The technical paper

1 INTRODUCTION

1.1 Overview of project

The project aims at developing a postgraduate roommate matching system which can allocate two students to one dormitory according to their habits and interest.

It is divided into three parts for the development of the project, web designing, database storing and matching algorithms.

1.2 Team organization

Group	Content	Tool	Members
Group1	web designing	Python	Jiaxu Li, Zhongran Liang
Group2	database storing	MySQL	Xin Huang, Lixin Zhu, Yinyi Liu
Group3	matching algorithms	Python	Duomi Huang, Ruitao Wang

1.3 Project management strategy

Frequency of meeting: once a week

internal procedures:

1. The web designing and algorithms are processing simultaneously.
2. After finishing web designing, the MySQL database connects to web by using python in flask frame.
3. Create a matching database and two tables to store the information collected from users.
4. After storing, make a program to export two excel files from database and transmit to algorithm to run.
5. After running, the result comes out and is showed in the web.

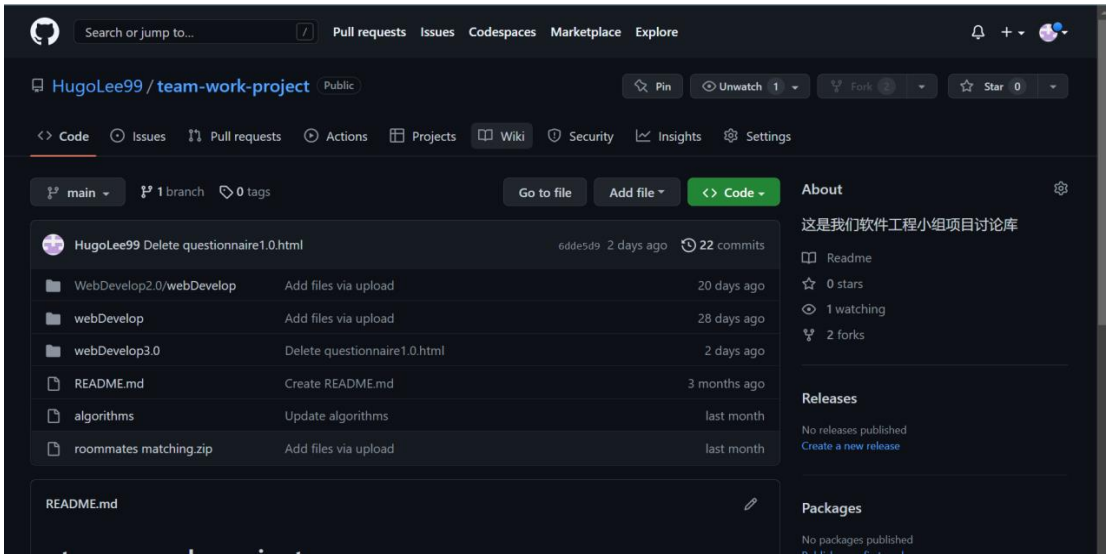
2 THE PROJECT PLAN AND RISK ASSESSMENT

Page Design Part:

Initially, we plan to design our own page totally, however when we found that is quite time

wasting, and the rendering is not good enough. So, we tried to design based on some templates which are copyrighted. (We do the most of the design and all of second level pages, the templates just offer the Home Page structure). In order to meet the demand from algorithm group, we learn the knowledge of JavaScript, and set a requirement on the sum of weights entered by users.

The biggest Obstacles we have is version control, versions are always updated out of sync. To solve this problem, we use GitHub repository to help us manage the versions.



The other obstacle is that at beginning, we have no idea of how to use JavaScript which is a big problem in developing pages. So, we search the document of JavaScript while we using it.

Algorithm Part:

We met many difficulties during the central algorithmic development.

At the early stage, our algorithm group tried our best to find a suitable algorithm to calculate the matching degree between two persons. However, we failed to complete this for a variety of reasons. For example, the interface mismatched our systems or the version of code is out of date. Even though we find an available algorithm, the complexity of code is immeasurable and we must spend much effort studying relevant knowledge. But our time is limited. Under this dilemma, we gradually thought out a method of searching our ideal code.

