**4.2.2 System Constraints**

During the architecture design process, the following key system constraints need to be met:

1. All users need to perform corresponding authentication, and the system can only be used after successful authentication.

2. For different functions, different users have different usage rights. When users use a function, they must first confirm their identity.

3. This system must be able to protect all the data of each user and ensure that it will not be leaked due to external attacks or unauthorized access.

4. The payment module in the system must be connected with third-party payment software through an interface to support the payment function of the system. This interface will be defined in detail in the interface specification later.

5. The map navigation module in the system must be able to connect with the third-party map navigation software through the interface to support access to the map of the user's current location in the system, and route navigation during the taxi process. This interface will be It is defined in detail in the interface specification.

6. All users can connect to this system through the network (4G network, WiFi, etc.), so as to use the system and interact with the system.

7. The system uses the client-server structure for the architecture design. The client system can be deployed in the mobile terminal of each user, and the server must be deployed in the internal dedicated UNIX server.

8. All system requirements defined in the requirements analysis specification document need to be taken into account in this document and implemented in the architecture design.

**4.2.3 Engineering design constraints**

|  |  |  |
| --- | --- | --- |
| Design elements | | Main constraint |
| Operating environment | operation system | Android / IOS / Linux |
| database | MySQL 14.0 and above |
| WeChat Mini Program | Android/ IOS  WeChat software installed in the system |
| Client software | operation system | Android/IOS |
| Development environment support | operation system | Linux Ubuntu 16.04and above |
| development tools | Eclipse |
| Web framework | Spring MVC |
| RAM | 16GB |

Table Engineering design constraints

**5. Solution**

**5.1 Related architecture patterns**

**5.1.1 C/S pattern**

C/S pattern is one of the software system architecture. It is based on Local Area Network. This pattern allows our application and server run separately so that enhance the stability and flexibility of our system. The most important features of C/S pattern are the security and the quick response. The architecture of C/S pattern is shown below.

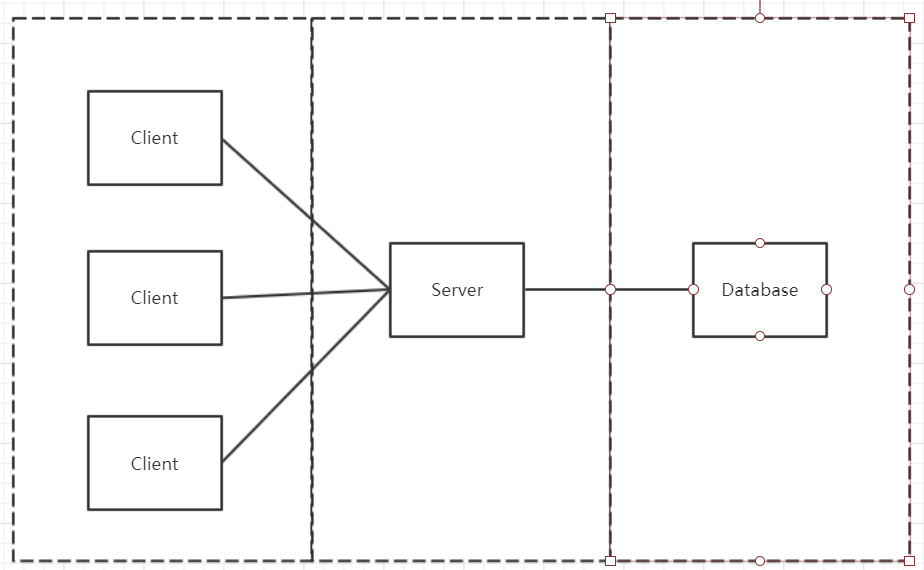


Figure Example of C/S pattern

Here is an example of three-tier C/S pattern. The pattern is composed of three different tiers: client layer, application logic layer and data service layer. The functions of the three tiers is relatively independent.

**5.1.2 B/S pattern**

As the development of the Internet, the C/S pattern is extended to adapt to the variety requirements. So, the B/S pattern emerges as the time requires. B/S pattern is implemented by the web browser and built on the Wide Area Network. B/S allows user in different places and network interfaces to share data, not just in a local area. Furthermore, B/S pattern is easy to maintain and extend. All the features above correspond to our TTMS system. The architecture of B/S is presented below.

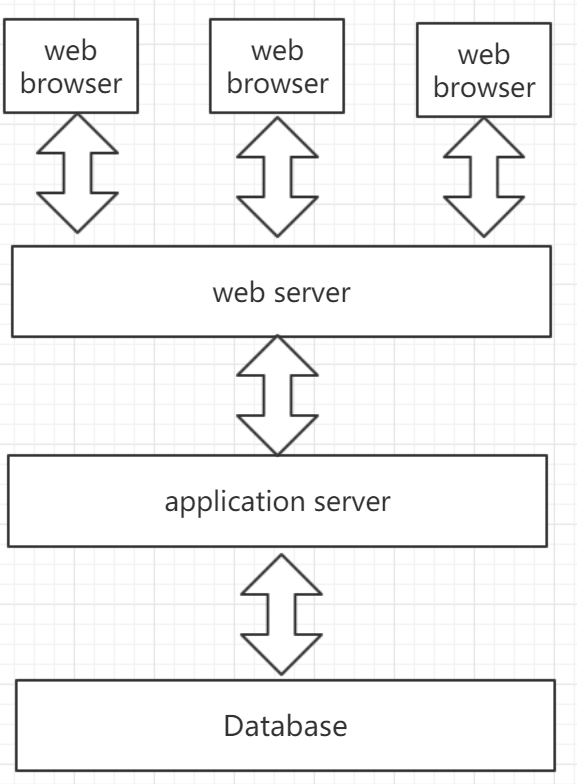


Figure Architecture of B/S pattern

The data collected from web browser will be sent to the web server. Then the data is sent again to the application server. The application server can process the request and perform the business transactions based on the data in the database server.

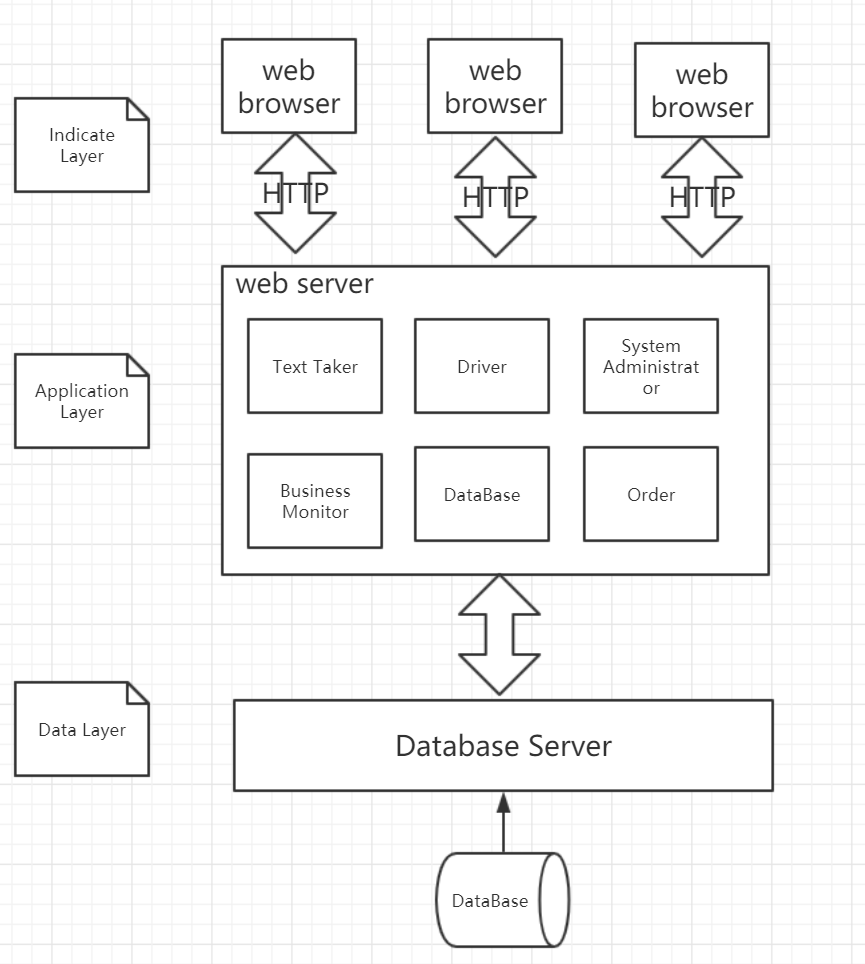
**5.2 Overview of Architecture**

TTMS is an extremely complex distributed application system. The entire system can be separated into Taxi Taker subsystem, Driver subsystem, System Administrator subsystem, Business Monitor subsystem and Database subsystem. There are many relationships and data flow between those subsystems. TTMS implements the functions based on the interaction between each subsystem.

