**《软件工程综合课程设计》报告**

题目——高铁查询系统

学 号：16103218

姓 名：魏 远

时 间：2019年9月5日

**设计说明**

本系统主要是做一个用户可以查到从出发地到目的地的高铁，并能获取到出发时间，到达时间，总行程时间等信息。还可以对自己的需求进行排序以获取适合自己行程安排的列车。

户根据自己的需求输入出发地和目的地，系统调用数据库中的车站表提取出经过出发车站且途经目的车站的列车号，再由列车号调用数据库中的各列车表获取列车从出发地的出发时间、出发车站、到站时间、到达车站、列车号等数据，

系统将其制成表格的形式展示出来。

用户获取到系统展示的数据后，可以依据自己的需求对数据进行出发最早、出发最晚、时长最短排序。选择适合自己的最佳路线。还可以在当前页面再次查询。对后续行程安排进行预知。

**1 需求分析**

顶层数据流图说明了整体而言的数据流向，用户输入条件信息，系统先查询数据库中的车站表信息，获取到列车号之后再查询数据库中列车号表得到用户需要的信息。



图 1-1顶层数据流图

系统查询自子数据流图更加详细的说明了系统调用数据库的方式，数据库中有一张所有高铁列车的从始发站到终点站的trains表，有600多张车站表，3000多张列车表。系统先依据用户的出发地和目的地调用出发车站表，获取途径出发地的列车及后面会经过的车站数据。再根据获取到的列车数据调用列车表(表明即是列车号)。数据库返回列车号、出发车站、出发时间、目的地、到达时间、总时长数据。系统制成表展示给用户。



图1-2 系统查询子数据流图

层次框图比较好理解，就是系统的层次结构，我这里以界面和数据展开。



图1-3层次框图

这个系统就是一个简单的查询系统，即用户提供条件获得数据，然后对数据进行进行排序，即一下三个用例

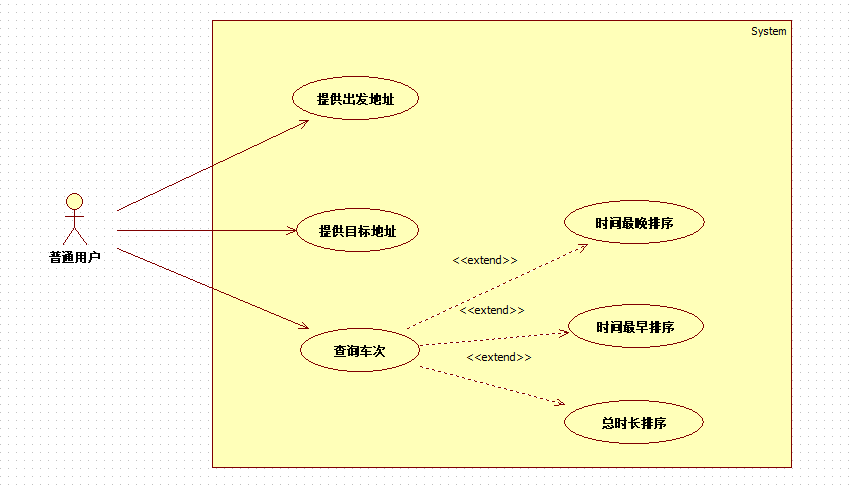


图1-4 用例图

**2 概要设计部分**

用户输入出发车站和目的车站，系统得到出发车站后判断出发车站是否有高铁经过，如果没有就要求用户重新输入出发车站。如果有就依据出发车站名调用该表获得列车后面会在目的车站停靠的列车号。再依据列车号调用该列车表获取到用户需要的列车信息，再返回给系统制成表格形式反馈给用户。