Firstly, we checked the code's test cases, which could help us screen out a lot of code. Then we checked the module in this code. If the module is out of our knowledge, we gave it up. If the

first two requirements are satisfied, we read the content of code and extracted its core idea. Finally, we developed our own algorithm based on the ideas of acquisition. After taking this strategy, we eventually found out our ideal code and got some ideas of matching. Through a period of discussion, we determined the primary design of the algorithm. We would like to call it “item scoring method” and “weight assignment method”. We will explain the two methods in the following content. Next, we came to the second stage--writing test. The most significant trouble we met was the detailed function’s setup. It was difficult for us to plan the functions and determine detailed output in advance. However, we still made no effort to achieve it by surfing a great number of examples. With our persistent effort, we successfully used unittest module in python to write some tests for our future algorithm. The next stage is completing code. What annoyed us most in this stage is how to make sure the two who have matched are the best combination. Our algorithm group discussed for a long time. Finally, we decided to list all possible combination and sum up each score. The scores are arranged from the highest to the lowest. And we picked it up one by one. In this way, we can ensure the combination is the best under current situation. It is also worth mentioning that our scoring mechanism for each item undergo three iterations. We think the current matching result is relatively reasonable.

Databases Storing:

The main obstacle is that MySQL database is a totally new content for us which we haven't approach to it.

Then we organize our veins. We first need to know how web develops and search on the Internet that how to connect the web and database. In web developing, it uses flask frame, then we code in the flask frame to store the value that enter by users in the web. All programs are run in python, so we use pymysql package to connect the root MySQL database and enter the MySQL statements to operate the data to store into the database.

After storing into the databases, we find the way to export into excel files on the internet.

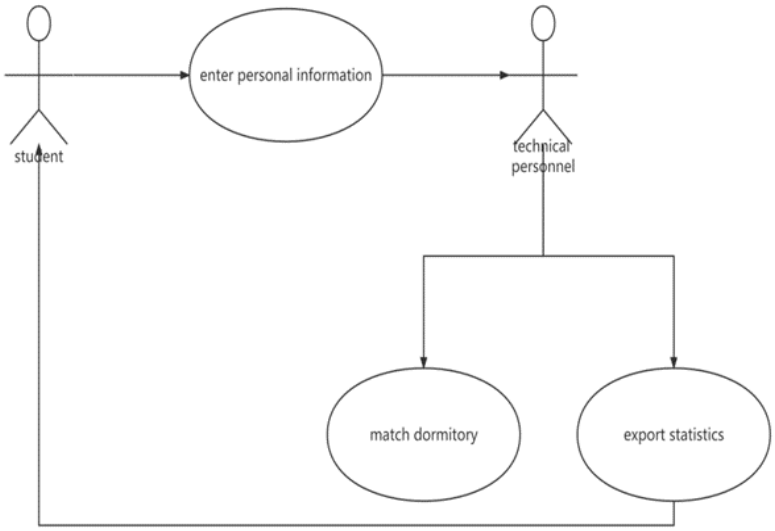
All in all, after we organize our veins and correctly find the methods on the Internet, all of them are successfully solved and just need to revise according to other parts' requirements.

3 REQUIRMENTS

3.1 Requirements

3.1.1 Functional requirements

Functional requirements for the Postgraduate Roommate Matching system, used to allocate postgraduate dormitories:



1. A user shall enter the relative information about their habit and interest.
2. The system shall collect up to 20 students' information into database, then transform to two excel tables to program.
3. The system shall automatically come out with all matching result.

3.1.2 Non-Functional requirements

Product requirements

1. The system can satisfy the basic function of roommates matching according to individual's requirement: interest and habits
2. Through number of users entering information, after executing, the program can provide the result on the website.

Organizational requirements

1. The system programming language is python, so the executing environment is python. And it should install MySQL database.
2. After entering the website and entering all users' information, the program should be executed again to come out with result.