**5.3 Structural View**

TTMS uses the multiple-tier architecture design pattern.

**5.3.1 Conceptual Architecture**



The conceptual Architecture incorporates three tiers: Indicate Layer, Application Layer and Data Layer.

* Indicate layer collects the input from users and send their request to the business processing part. The response from server will also be displayed in this layer.
* Application layer comprises the system server. The request from Indicate layer is expected to be processed in this layer. The response will be generated by server and send back to the waiting user.
* Data layer provides necessary data to help Application layer process request and records the information of transaction in the database.

**5.3.2 Modular Architecture**

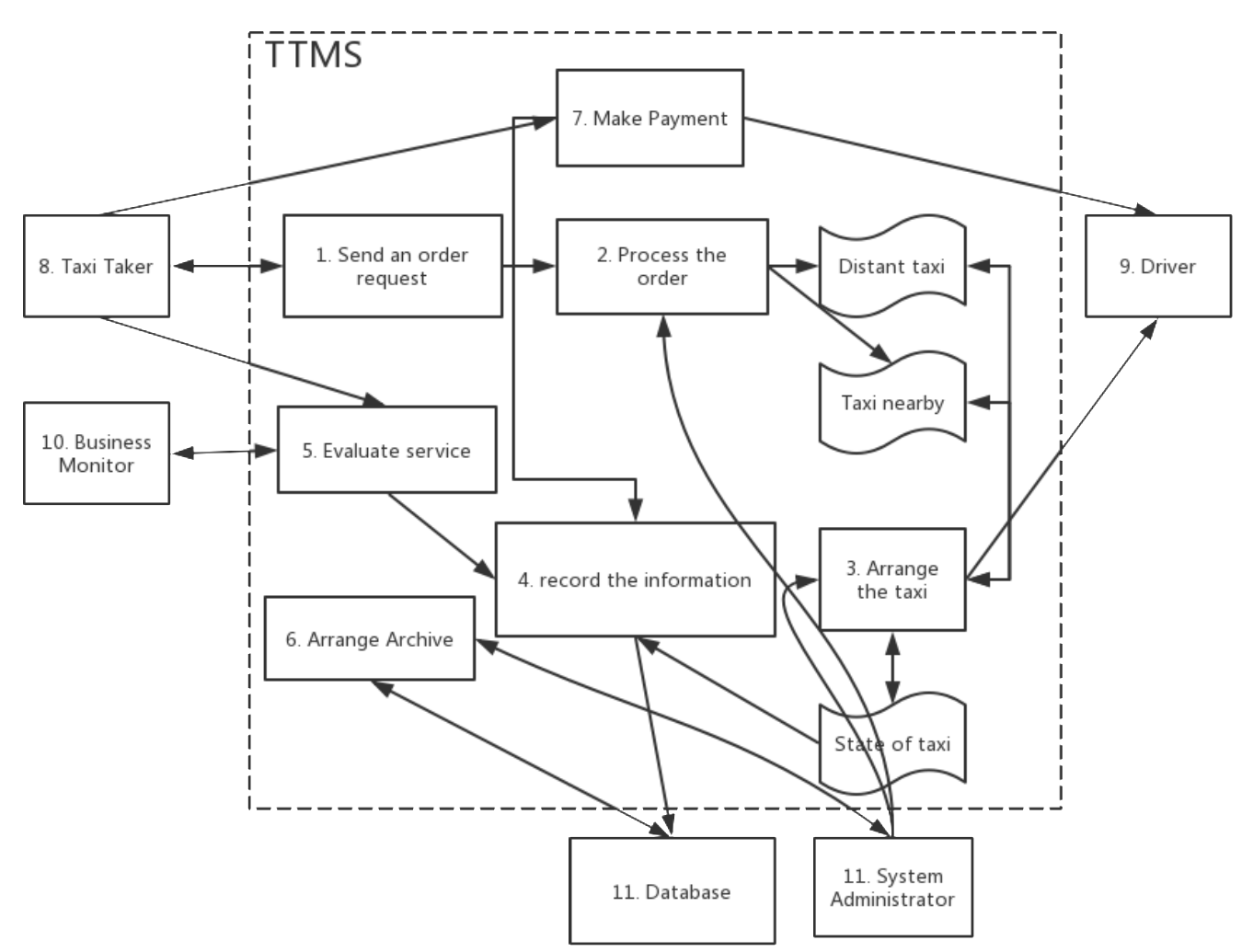


Figure 1Data Flow of different module

Module#1: Send an order request

When the users of our software system want to take taxi, they are required to choose one of the modes from competitive mode and assigned mode by touching the virtual button on their screen. After that, they have to choose their destination and press button to send their order to be accept.

Module#2: Process the order

When the user’s request order is sent to TTMS’s server, system administrator has to deal with the request immediately. Taxi taker’s position and the detail of the order are required in this process. These data are encapsulated into packet and send to the drivers. System administrator has to decide the visible scope of the order based on the condition of the amounts of available taxi nearby. Normally, the order request will be sent to the nearby taxi to make sure the efficiency. If the distant taxi is needed, the taxi fare will be increased correspondingly.

Module#3: Arrange the taxi

When the taxi drivers have just accepted the order which the system administrator sending to them, system administrator have two schedule strategies. Using which one is based on the taking mode of the order.

Module#4: Record the information

When the driver picks up the taker, the service has already begun. The database system will record the information of the order, the car and the driver automatically. When the driver finishes his service, the taxi taker should pay the money and give the evaluation to the driver. These details and evaluation will also be recorded in database as important backup archive.

Module#5: Evaluate Service

When the taxi drivers finish their service, the taxi takers should evaluate their driver. Business monitor can check or analyze the evaluation of drivers. Business monitor are also responsible for receiving the suggestions and complaints from taxi takers.

Module#6: Arrange Archive

System administrator is in charge of checking driver’s information and has right to modify the database. (create item, delete item, update item, extract item)

Module#7: Make Payment

When the taxi drivers finish their service, the taxi taker should pay money to them. The money is calculated based on the travel distance. After the taxi taker making the payment, a receipt is supposed to be generated for the customer.