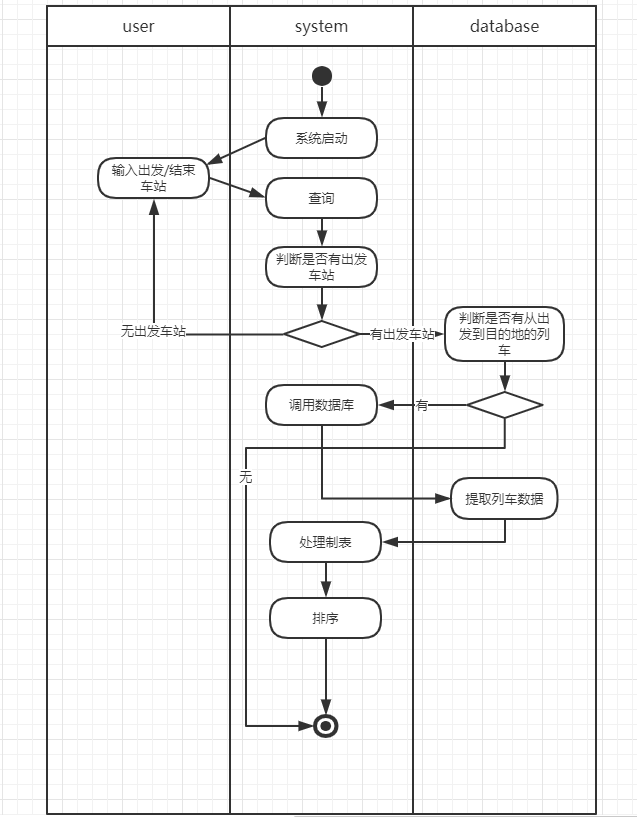


图2-1活动图

用户启动系统提供自己的出发车站和目的车站，系统判断所有高铁车站是否包含出发车站，如果没有就提示出发车站没有高铁，如果有就调用数据库中的高铁车站表，提取出后续停靠车站包含目的地车站的列车号返回给系统，系统调用数据库中列车号表提取出出发时间，结束时间，总时长等信息制成表格展示给用户。

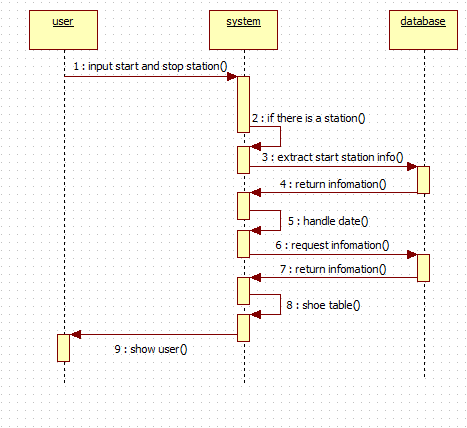


图2-2 时序图

**3 详细设计部分**

系统中有User,Example,Database三个实体类。User类中有start\_station,stop\_station两个字段供输入，Example类中含有(Qlable,HpushButton,QHBoxLayout,QlineEdit,QtableWidget,Qconobox)等各种属性，具有\_\_init\_\_(),initUI(),search(), center()

show\_train(),cancel(),station\_no\_train(),stop\_no\_train()等方法。数据库中提供用户需要的各种信息，包括列车号，出发时间，结束时间，出发车站，结束车站，总时长。

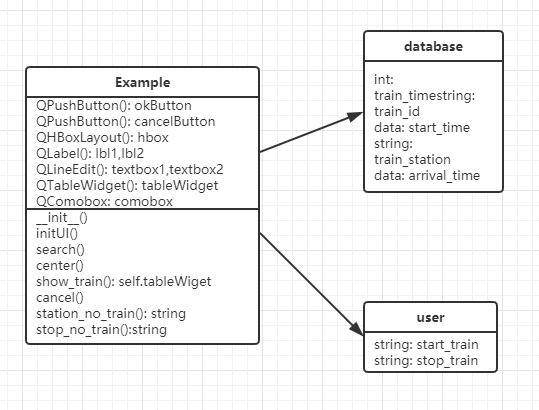


图3-1 类图

在这里我把三个状态图放在一起容易理解，User就是处于输入玩查询条件后等待系统的回应，然后选择排序等待排序结果，Example类处于调用数据库的过程中，对数据库中的数据进行处理。数据库一直处于被调用和查询状态。