External requirements

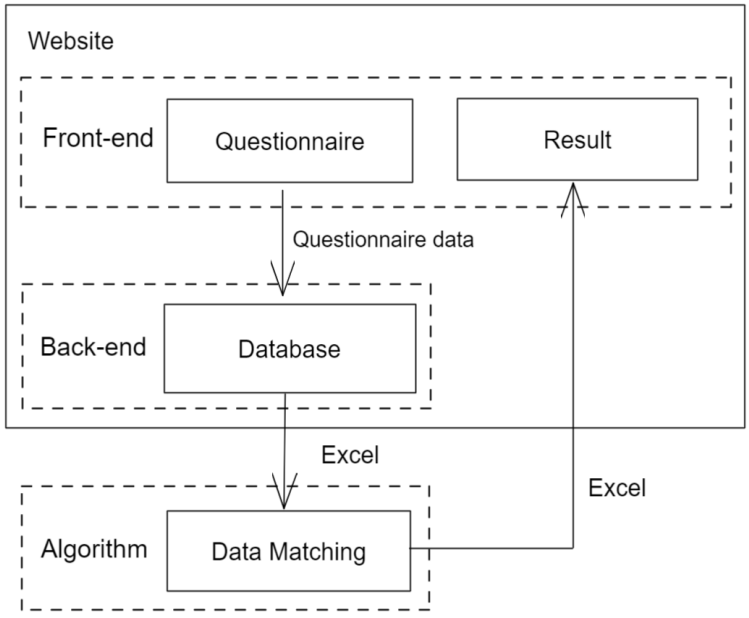
1. Once the program executes, there must be up to two users enter because the matching should have more than two users to match.

2. The standard of the matching algorithms just according to general idea. It is possible to appear disagreement about the result. So, after matching, it might need some change for individuals.

4 DESIGN

4.1 Overview of design

We designed a web page to help users optimally assign cohabitants using an algorithm. By doing this, we through the web page to collect user habits and personality, and then through the server and algorithm processing results, show on the web page.



4.2 key component designs

1. By using html and CSS we design our website and design the questionnaire, which can be used to interact with user.
2. The information collected from front end would be stored in MySQL database. The web is in the flask frame, so the value is stored in specific form. Through pymysql, the program connects to the MySQL database and store the information entered by users by MySQL statements.
3. After collecting data, the program would export two excel files from database and transmit to algorithm.

4. Collecting data from the front side and transforming it into our program. To achieve this, we defined a Person class in the program to store each person's information. Each Person class's instance represents one person. Then we defined a function to create everyone's class instance from the excel table and stored them in a python list.
5. calculating the matching degree between two people. We firstly set up a score mechanism for each information provided by questionnaire. In short, we gave a score according to this term's matching degree between two person's information. In order to achieve it, we defined function for each item in the questionnaire to calculate the single item score. The next stage is to sum up all the item score so that we can figure out the matching degree of these two people. But this raised a problem. All questions in the questionnaire belongs to multiple choice question. The convergence rate of options is very high, which will lead to phenomenon that many pairs show the same degree of matching. Once two pairs with the same score come the same person. We will get in the trouble selecting which pair. To avoid this phenomenon, we introduce "weight assignment method". Let user enter a weight to each question representing how important they think this item. We multiply each item's score by its corresponding weight that one person gave. In this way, we get the matching score between two persons from the perspective of one side. And if we replace the weight with another one, we can gain the score from the perspective of another side. Summing up the two perspective's score. The higher total score, the better matched the pair. We can avoid matching the same degree as much as possible. In our code, we evaluated all possible combination's score and store it in the 2D-list.
6. Matching dormitory according to their matching degree. To solve this problem, we defined a new dormitory class to store a dormitory's information including two person's instance and their total score. Then we defined a function to complete match. The 2D-list that was gained before are traversed in the function. We gain all combination of dormitory and store them in the list called "dormitoryScore" although all person are matched with others. The next stage is obtaining the final dormitory result. We sorted "dormitoryScore" list from the highest score to the lowest one. Next, we picked up the dormitory from the top of the list one by one. If one person has been matched in the other dormitory, we skip the current dormitory combination. At last, we can confirm the final list of dormitories.