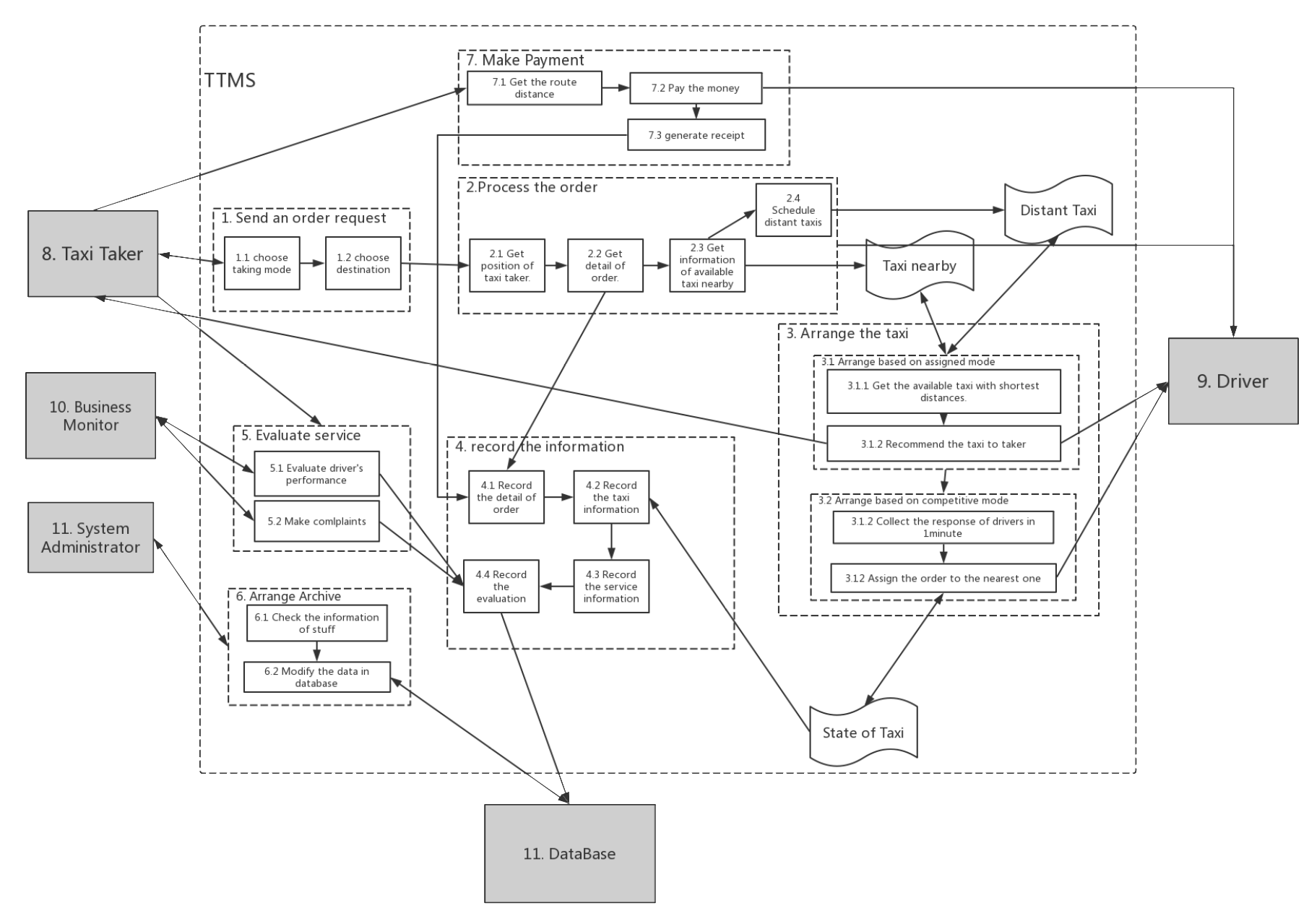


Figure Module with more details

**5.3.3 Runtime Architecture**

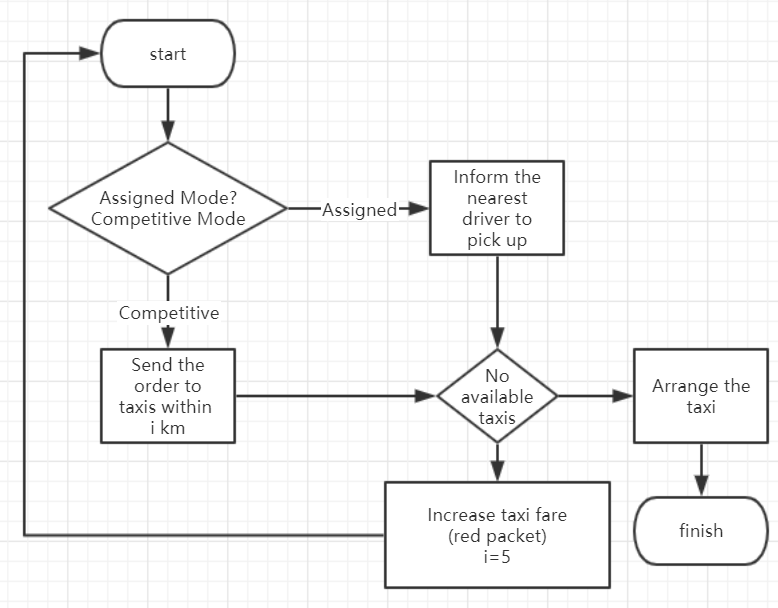


Figure The workflow of certain transaction

In the runtime architecture, we mainly discuss the most essential function of TTMS. As the flow chart shown above, the transaction starts with the option of the user order’s mode. In the assigned mode, the system will figure out the nearest available car to serve taxi taker. As for the competitive mode, the order request will be sent to the idle driver within 3 kilometers, all the drivers compete for the order. If there are no available taxis for a while, the taxi taker can increase the taxi fare with “red packet” to schedule the distant taxis.

**6. Use-Case View**

This section gives a brief description of the use-case view of our software architecture. The use case view describes a set of use cases that represent some significant, central functionality. It also indicates the set of use cases that have a substantial coverage or that stress or illustrate a specific, delicate point of the architecture. The Use cases in TTMS are presented below.

**6.1 Architecturally-Significant Use Cases**

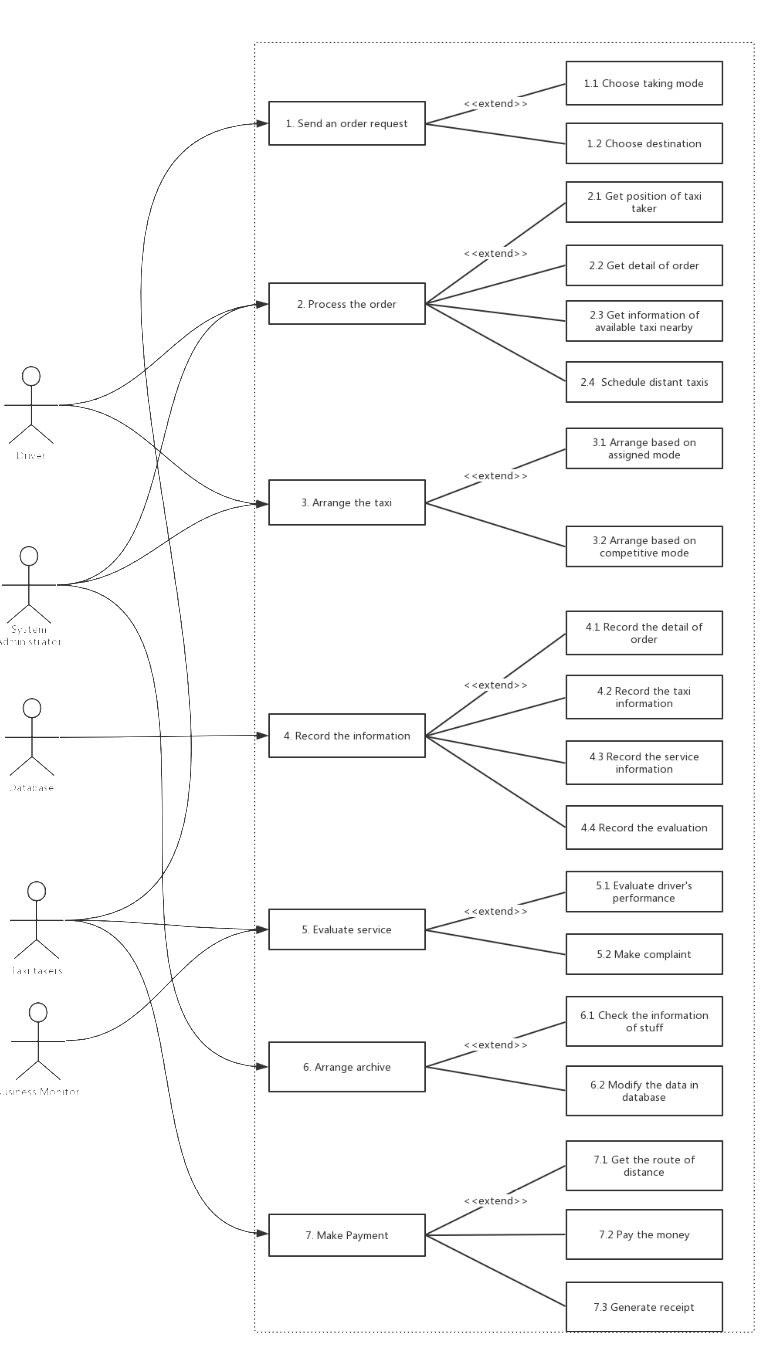


Figure 1 The total use case diagram of TTMS

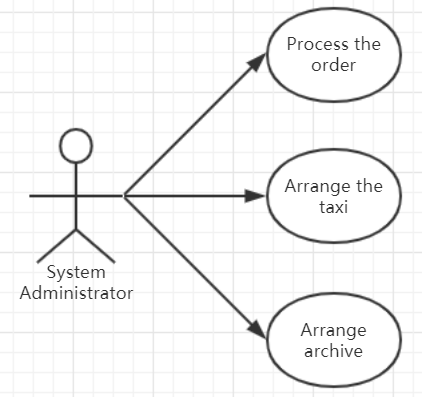
**6.2 User Role**

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.