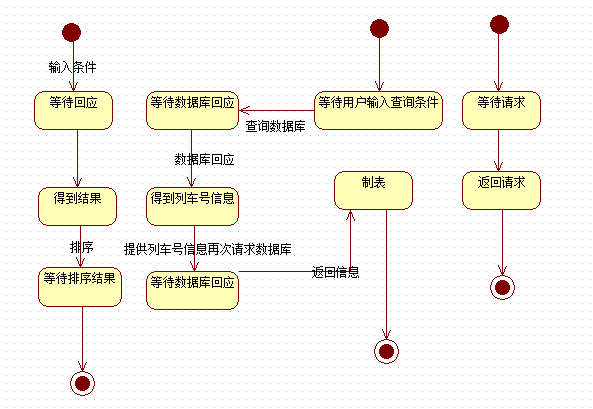


图3-2状态图

**4 实现说明部分（后台）**

1：爬取数据：

1. **def** get\_Trainid():
2. headers = {
3. "Accept-Language": "zh-CN,zh;q=0.9",
4. "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
5. }
7. result = requests.get(url="http://www.jt2345.com/huoche/checi", headers=headers)
8. html = result.text.encode("ISO-8859-1")
9. soup = BeautifulSoup(html.decode("gbk"),"html.parser")
10. columns = soup.find("center")
11. results = columns.find("table").find\_all("tr")
13. **del** results[0].contents[0]
14. **del** results[644].contents[-1]
16. **for** result **in** results:
17. get\_Info(result)
18. **def** get\_Info(result):
20. **for** i **in** result:
22. href = "http://www.jt2345.com" + i.find("a").attrs["href"]
24. train\_name\_info = {
25. "id":i.string,
26. "href":href,
27. }
29. all\_trains.append(train\_name\_info)

爬取http://www.jt2345.com 网站里的数据，获取到所有车次和车次的详细信息的链接。

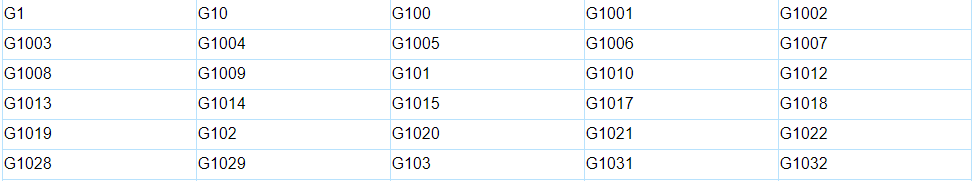


图4-1 要爬取的数据截图举例



图4-2爬取下来的部分数据

1. **def** parse(stations,station\_id):
3. train\_info = []
5. **for** station **in** stations:
7. train\_time = station.find\_all("td")[0].string
8. train\_station = station.find\_all("td")[1].string
9. arrival\_time = station.find\_all("td")[2].string
10. driving\_time = station.find\_all("td")[3].string
11. residence\_time = station.find\_all("td")[4].string
13. train\_times = {
14. "station\_id": station\_id,
15. "train\_time": train\_time,
16. "train\_station": train\_station,
17. "arrival\_time": arrival\_time,
18. "driving\_time": driving\_time,
19. "residence\_time": residence\_time
20. }
22. train\_info.append(train\_times)
24. **print**(train\_info)
26. train\_all\_infos.append(train\_info)

根据链接爬取所有列车的停靠站信息(station\_id,tain\_time,train\_station,arrival\_time,driving\_time,residence\_time)

