**WiFi advertising system in an underground railway station**

**Design document**

Directory

[1. Documentation introduction 3](#_Toc57236803)

[1.1. Writing purpose 3](#_Toc57236804)

[1.2. Document scope 3](#_Toc57236805)

[1.3. Reader object 3](#_Toc57236806)

[2. Project introduction 3](#_Toc57236807)

[2.1. Project description 3](#_Toc57236808)

[2.2. Project background 3](#_Toc57236809)

[2.3. Project objectives 4](#_Toc57236810)

[2.4. Project user 4](#_Toc57236811)

[3. Description of requirements 4](#_Toc57236812)

[3.1. Overall demand 4](#_Toc57236813)

[3.2. Functional requirements 4](#_Toc57236814)

[3.2.1. Requirement numbering rules 4](#_Toc57236815)

[3.2.2. Overall module division 5](#_Toc57236816)

[4. Functional requirements 5](#_Toc57236817)

[4.1. Business function requirements 5](#_Toc57236818)

[4.1.1. Query data query001 5](#_Toc57236819)

[4.1.2. Basis function query001 5](#_Toc57236820)

[4.2. Interface requirements 7](#_Toc57236821)

[4.2.1. Station location change interface data001001 7](#_Toc57236822)

[4.2.2. Subway station surrounding facilities to change the interface data001002 8](#_Toc57236823)

[4.2.3. Change of interface for service around subway station data001003 8](#_Toc57236824)

[5. Database environment description 9](#_Toc57236825)

[5.1. Database system 9](#_Toc57236826)

[5.2. Design tools 9](#_Toc57236827)

[5.3. Database configuration 9](#_Toc57236828)

[6. The database naming convention 9](#_Toc57236829)

[6.1. Table name specification 9](#_Toc57236830)

[6.2. Data item name specification 9](#_Toc57236831)

[6.3. Data table structure definition 10](#_Toc57236832)

[7. Database design 10](#_Toc57236833)

[7.1. Logic design 10](#_Toc57236834)

[7.1.1. wifiad\_station 10](#_Toc57236843)

[7.1.2. wifiad\_location 11](#_Toc57236844)

[7.1.3. wifiad\_service 11](#_Toc57236845)

[7.1.4. wifiad\_facility 12](#_Toc57236846)

[7.2. Physical design 12](#_Toc57236847)

[8. Design 13](#_Toc57236848)

[8.1. System Architecture 13](#_Toc57236849)

[8.2. Web Server Application Object Model 13](#_Toc57236850)

[8.3. Concurrency and Scalability 14](#_Toc57236851)

[9. Appendices 14](#_Toc57236852)

[9.1. A Network Topology 14](#_Toc57236853)

# Modeling

## Network Topology diagram

Fig1. Network Topology diagram

Figure 1 diagrams the system’s current network, and the modifications proposed. It consists of a simple eithernet-based intranet. WiFi connectivity for the mobile devices of the system’s patrons is provided via a WiFi Access Point. Our proposed system would add a server to the intranet to host the web server application and database components.

## Packet filtering in AP

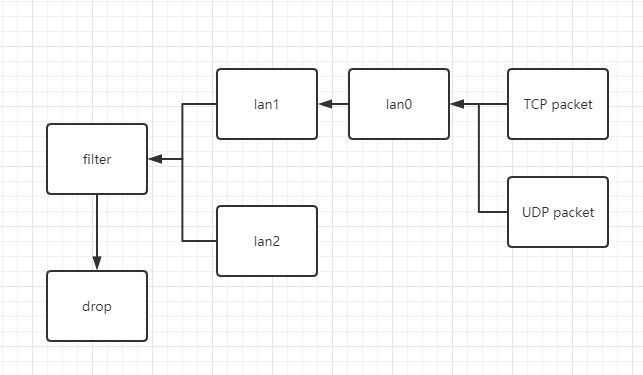


Fig2. 1.2. Packet filtering in AP

lan: 0 Local network interface for wireless network access

lan: 1 Local network interface connected to the main server

lan: 2 Local network interface connected to the DNS server

All the TCP packet from lan:0 will be transported to lan:1

UDP packet from lan:0 will be divided into two parts, DNS query packet and others. DNS query packet will be transported to lan:2 which connected to local DNS server and other packets will be dropped.