The user role in TTMS includes system administrator, taxi taker, driver, business monitor.

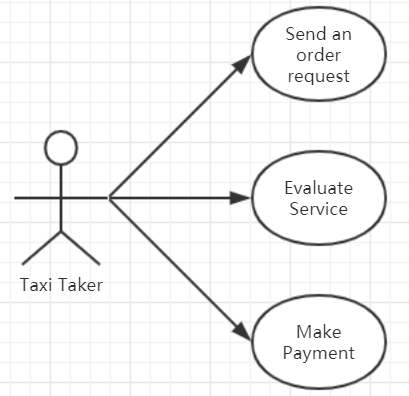
**6.2.1 User Role #1**

System Administrator: Stand in the top layer of TTMS. System administrator can process the request from taxi taker and schedule the taxis in a suitable way. He can also manage the database.



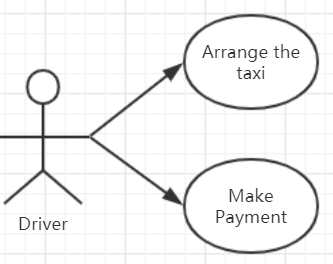
**6.2.2 User Role #2**

Taxi Taker: the applicant of service. His main job is to start the service, evaluate the service and pay for the service.



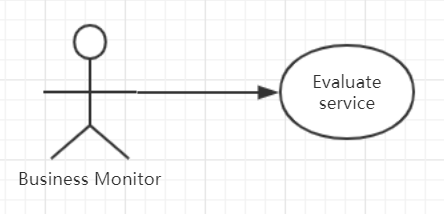
**6.2.3 User Role #3**

Driver: the provider of the service. His main job is to obey the instructions of system administrator.



**6.2.4 User Role #4**

Business monitor: the monitor of the service. His main job is to analyze the evaluation data to make sure the quality of the service.



**6.3 Use Case Description**

**6.3.1 Send an order request**

6.3.1.1 Introduction

When the users of our software system want to take taxi, they are required to choose one of the modes from competitive mode and assigned mode by touching the virtual button on their screen. After that, they have to choose their destination and press button to send their order to be accept.

6.3.1.2 Inputs

Taxi takers click one of the mode buttons.

Taxi takers input or choose their destination name on online map.

Taxi takers click the sending button to send the order to the server.

6.3.1.3 Processing

When the taxi taker finish choosing the mode button, the system is able to initialize the type of the taker’s order. After choosing destination, the system should get the geoinformation of destination and calculate the distance. The data of the order will be encapsulated into a packet and send to the specified taxi drivers based on the mode.

6.3.1.4 Outputs

Competitive mode: An order reminder to available drivers nearby.

Assigned mode: A recommendation of certain driver to taxi taker.

6.3.1.5 Error Handling

Prompt the user to choose destination when the user press sending button without selecting destination.

Remind the user to reselect a destination when the user chooses a destination that cannot be reach.

When the users’ geoinformation are not available, remind them to open the GPS module of their phone.

**6.3.2 Process the order**

6.3.2.1 Introduction

When the user’s request order is sent to TTMS’s server, system administrator has to deal with the request immediately. Taxi taker’s position and the detail of the order are required in this process. These data are encapsulated into packet and send to the drivers. System administrator has to decide the visible scope of the order based on the condition of the amounts of available taxi nearby. Normally, the order request will be sent to the nearby taxi to make sure the efficiency. If the distant taxi is needed, the taxi fare will be increased correspondingly.

6.3.2.2 Inputs

Taxi taker’s request order.

The position of the taxi taker.

Available taxis in certain distance.

6.3.2.3 Processing

The system will obtain the user’s position and check the available taxis within 3km. If there are no or less available taxis in this scope. The system administrator can send the request to schedule the distant taxis with higher taxi fare to serve the current customer.

6.3.2.4 Outputs

The order request sent to the taxi drivers.

6.3.2.5 Error Handling

Switch to another proper driver in a short time when the current chosen driver expressing he is not available.

Remind the driver in time when the taxi takers cancel the order.

**6.3.3 Arrange the taxi**

6.3.3.1 Introduction

When the taxi drivers have just accepted the order which the system administrator sending to them, system administrator have two schedule strategies. Using which one is based on the taking mode of the order.

6.3.3.2 Inputs

The order request and taxi taker’s position.

6.3.3.3 Processing

When it comes to the order with assigned mode, system administrator will calculate the available driver who is nearest to the current taxi taker based on the algorithm. Then, the driver will be recommended to the takers.

When it comes to the order with competitive mode, system administrator will send the order request to the available taxi drivers within the scope. The driver can send their response to compete for the order. After 1 minute, the nearest responder will gain the order.

6.3.3.4 Outputs

The response from driver will be sent to the taxi taker. Taker can see the information of the driver who is serving him.

6.3.3.5 Error Handling

Remind the driver in time when the taxi takers cancel the order.

If the server doesn’t give response for a while, remind the taker try it again.

**6.3.4 Record the information**

6.3.4.1 Introduction

When the taxi drivers finish their service, the related information of this service shall be recorded in the database.

6.3.4.2 Inputs

The order request from taxi taker.

The information of the taxi and driver.

The information of the service. (cost, distance, etc)

The evaluation about the taxi driver.

6.3.4.3 Processing

When the driver picks up the taker, the service has already begun. The database system will record the information of the order, the car and the driver automatically. When the driver finishes his service, the taxi taker should pay the money and give the evaluation to the driver. These details and evaluation will also be recorded in database as important backup archive.

6.3.4.4 Outputs

The information related to the driver, the taker and the service will be stored in the database.

6.3.4.5 Error Handling

Deploy the server of database system in Linux platform to maintain the stability. Always do data backup for the database in case of the loss of data.

**6.3.5 Evaluate the service**

6.3.5.1 Introduction

When the taxi drivers finish their service, the taxi takers should evaluate their driver. Business monitor can check or analyze the evaluation of drivers. Business monitor are also responsible for receiving the suggestions and complaints from taxi takers.

6.3.5.2 Inputs

After the service, the taxi taker is expected to give their evaluation to the driver.

6.3.5.3 Processing

When the taxi drivers finish their service, the taxi takers should evaluate their drivers. The evaluations information will be stored in the database.

Business monitor is in charge of checking and analyzing the data based on some algorithms. Business monitor also collect the suggestions and complaints from users. For the abnormal cases, business monitor is responsible for informing the related department to reform. In the TTMS scenario, they can inform system administrator to modify the database according to their analysis result.

6.3.5.4 Outputs

The output are analysis result of the business monitor and the evaluation information stored in the database.

6.3.5.5 Error Handling

The business monitor judges the driver’s performance based on the big data analysis, not just certain service in order to avoid some customer give negative comment deliberately.

**6.3.6 Arrange Archive**

6.3.6.1 Introduction

System administrator is in charge of checking driver’s information and has right to modify the database. (create item, delete item, update item, extract item)

6.3.6.2 Inputs

New driver’s information.

Message from business monitor.

6.3.6.3 Processing

System administrator logs in the system with special key and acquire the right to manage database.

System administrator carries out the operation based on the requirement. For example, the message from business monitor.