4.3 Data Organization

The data collected from the users are divided into two parts:

1. The choices that users altered. These are collected into one table with name and gender.
2. The degree that users attach importance to. These values are number in the range of 0-15 and collected into another table with name.

4.4 Databases Structure

The databases used in our program is MySQL database, which is in open source. The information collected in database named "matching" and there are two tables in it.

4.5 User Interface Design

In terms of interface, we use the web as a bridge between us and our users. Python Flask package is used as the framework of web development. We used the post and get interfaces to submit the user form. The data is then interacted with the backend.

5 THE PROOF-OF-CONCEPT CODING AND INTEGRATION

The achievement of algorithm is completing two person's matching and outputting the result. The process of development is not required any budget. The data which are used to test can be created by our own. And the programming language is python which we have learned before. In addition, we have decided the main idea of code. Therefore, we think the development of algorithm is available from the perspective of POC.

There are two key parts of algorithm.

1. Calculating matching score between two persons.
2. Matching dormitory.

To gain the matching score, we defined a function called "sum_up_score", which takes one parameters "personList" which stores all person's instance. In this function, we traversed the personList and paired people up. Then calculating all combination's matching scores from two person's perspective. One combination's score is consisting of the total of product of one item's score and its weight. Finally, we assigned all scores to the 2D-list.

```
def sum_up_score(personList):
    '''sum up the score between two people.'''
    allscore=[]
    for i in range(len(personList)):
        onescore=[]
        for j in range(len(personList)):
            score=(score_habit(personList[i],personList[j])*float(personList[i].imp_habit)
                   +score_social(personList[i],personList[j])*float(personList[i].imp_social)
                  +score_cleanliness(personList[i],personList[j])*float(personList[i].imp_cleanliness)
                   +score_bath(personList[i],personList[j])*float(personList[i].imp_bath)
                  +score_wash_clothes(personList[i],personList[j])*float(personList[i].imp_wash)
                   +score_clean(personList[i],personList[j])*float(personList[i].imp_clean_frequency)
                   +score_heavy_food(personList[i],personList[j])*float(personList[i].imp_heavy_taste)
                   +score_outsound(personList[i],personList[j])*float(personList[i].imp_outsound)
                  +score_play_game(personList[i],personList[j])*float(personList[i].imp_play_games)
                  +score_tem_airconditioner(personList[i],personList[j])*float(personList[i].imp_tem)
                   +score_hobby(personList[i],personList[j])*float(personList[i].imp_hobby))
            onescore.append(score)
        allscore.append(onescore)
    return allscore
```

After obtaining the matching score, we need to finish the allocation of dormitory. That is how to pick up the most suitable combination.

```
def matchDormitory(personScoreList):
    '''match the two person's dormitory'''
    i=0  #person one
    j=1  #person two
    dormitoryScore=[]  #all dormitory combination's score and persons
    Score=[]  #all dormitory combination's score
    while i < len(personScoreList):
        while j < len(personScoreList[i]):
            sumScore=personScoreList[i][j]+personScoreList[j][i]
            orderScore=(str(len(Score)),sumScore)       #combination's number and score
            Score.append(orderScore)
            dor=dormitory(sumScore,i,j)     #one possibility of dormitory
            dormitoryScore.append(dor)
            j+=1
        i+=1
        j=i+1
```

We firstly list all the combinations of dormitory in “dormitoryScore” list and store its corresponding score in the “Score” list by loop.

Then we sort the list from the highest score to the lowest score so that we can pick up dormitory better.

```
Score.sort(key=lambda x:x[1],reverse=True)      #sort the socre from big to small
```

Finally, we select the combination of dormitory from the “dormitoryScore” list from head to tail. If one person has been matched before, we jump to the next combination until all persons are matched.