## Sequence Diagram

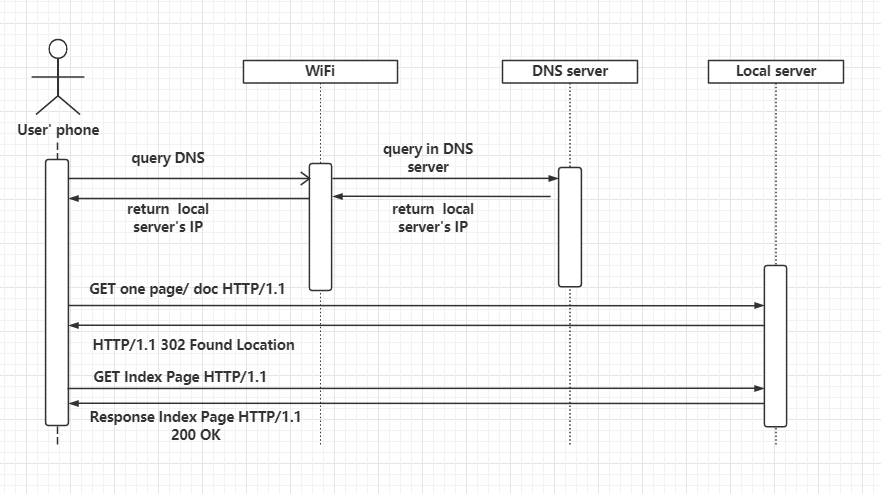


Fig3. Sequence Diagram

When smart phones try to get network service by domain name.

AP get a DNS query then transport it to local DNS Server.

DNS Server return the IP of main Server to Smart Phone.

Smart phone sends a GET request to Main Server.

Main Server could not find the resource then return the redirect response with code 302 and location of Index page.

Smart phone gets the redirection response then try to GET Index Page from Main Server.

Main Server return Index page with success code 200.

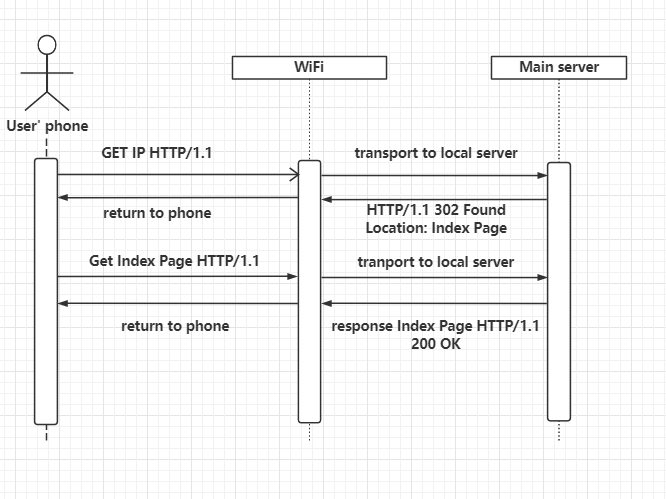


Fig4. Sequence Diagram

When smart phones try to get network service by IP.

AP get a http ‘GET’ request then transport it to Main Server, because the HTTP protocol is implemented based on the TCP protocol.

Main Server could not find the resource then return the redirect response with code 302 and location of Index page.

Smart phone gets the redirection response then try to GET Index Page from Main Server.

Main Server return Index page with success code 200.

## Concurrency and Scalability

Concurrency is achieved by isolating the handling of request for each user: there is no shared data between user sessions of the application except for where data is written to/queried from permanent storage. This allows mosts kinds of concurrency conflicts to be resolved in the database transaction layer. However, certain classes of simultaneously write conflicts can occur if multiple staffpersons attempt that of another. Though unlikely, if it is necessary to avoid this scenario, we can employ 2-phase commits to ensure that a station’s information has not been changed in the database since the staffperson began editing the station’s information.

The biggest scalability roadblock in web server applications is typically thread IO blocking: Threads and context-switching create a great deal of overhead and slow down response times. Thread handling HTTP requests may be user up by blocking IO operations. Using continuations supported by Scala, it is possible to easily implement asynchronous calls and callbacks that preclude blocking threads on IO, thus freeing up those threads to handle other HTTP requests while waiting on IO to complete, thus improving scalability drastically. In the unlikely case that additional scalability is needed, the web application can be deployed on additional web server running side-by-side with a shared database.