6.3.6.4 Outputs

Creation, modification and deletion of the drivers’ data stored in database.

6.3.6.5 Error Handling

Every operation should be checked thoughtfully in case of incorrect operations.

**6.3.7 Make payment**

6.3.7.1 Introduction

When the taxi drivers finish their service, the taxi taker should pay money to them. The money is calculated based on the travel distance. After the taxi taker making the payment, a receipt is supposed to be generated for the customer.

6.3.7.2 Inputs

Travel distance.

Taxi fare.

6.3.7.3 Processing

System are able to figure out the total money the taker should pay based on the distance they have already traveled during this service.

The taxi taker clicks the “pay” button to pay the appointed money.

The system will generate a receipt of this service for the taker automatically.

6.3.7.4 Outputs

Receipt for the taxi taker.

6.3.7.5 Error Handling

If the taxi user cannot use credit card to pay for this trip, then cash is OK. The receipt should generate manually.

**7. Logical View**

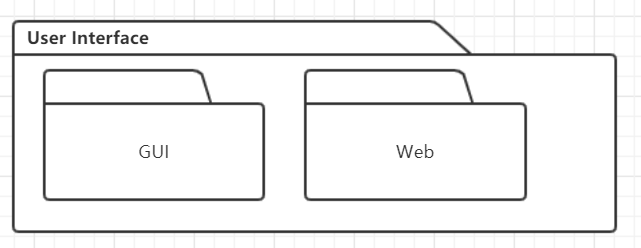
This section gives a brief description for the architecture of TTMS. Describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers.

Class diagrams and domain models are included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

**7.1 Architecture Overview – Package and Subsystem Layering**

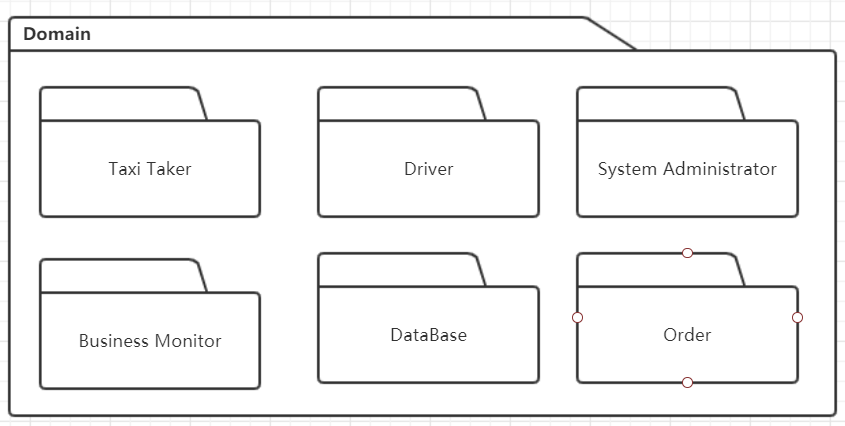
The logical view of TTMS is comprised of the 3 main packages: User Interface Layer, Domain Layer, and Technical Services Layer.

**7.1.1 User Interface Layer Package**

**

The User Interface Layer Package contains classes like GUI from java or class related with webpages for each of the forms that the actors use to communicate with the TTMS System. Boundary classes exist to support login, sending takers’ request order, helping drivers accept the order, display the geoinformation and make payment.

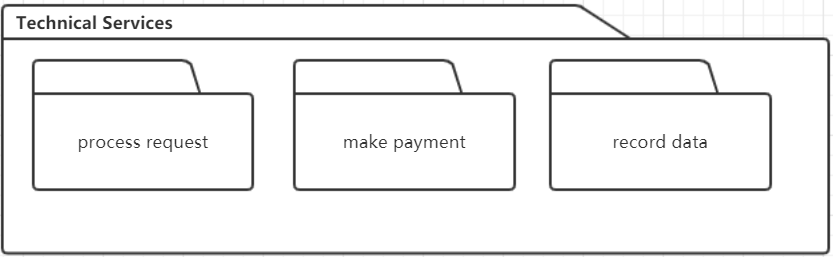
**7.1.2 Domain Layer Package**

**

The Domain Layer Package includes entity classes for TTMS to process the requirements from taxi takers. The classes are abstracted from the use cases, including Taxi Taker, Driver, System Administrator, Business Monitor, Database and Order.

Taxi takers request the services; drivers provide the services; business monitor check and evaluate the services; order represents a certain service; database record the data of services; system administrator schedules and manages the services.

**7.1.3 Technical Services Layer Package**

**

The Technical Services Layer Package contains control classes for interfacing with the TTMS system. For instance, the classes related to process the transaction the services, like processing taxi takers’ order, managing things related with payment and recording the service information. This part often can be reused in other similar software system.

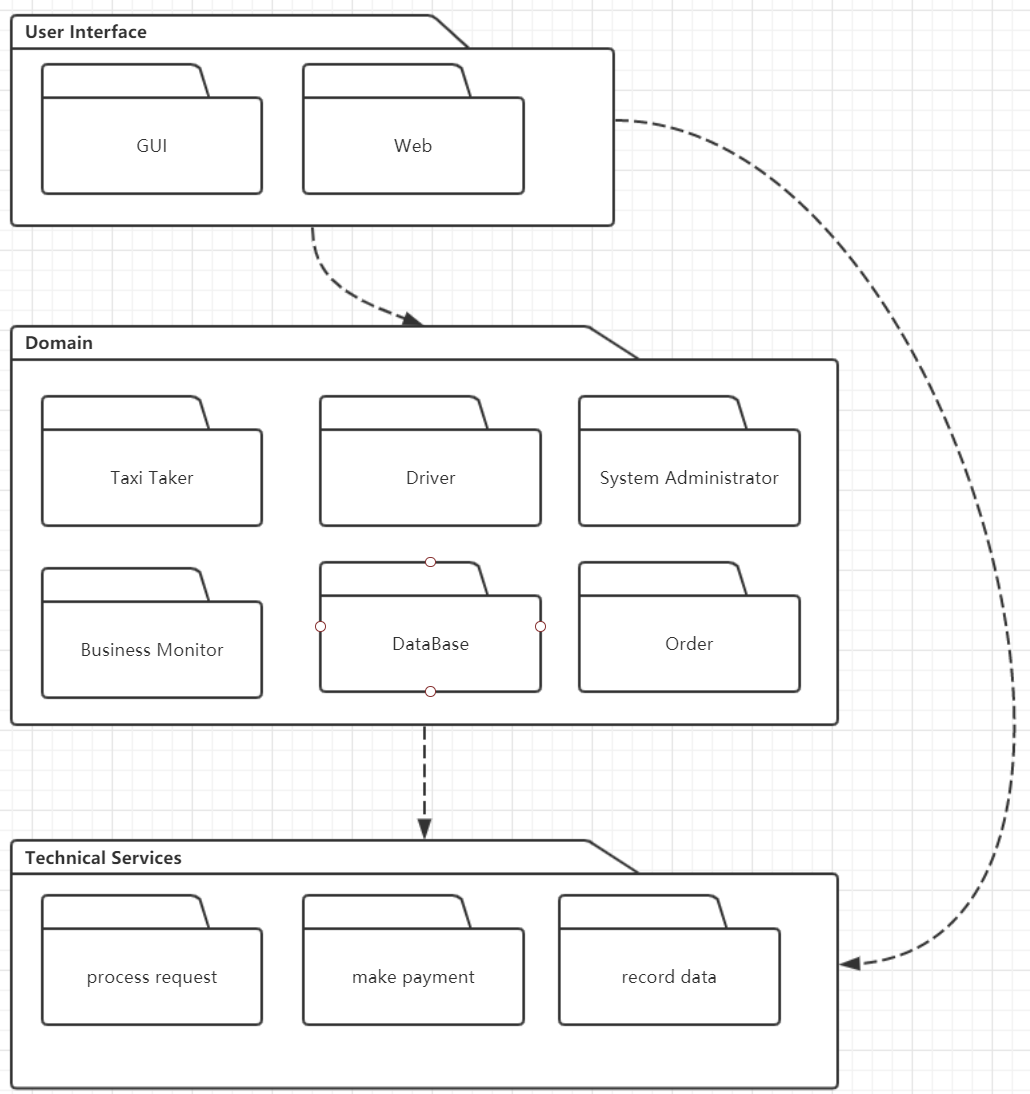
**

Figure 2 Overview of Logical Architecture

Both of the User Interface Layer and Domain Layer can have access to Technical Service Layer. And User can also interact with the entities in system based on the user interface layer.