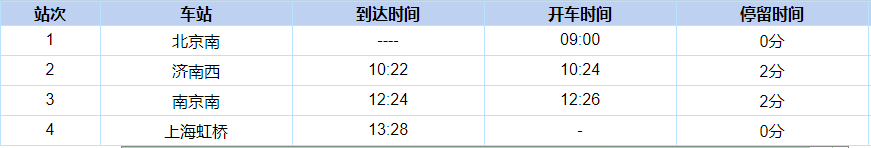


图 4-3要爬取的数据截图举例

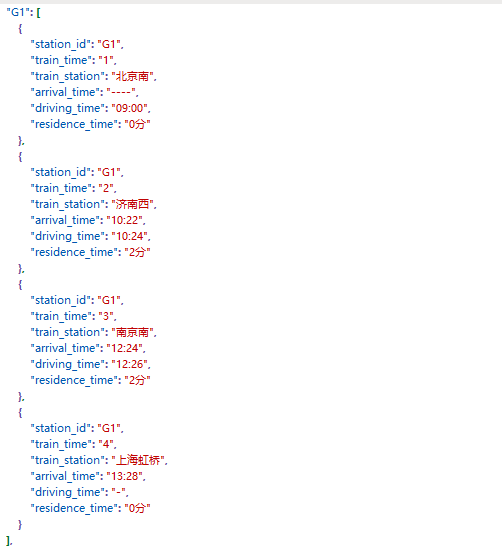


图 4-4爬取的数据部分截图

1. **def** parse\_html(infos):
2. train\_id = infos[0].find\_all("td")[2].string
3. model = infos[1].find\_all("td")[1].string
4. start = infos[2].find\_all("td")[1].string
5. start\_time = infos[3].find\_all("td")[1].string
6. stop\_time = infos[4].find\_all("td")[1].string
7. total\_time = infos[5].find\_all("td")[1].string
8. stop = infos[6].find\_all("td")[1].string
10. train = {
11. "train\_id": train\_id,
12. "model": model,
13. "start": start,
14. "start\_time": start\_time,
15. "stop\_time": stop\_time,
16. "total\_time": total\_time,
17. "stop": stop
18. }
19. **print**(train)
20. general\_trains.append(train)

获取列车的全程信息(出发车站，出发时间，到达时间，全程时间，终点站)



图 4-5爬取的数据部分截图



图 4-6爬取的数据部分截图

1. **def** handle\_data():
3. all\_Infos = {}
5. **for** station **in** all\_stations.keys():
6. all\_Infos[station] = {}
8. **for** station **in** all\_Infos.keys():
10. **for** results **in** all\_stations[station]:
11. all\_Infos[station][results] = []
13. with open("G:\软件工程课程设计\全国高铁信息\高铁具体信息.json", "r", encoding="utf-8") as jf:
15. results = json.load(jf)
16. **for** station\_start **in** all\_Infos:
18. **for** train **in** all\_Infos[station\_start]:
19. stations = []
21. **if** len(results[train]) > 10:
22. **pass**
24. **else**:
25. **for** station\_info **in** results[train]:
27. stations.append(station\_info["train\_station"])
29. **for** station **in** stations:
31. **if** station == station\_start:
32. i = stations.index(station)
34. **for** n **in** stations[i:]:
35. all\_Infos[station\_start][train].append(n)
37. **for** station **in** all\_Infos:
38. **for** train **in** list(all\_Infos[station].keys()):
39. **if** len(all\_Infos[station][train]) <= 1:
40. **del** all\_Infos[station][train]
41. **else**:**pass**
42. times = []
44. **for** station **in** list(all\_Infos.keys()):
45. **if** len(all\_Infos[station]) == 0:
46. **del** all\_Infos[station]
48. **for** station **in** list(all\_Infos.keys()):
49. **if** len(all\_Infos[station]) == 0:
50. **print**(station)