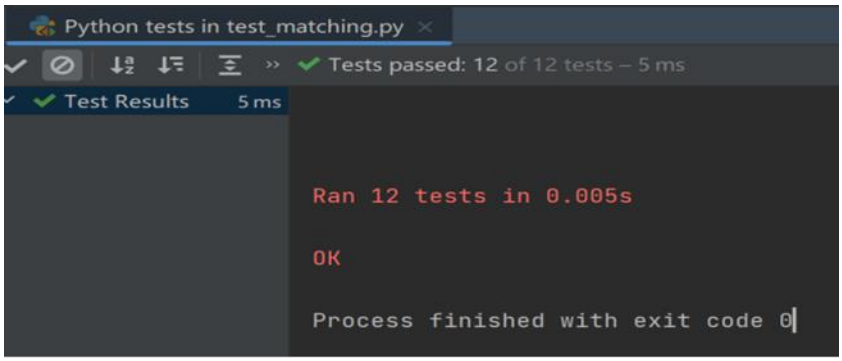
```
Dormitory=[]
DormitoryCode=[]
total=[num for num in range(len(personScoreList))]  #the set of number
if len(personScoreList) % 2 == 0:
    while len(Dormitory) < len(personScoreList)/2:
        Dor=(sheet1['A'+str(dormitoryScore[int(Score[0][0])].person1+2)].value,sheet1['A'+str(dormitoryScore[int(Score[0][0])].person2+2)].value)
        if checkPerson(DormitoryCode,dormitoryScore[int(Score[0][0])].person1,dormitoryScore[int(Score[0][0])].person2):
            DormitoryCode.append((dormitoryScore[int(Score[0][0])].person1,dormitoryScore[int(Score[0][0])].person2))
            Dormitory.append(Dor)
        Score.pop(0)
else:
    while len(Dormitory) < (len(personScoreList)-1)/2:
        Dor=(sheet1['A'+str(dormitoryScore[int(Score[0][0])].person1+2)].value,sheet1['A'+str(dormitoryScore[int(Score[0][0])].person2+2)].value)
        if checkPerson(DormitoryCode,dormitoryScore[int(Score[0][0])].person1,dormitoryScore[int(Score[0][0])].person2):
            DormitoryCode.append((dormitoryScore[int(Score[0][0])].person1,dormitoryScore[int(Score[0][0])].person2))
            Dormitory.append(Dor)
        Score.pop(0)
    for m in DormitoryCode:
        for n in m:
            total.remove(n)
    Dormitory.append(sheet1['A'+str(total[0]+2)].value)


return Dormitory
```

6 TESTING AND EVALUATION

Algorithm part:

In order to test the availability of algorithm, we used unittest framework to verify the functions in the code. We created two person's instance and valid the result. The matching score is the same as expected. All the tests are passed. As shown in the following picture:



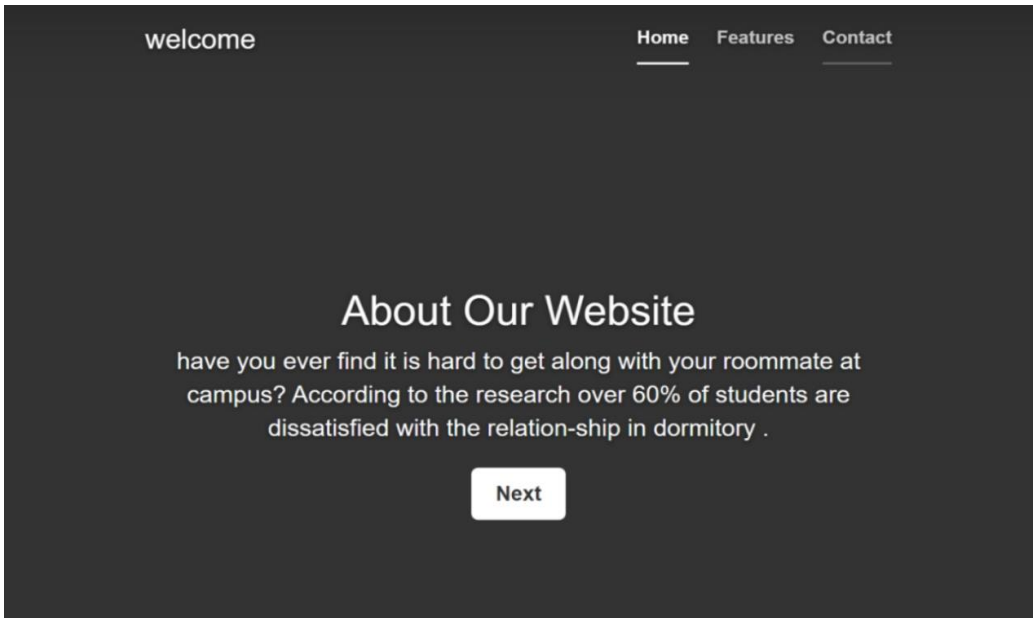
The algorithm basically meets the original requirements. We can complete two person's matching among several people according to their information.