**8. Process View**

This section describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. Also describes the allocation of objects and classes to tasks.

The Process Model illustrates the TTMS classes organized as executable processes. Processes exist to support Taxi Taker subsystem, Driver subsystem, System Administrator subsystem, Business Monitor subsystem and Database subsystem.

**8.1 Interaction and Configuration**

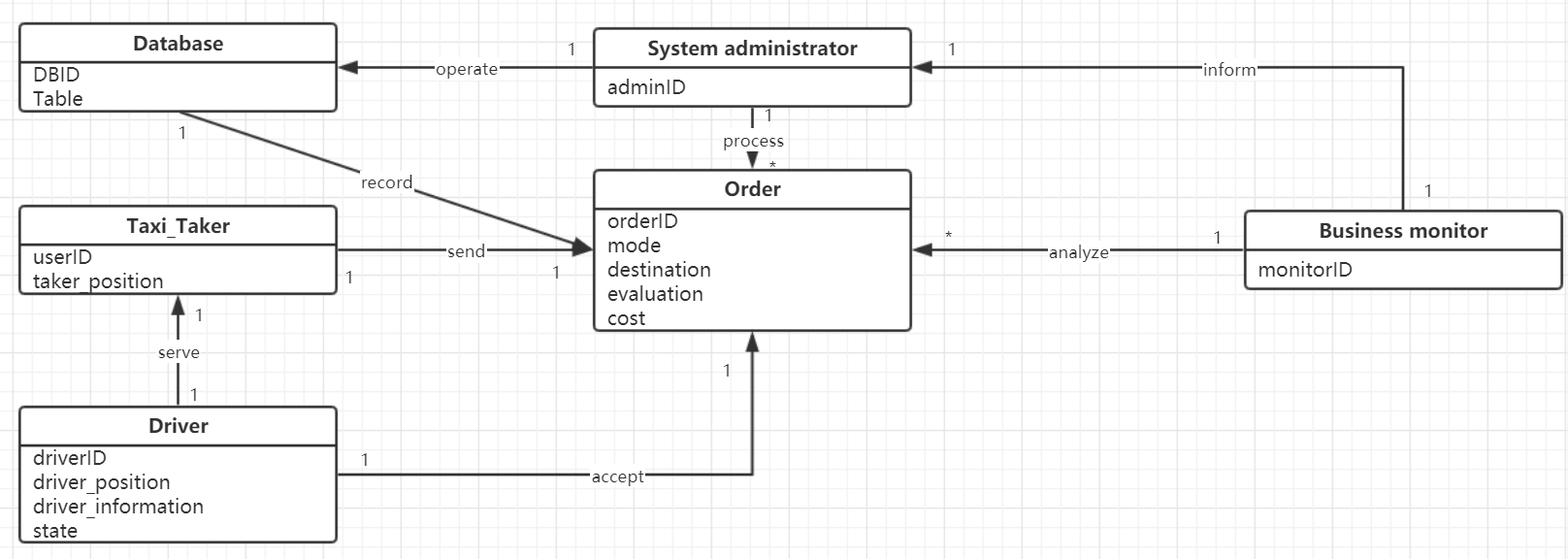
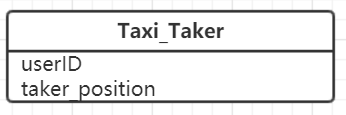


Figure 2 Overview of Class and relationship in TTMS

**8.2 Classes**

**8.2.2 Class#1: Taxi Taker**

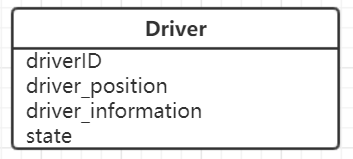


There are two attributes in class Taxi Taker.

UserID is the unique identification of taxi taker.

Taker position is a group of coordinates indicate the location of taker.

**8.2.3 Class#2: Driver**



There are four attributes in class Driver.

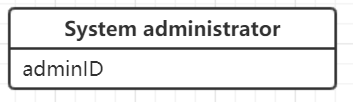
DriverID is the unique identification of each driver.

Driver position will be used to calculate the distance between taker and driver.

Driver information manifest the personal information of driver.

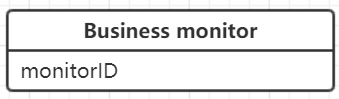
State reveals the taxi driver is idle or busy.

**8.2.4 Class#3: System Administrator**



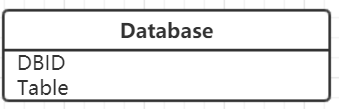
There is only one attribute, adminID, in class System\_administrator to play a role of unique identification of the administrator.

**8.2.5 Class#4: Business monitor**



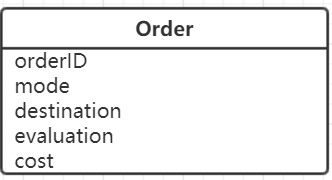
There is only one attribute, monitorID, in class Business\_monitor to identify the business monitor.

**8.2.6 Class#5: Database**



There are two attributes in class Database. The DBID is the unique identification of database. The table attribute represents the data stored in the database which are in a table form.

**8.2.7 Class#6: Order**



There are five attributes in class Order. OrderID is the unique identification of each order. The value of mode represents the type of the order, which is supposed to be set by the taxi taker before they send the request. The destination attribute contains the destination coordinates. Evaluation attribute record the customer’s comment for this service. Cost attribute is the money taker should pay for this service.

**9. Deployment View**

This section describes the various physical nodes for the most typical platform configurations. Also describes the allocation of tasks (from the Process View) to the physical nodes. The physical nodes comprise the Mobile phone, Application Server, Desktop PC and Database.

This section is organized by physical network configuration; each such configuration is illustrated by a deployment diagram

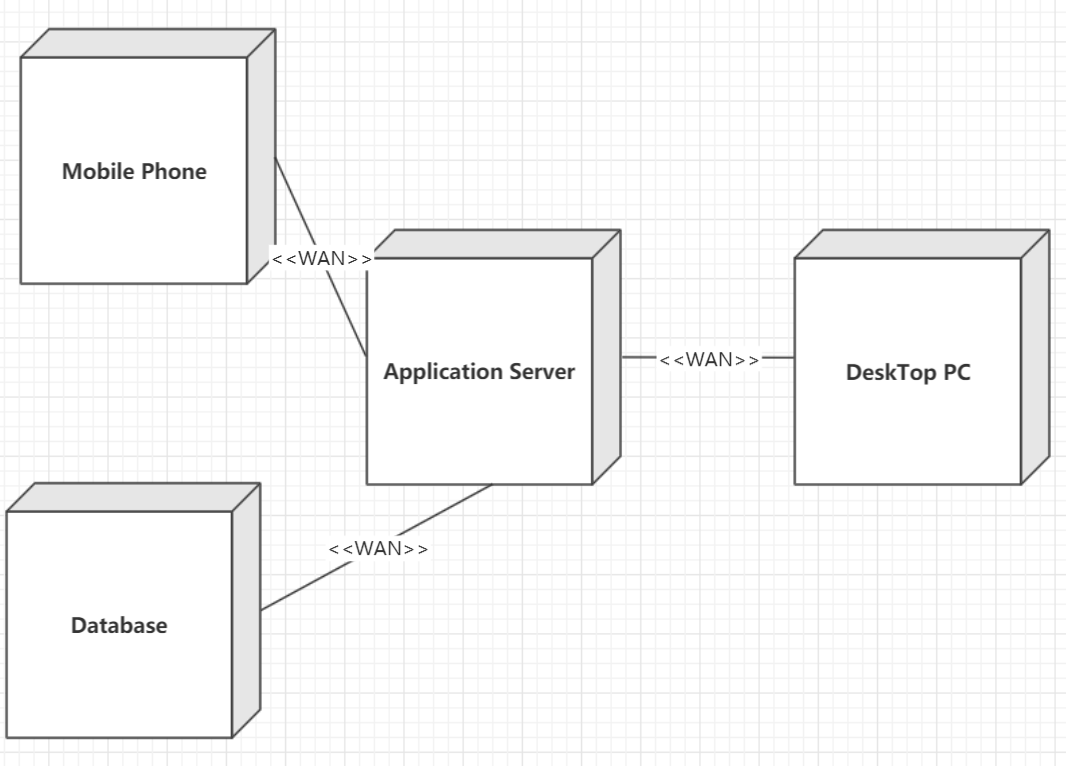


Figure Deployment View of TTMS

1. Mobile Phone

Taxi Takers send their order requests using their mobile device like cell phone which are connected to the Wide Area Network.