# Design

In this section, we describe our proposed design of the system. This proposed to design is made to maximize the flexibility, extensibility, and maintainability of the system’s design, while ensuring it meets all of the customer’s requirements. Furthermore, after deployment, it will server as part of the system’s technical documentation for future maintainers of the system.

## System Architecture

We propose to build a web application using a standard, 3-tier architecture, as illustrated in Figure 5.The application will consist of a web server application and a backend database for premanent for permanent storage. Users will connect and interact with the application via HTTP using standard web browsers running on client machines throughout the library, connected via a library intranet.

The main component of the system is the web server application, and will be responsible for validating and processing requests from the web browsers, enforcing bussiness rules, and rendering results as HTML to the client browser. The database component will be responsible only for presistence and enforcing basic integrity constraints.

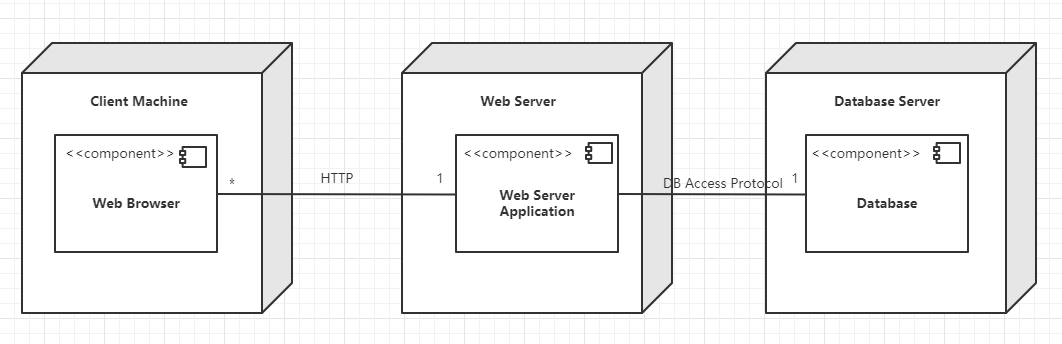


Figure 5: Deployment Model

## Web Server Application Object Model

In this section we will explain the design of our object model of our object model that will be user to implement the proposed web server application. This object model is divided into three layers: HTTP Routing/Hangling Layer, Presentation Layer and Business Layer. Each layer’s responsibilities are defines as follows:

HTTP Routing/Hangling Layer :Responsible for low-level HTTP Resquest/Response handling. Listens for new requests, applies routing rules to find corresponding server actions, and writes the result as an HTTP Respones.

Presentation Layer :Responsible for basic user-input validation, interpreting requests into actions and presenting the results as HTML to be rendered by the user’s browser.

Business Layer :Responsible for querying/updating permanent storage while enforcing business rules and ensuring its integrity.

## Access Point

DHCP setting:

IP range: 192.168.1.100 - 192.168.1.250 (up to 150 devices are allowed at same time)

IP address: 192.168.1.1 (deployed on Gateway)

DNS server address:

192.168.2.1

Main server address:

192.168.3.1.