7 CONCLUSION AND FUTURE WORK

At present, our program can only realize dormitory matching between two people. In the future, we hope to expand it to more dormitory allocation for more than two people (such as four people). In addition, in order to protect students' privacy, we hope that in the future, when reporting the results, we will only publish the results for the corresponding dormitory of the user, and will not see the results of other users. Many additional services can be added to the site, such as pre-check-in procedures, student feedback collection and so on.

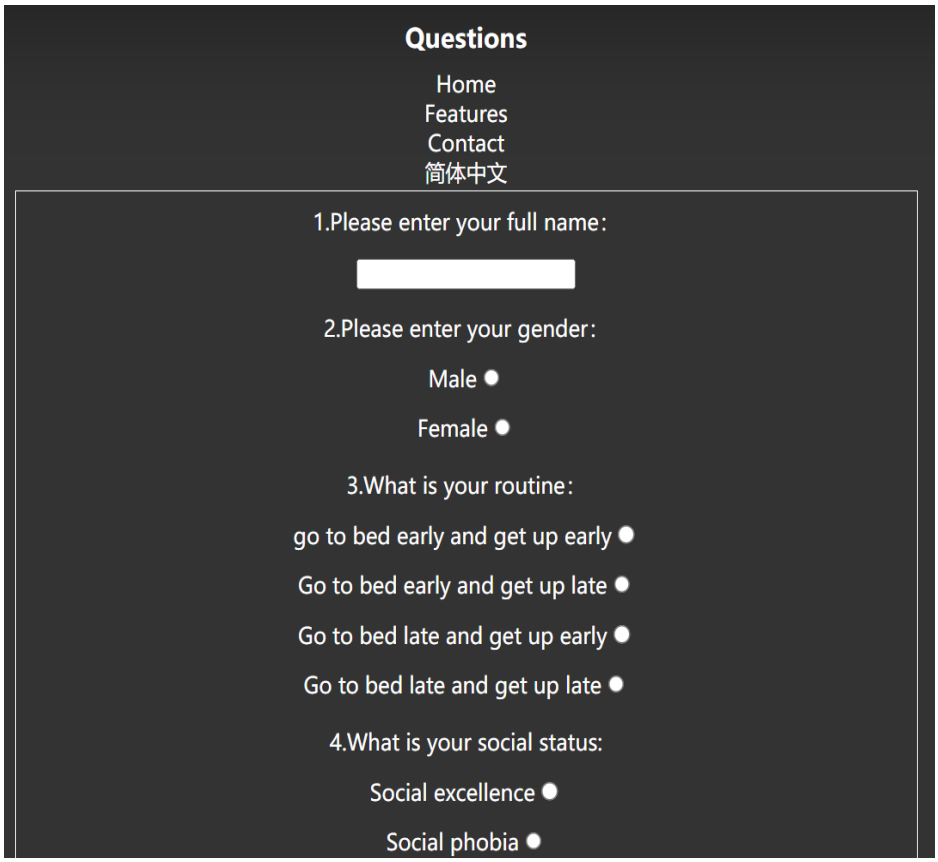
Appendix 1 - User Manual

1. Enter website address: XXXX. Then you can see the following page:

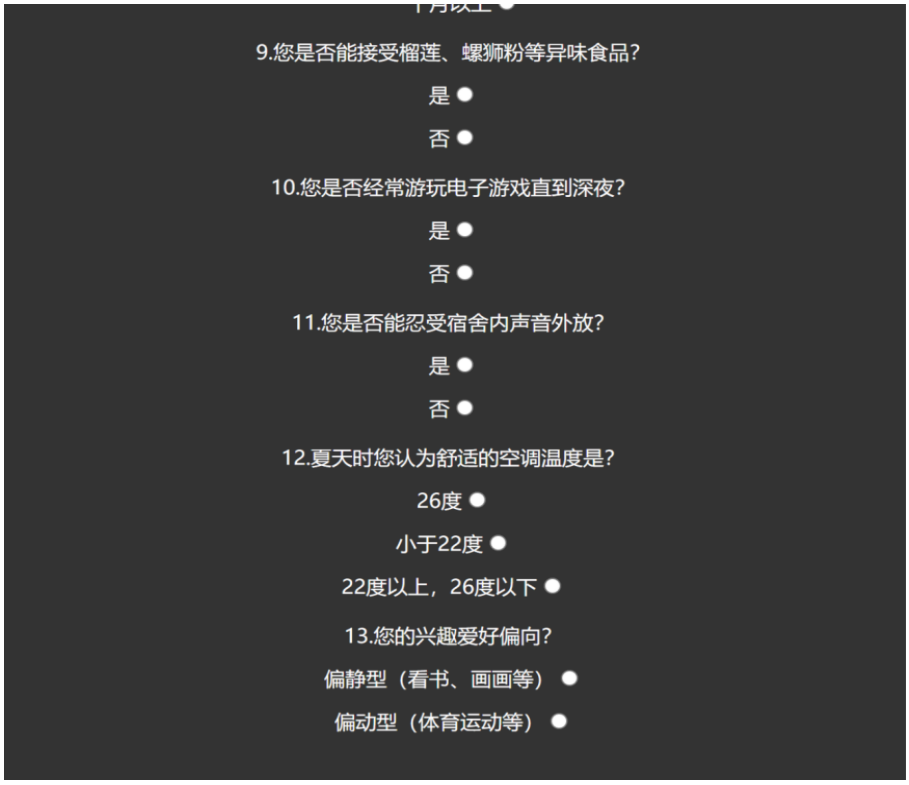


2. Click the “next” button. Then you can open the questionnaire. As shown in the following picture:

(If you want to change the language into English.just click the "English" tab.)



3. Next filling the blank and choosing the options according to your fact. The first 13 questions except for the first are the single choice questions.



4. After you finish all the choices, you are required to enter a weight for each question you have answered before representing how you value this aspect of habit. What you need to pay attention is that the total weight should be 100 points.

5. If you want to reset your options, click the white button. Finally, click the black button and submit the questionnaire.