Driver receive their assignments or compete for the orders using their Applications on mobile phones which are connected to the Wide Area Network.

2. Application Server

The Application Server is the main UNIX Server. All users and administrator have access to the Server through the Wide Area Network.

3. Desktop PC

System administrator operate the database or schedule the taxis using the desktop PCs which are connected to the Wide Area Network.

Business monitor analyze the business condition and evaluation of taxis through the desktop PCs

4. Database

The details of each transaction will be recorded in the database. The database shall be deployed in the cloud server.

**10. Development View**

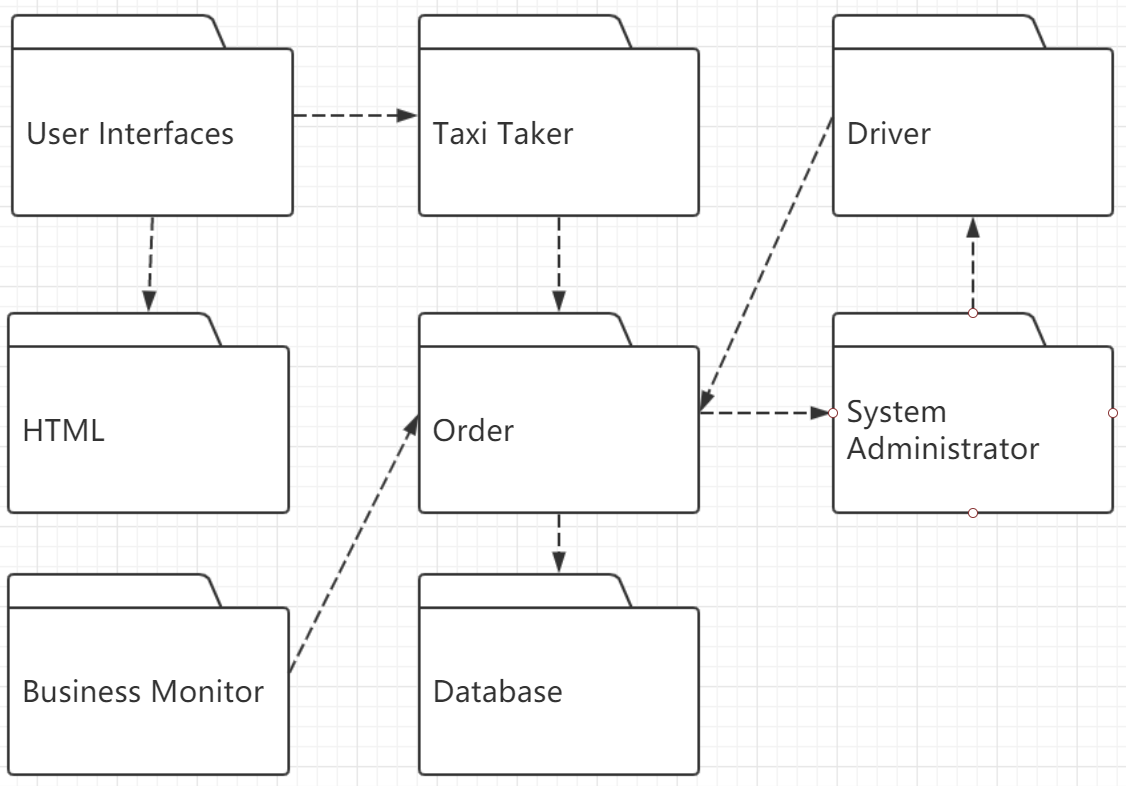


Figure Development View of TTMS

Development View describes the static architecture of software in the development environment, which is mainly written for the software programmers. The development view is composed of subsystem and module.

**11. Quality**

**11.1 Quality Requirements**

* The mobile user-interface shall be more than IOS 11.0 or Android 7.0.
* The user interface of the Taxi Taking Management System shall be designed for ease-of-use and shall be appropriate for a software-literate user community with no additional training on the System.
* Each feature of the TTMS shall have built-in online help for the user. Online Help shall include step by step instructions on using the System. Online Help shall include definitions for terms and acronyms.
* The TTMS shall be available 24 hours a day, 7 days a week. There shall be no more than 4% down time.
* Mean Time Between Failures shall exceed 300 hours.
* Upgrades to the client portion of TTMS shall be downloadable from the UNIX Server over the internet. This feature enables all users (taxi takers and drivers) to have easy access to system upgrades.

**11.2 Scenario Analysis**

**11.2.1 Use-case Scenarios**

* Scenario#1: Taxi takers hope they can send their request easily and their orders be adopted by taxi drivers in a relatively short time.
* Scenario#2: Drivers hope they can be assigned to specified customer or get orders when they are idle.
* Scenario#3: Business monitor assesses the condition of takers’ evaluations to drivers and inform system administrator to modify the database if necessary.
* Scenario#4: Database records the information about the orders automatically when the order has been processed.
* Scenario#5: System administrator schedules the available taxis in the scope when taxi takers’ order come.

**11.2.2 Growth Scenarios**

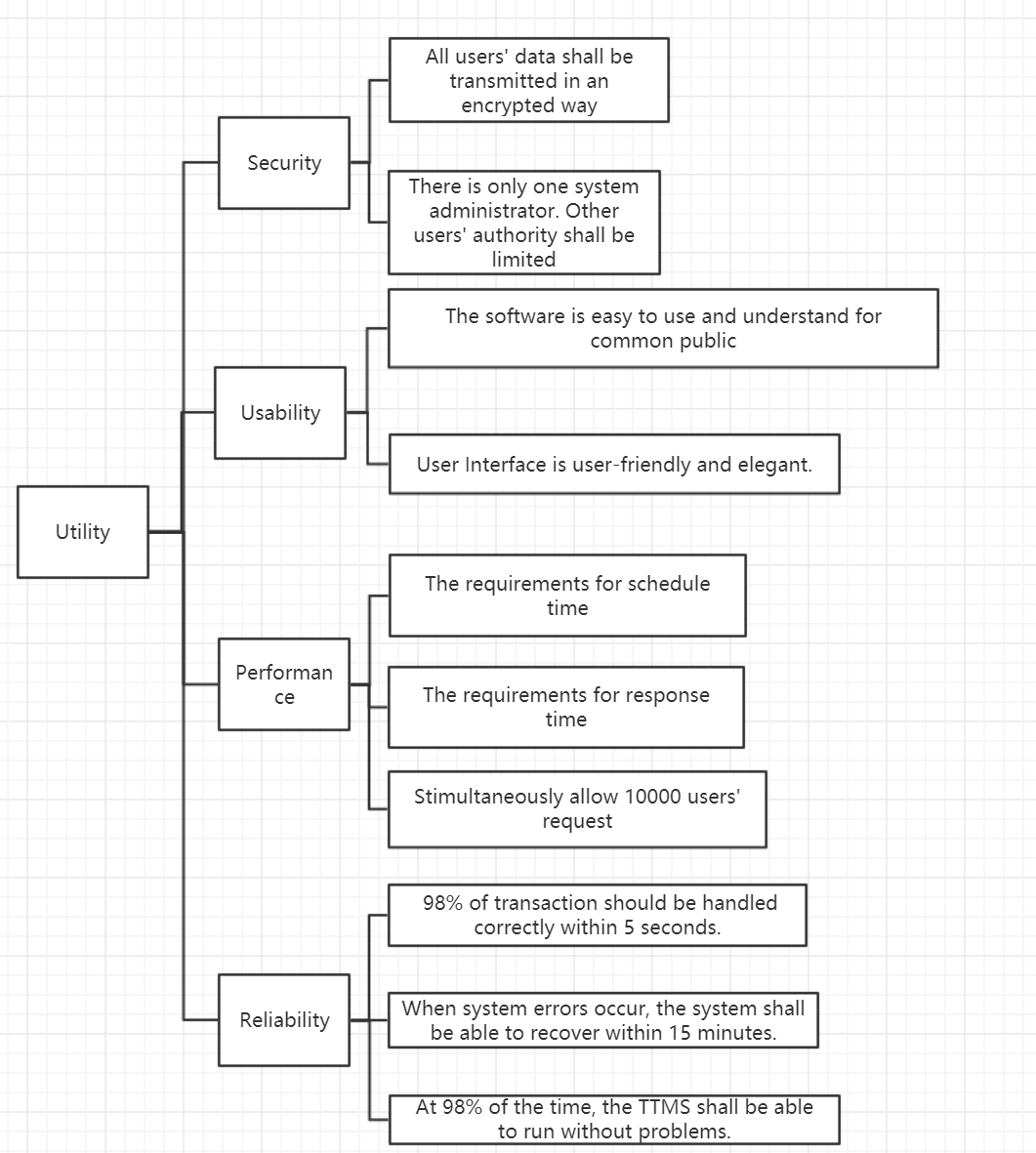
* Growth Scenario#1: In the future, it may be necessary to change the user interface, which only needs to change the logic of the indicate layer, without changing the code of the business logic layer and the data layer. It is hoped that the system can complete the modification only by adding 2 weeks of work.
* Growth Scenario#2: In the future, it may be necessary to add the amounts of data servers or improve the performance of the data servers, which can decrease the time when user accesses the webpages.
* Growth Scenario#3: In the future, it may be necessary to extend the scale of data tables and construct indexes for specified columns.