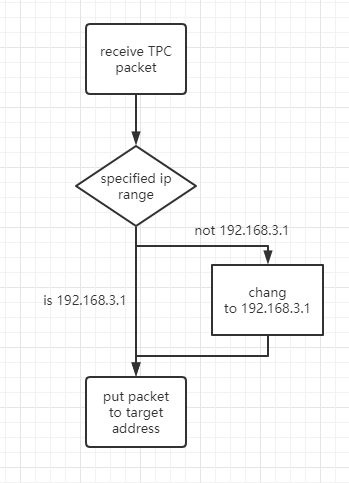


Fig6. TCP filter

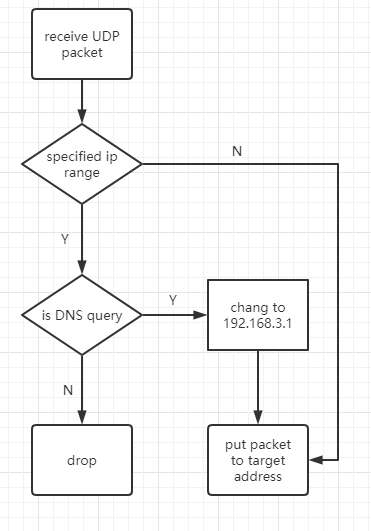


Fig7. TCP filter

## DNS server

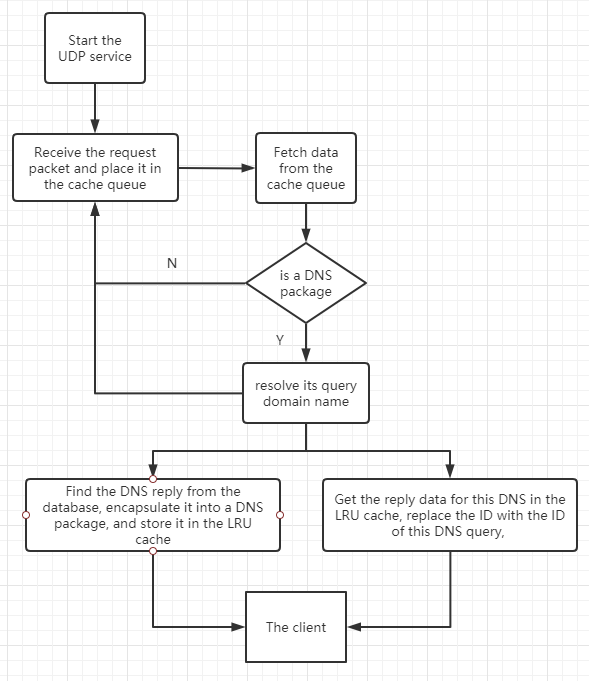


Fig8. DNS model

## Spring MVC model

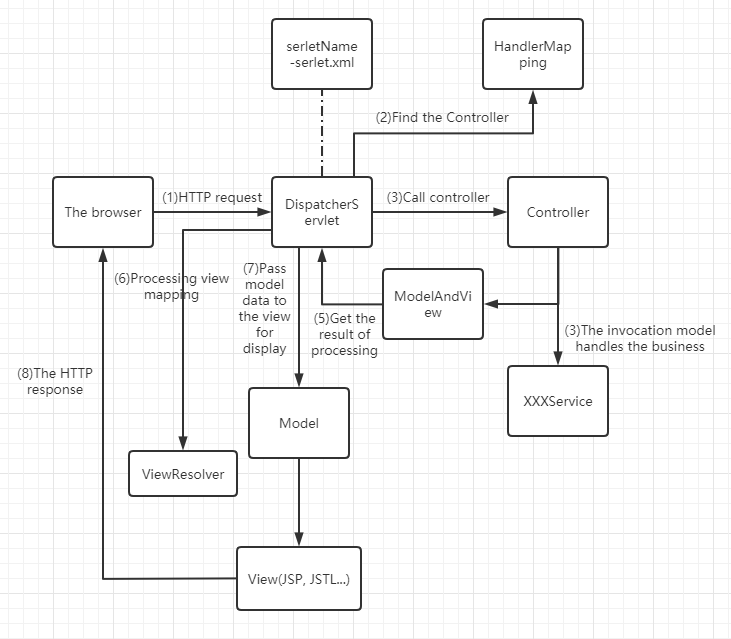


Fig9. Spring MVC working schematic

The Spring MVC workflow is as follows:

1.The client request is submitted to the DispatcherServlet.

2.It is up to the DispatcherServlet Controller to look for one or more HandlerMapping and find the Controller that handles the request.

3.The DispatcherServlet submits the request to the Controller.

4.The Controller calls the business logic processing and returns ModelAndView.

5.The DispatcherServlet looks for one or more ViewResolver view parsers and finds the view specified by ModelAndView.

6.The view is responsible for displaying the results to the client.