6. Please wait for a time. When all students finish submitting, we will return matching result on the website.

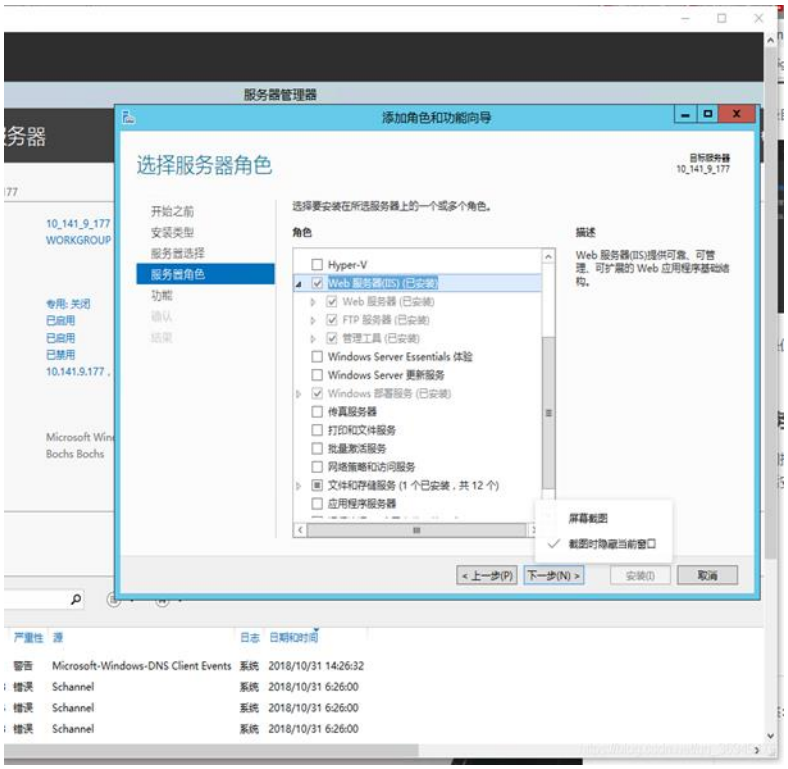
Appendix 2 - Maintenance Manual

1. We want everyone to be able to access our own website through the domain name, first we need to prepare a domain name, and a server Domain name can be purchased through Tencent Cloud, Ali Cloud, etc. I use the Tencent cloud server and Tencent Cloud to buy the domain name.

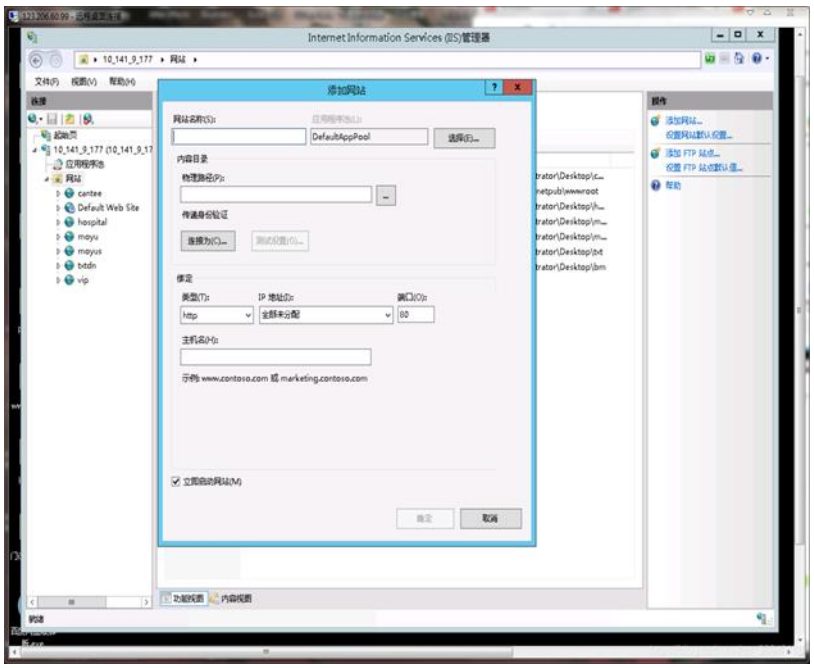
2. Log in to the website where you purchased the domain name and resolve the domain name.

Record the value as 123.206.60.99, that is, fill in the ip address of the website.

3. Server Configuration. We press win+R key, enter mstsc, fill in the public ip address of the server, then log in the server, the role is Administrator, the password is your server login password, install IIS manager after entering the server, we open the server manager, click management, add roles and functions, and then press the wizard, go down by default. Select all the boxes I checked in the image below and add them.



4. After the addition, open IIS Manager. Then, copied the website we made to the server.



5. Then everyone can access our website through internet.