**11.2.3 Exploratory Scenarios**

* Exploratory Scenario#1: Add the 5G technology to the system technical stack. With the help of 5G, we can obtain more accurate position information and have better real-time performance on the Internet.
* Exploratory Scenario#2: The system can be connected with traffic management system to automatically match vehicle information and owner information. Furthermore, the traffic management system can help TTMS know the traffic condition more accurate and real-time.
* Exploratory Scenario#3: When there is something wrong with database system, the system can be recovered from the log files.

**11.3 Prototype Analysis**

Prototype Analysis is based on the Utility Tree method. The utility tree of TTMS is shown below.



**11.4 Risk**

* Project progress requirements: Assess if the architecture of the project supports the project progress requirements. In the iterative model, the first version of the product must be completed within 6 months, the second version within 9 months, and the third version within 12 months.
* Change of the requirements: The client may modify or add some requirements in the process of developing software system. The architecture is supposed to adapt to the new changes.
* Maintenance of the system: In order to ensure the lifespan of the software, the software must adapt to the frequent changes of business requirements, and modify the software according to the change of business requirements. The cost and cycle of modification are directly related to the software architecture.
* Insufficient Cost: The wrong estimation of the architecture may result the shortage of money.

**12. Extensibility**

Extensibility means the software system shall enable more taxi drivers, taxi takers and taxi companies take part in the software activities.

* When we design the architecture of TTMS, we shall make the components have high cohesion and low coupling.
* As for the hardware system, we should use the database, web container, application servers with high performance.
* The system is supposed to provide several public interfaces to the outside environment so that the system can be easy to integrate with new components in the pursuit of development.
* Optimize the algorithm and process in the system to give quick response to users and allow high concurrency.

**13. Modifiability**

In fact, we can extend TTMS design model to variety area. The taxi taker subsystem, driver system, administrator subsystem, monitor subsystem and database subsystem can be reused in or sold to other similar software like sharing bicycle, sharing car and e-commerce platform.

The taxi taker could represent the user who request the services.

The driver could represent the entity who is scheduled to provide services.

The administrator can be mapped into anything system as the controller of the system.

The database represents the storage subsystem in any other similar system.

The monitor subsystem is not necessary but we also can transplant it to similar system if we have similar requirements.

For example, TTMS can be transformed into a model for e-commerce platform like Jidong. We can call it ECMS. Let us explain how it works.

The entity who send the order request is referred as *customer.*

The entity who provide service is the *warehouse*.

And we do not have to modify the functions of remain entities.

The system administrator is able to process the order sent by the customer and arrange the nearest warehouse where the item is available to assign postman to deliver the item

The flowing data flow diagram represents the transformed model in a modular architecture.

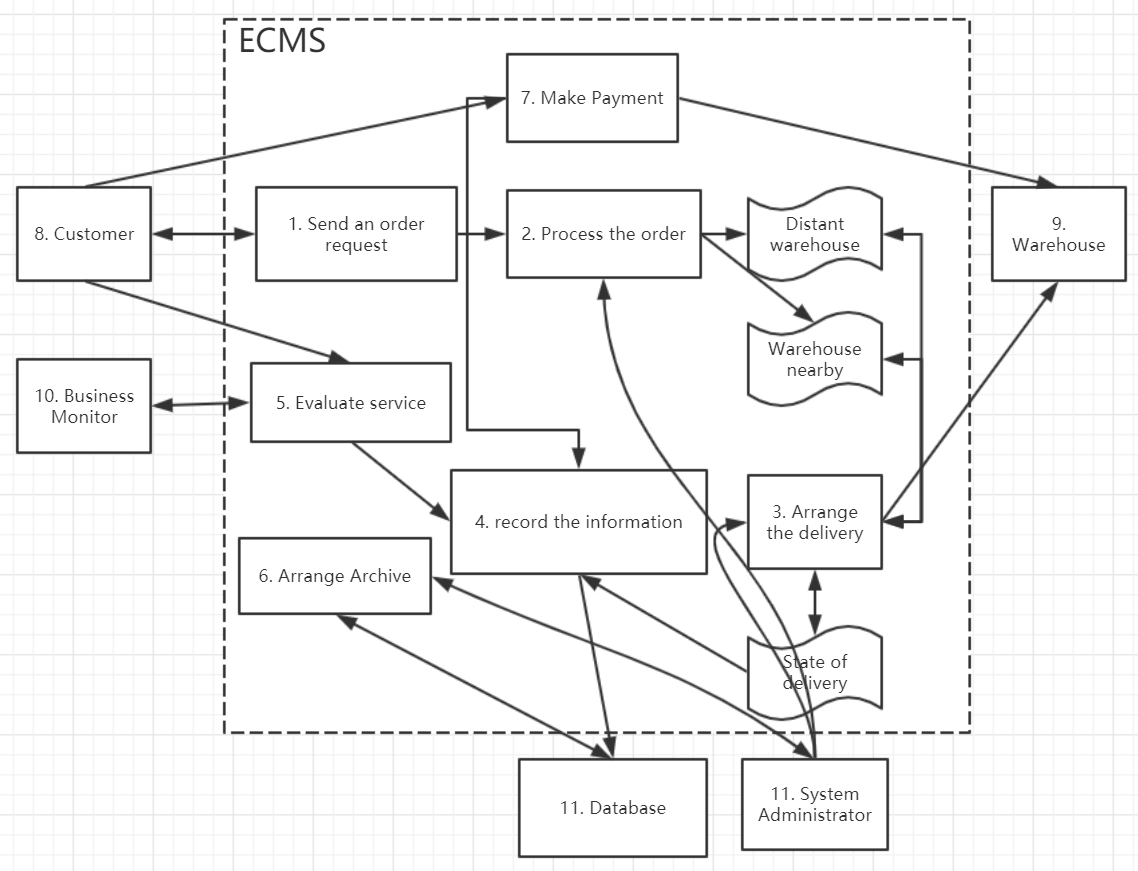


Figure 4 Extend the TTMS model to E-commerce area

These subsystems can become the valuable assets of company because they can be reused in the following projects and sold to other company to earn money.