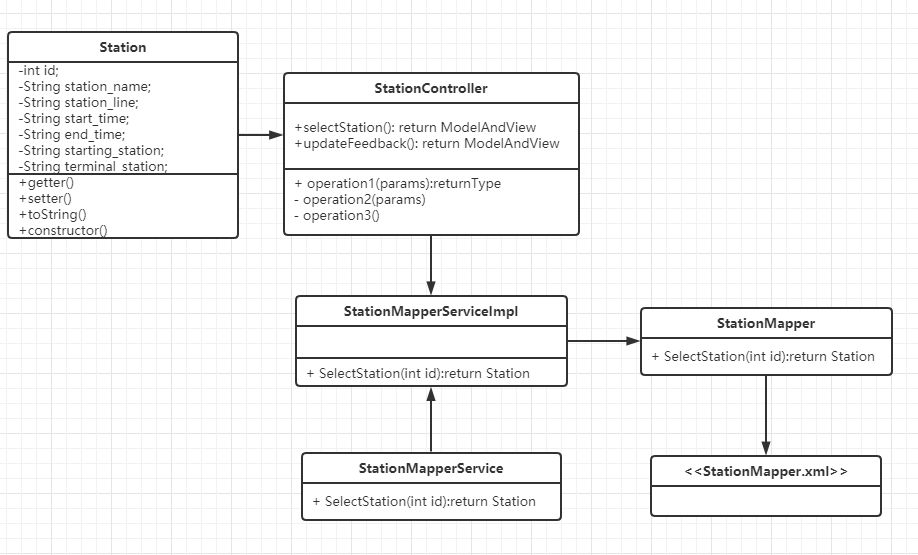


Fig10. UML Class Diagram

## Project user

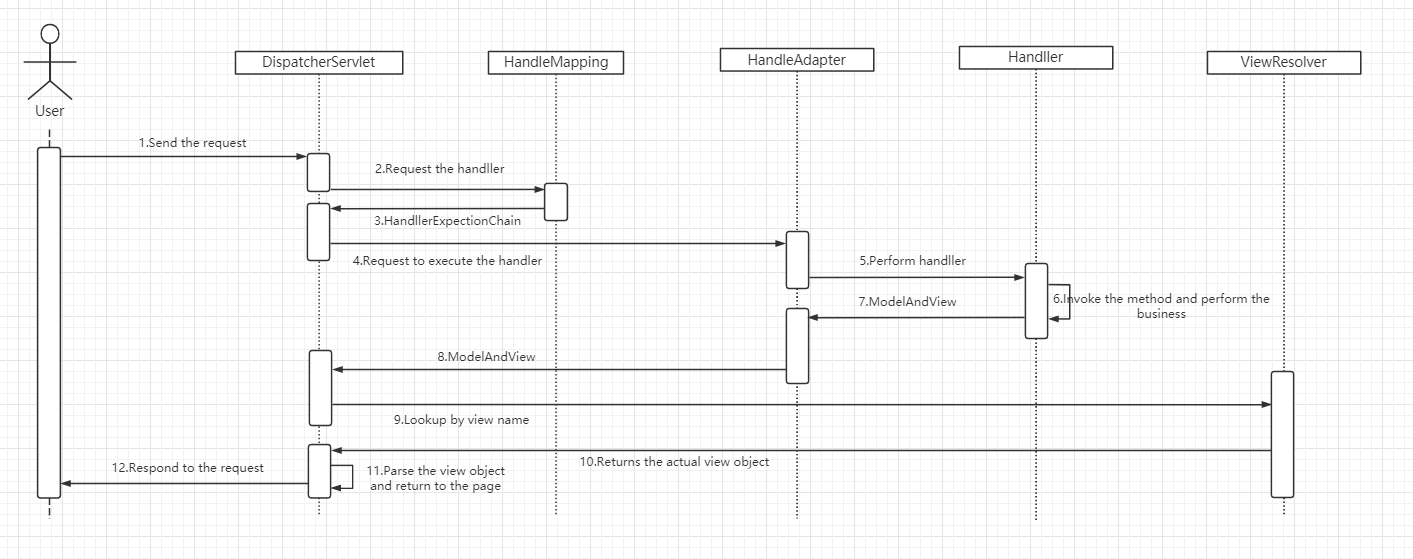


Fig11. Spring MVC HTTP request squence diagram

# Database environment description

## Database system

MYSQL 8.0

## Design tools

MySQL Workbench 8.0 CE

## Database configuration

|  |  |
| --- | --- |
| **Table space** | wifiad |
| **Table space initialization size** | 500M |
| **Self-Increment** | 10M |
| **User name** | root |
| **Password** | WiFiAD123 |

# The database naming convention

## Table name specification

Names must begin with wifiad\_

Name must be clear

Names must be easy to understand

Names should not be too long

The naming should be as informative as possible

Name to extract keywords. For example: user information table: LNG \_USER

## Data item name specification

Name must be clear

Names must be easy to understand

Names should not be too long

The naming should be as informative as possible

## Data table structure definition

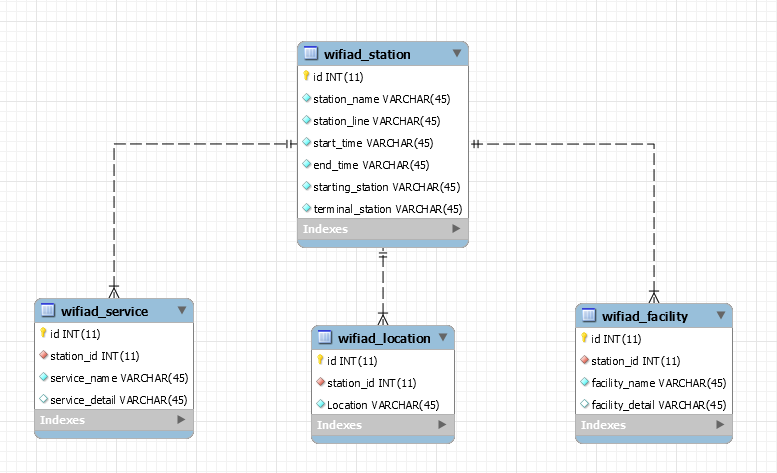


Figure 12: ER Diagram

# Database design

## Logic design



### wifiad\_station

#### Table structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Describe | Datatype | Primary key or not | Note |
| id | ID | INT(11) | PK AI |  |
| station\_name | station name | VARCHAR(45) |  |  |
| station\_line | station line | VARCHAR(45) |  |  |
| start\_time | start time | VARCHAR(45) |  |  |
| end\_time | end time | VARCHAR(45) |  |  |
| starting\_station | starting station | VARCHAR(45) |  |  |
| terminal\_station | terminal station | VARCHAR(45) |  |  |

#### Table relationships

id is the primary key of wifiad\_station

### wifiad\_location

#### Table structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Describe | Datatype | Primary key or not | Note |
| id | ID | INT(11) | PK AI |  |
| station\_id | station id | INT(11) | FK |  |
| location | station location | VARCHAR(45) |  |  |

#### Table relationships

id is the primary key of wifiad\_location

station\_id is foreign key of wifiad\_station

### wifiad\_service

#### Table structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Describe | Datatype | Primary key or not | Note |
| id | ID | INT(11) | PK AI |  |
| station\_id | station id | INT(11) | FK |  |
| service\_name | service name | VARCHAR(45) |  |  |
| service\_detail | service detail | VARCHAR(45) |  |  |

#### Table relationships

id is the primary key of wifiad\_service

station\_id is foreign key of wifiad\_station

### wifiad\_facility

#### Table structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Describe | Datatype | Primary key or not | Note |
| id | ID | INT(11) | PK AI |  |
| station\_id | station id | INT(11) | FK |  |
| facility\_name | facility name | VARCHAR(45) |  |  |
| facility\_detail | facility detail | VARCHAR(45) |  |  |

#### Table relationships

id is the primary key of wifiad\_facility

station\_id is foreign key of wifiad\_station

## Physical design

According to the above logical design relationship, the following table is obtained.

CREATE TABLE `wifiad\_facility` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`station\_id` int(11) NOT NULL,

`facility\_name` varchar(45) NOT NULL,

`facility\_detail` varchar(45) DEFAULT NULL,

PRIMARY KEY (`id`)

)

CREATE TABLE `wifiad\_location` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`station\_id` int(11) NOT NULL,

`Location` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

)

CREATE TABLE `wifiad\_service` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`station\_id` int(11) NOT NULL,

`service\_name` varchar(45) NOT NULL,

`service\_detail` varchar(45) DEFAULT NULL,

PRIMARY KEY (`id`)

)

CREATE TABLE `wifiad\_station` (

`id` int(11) NOT NULL,

`station\_name` varchar(45) NOT NULL,

`station\_line` varchar(45) NOT NULL,

`start\_time` varchar(45) NOT NULL,

`end\_time` varchar(45) NOT NULL,

`starting\_station` varchar(45) NOT NULL,

`terminal\_station` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

)