53. **return** all\_Infos

获取所有车站经过的列车号及列车之后途径的车站，之所以这样做，是为了以空间换时间，达到查询速度正常的效果。否则直接将数据放在数据库里面查询的时间太慢无法接受

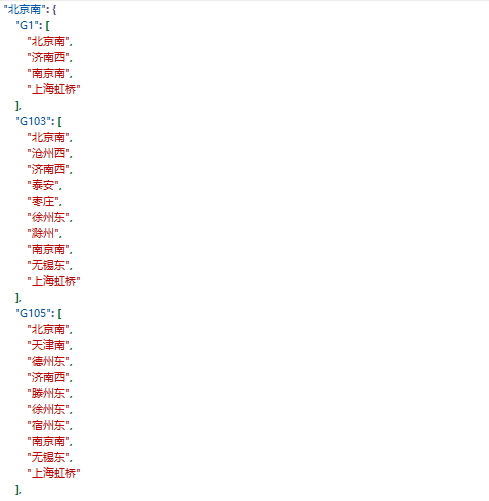


图 4-7 数据处理部分截图

1. create\_table\_all\_station = """\
2. create table example(
3. train\_id varchar(10) PRIMARY KEY,
4. start\_station VARCHAR(10),
5. stop\_station1 varchar(10),
6. stop\_station2 varchar(10),
7. stop\_station3 varchar(10),
8. stop\_station4 varchar(10),
9. stop\_station5 varchar(10),
10. stop\_station6 varchar(10),
11. stop\_station7 varchar(10),
12. stop\_station8 varchar(10),
13. stop\_station varchar(10)
14. )
15. """
17. **def** create\_table():
19. with open("G:\软件工程课程设计\全国高铁信息\高铁车站出发信息.json", "r", encoding="utf-8") as jf:
20. results = json.load(jf)
21. **print**(results.keys())
23. **try**:
24. with conn.cursor() as cursor:
25. **for** station **in** results.keys():
26. **print**("create\_table" + str(station))
27. sql = create\_table\_all\_station.replace("example",station)
28. cursor.execute(sql)
30. **finally**:
31. conn.close()

创建所有的车站表(列车号，当前车站，后续停靠车站)

1. insert\_to\_station\_tables = """\
2. INSERT INTO example(train\_id,start\_station,stop\_station1,stop\_station2,stop\_station3,stop\_station4,stop\_station5,
3. stop\_station6,stop\_station7,stop\_station8,stop\_station)
4. VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)
5. """
7. **def** insert\_to\_station\_table(): # 向车站数据库中插入车次数据
8. **try**:
9. with conn.cursor() as cursor:
10. with open("G:\软件工程课程设计\全国高铁信息\高铁车站出发信息.json", "r", encoding="utf-8") as jf:
11. results = json.load(jf)
13. **for** station **in** results.keys():
14. sql = insert\_to\_station\_tables.replace("example",station)
16. **for** train **in** list(results[station].keys()):
17. station\_train = []
18. station\_train.append(train)
20. **if** train **in** results[station]:
22. **for** station\_stop **in** results[station][train]:
23. station\_train.append(station\_stop)
25. **if** len(station\_train) < 11:
27. **for** i **in** range(11-len(station\_train)):
28. station\_train.append("null")
30. **print**(station\_train)
32. cursor.execute(sql,(station\_train))
33. conn.commit()
34. **finally**:
35. conn.close()

向所有车站表中插入数据

1. create\_train\_table\_sql = """\
2. create table example(
3. train\_time int PRIMARY KEY,
4. train\_id VARCHAR(10),
5. train\_station VARCHAR(10),
6. arrival\_time varchar(10),
7. driving\_time varchar(10),
8. residence\_time varchar(10)
9. )
10. """
12. **def** create\_table\_train():
13. **try**:
14. with conn.cursor() as cursor:
16. with open("G:\软件工程课程设计\全国高铁信息\高铁具体信息.json", "r", encoding="utf-8") as jf:
17. results = json.load(jf)
18. # print(results.keys())
20. **for** train **in** list(results.keys()):
21. sql = create\_train\_table\_sql.replace("example",train)
22. cursor.execute(sql)
24. conn.commit()
25. **finally**:
26. conn.close()

创建所有的列车表(列车号，停靠站次，到达时间，出发时间，停靠时间，停靠车站名)

1. insert\_to\_train\_table= """\
2. INSERT INTO example(train\_time,train\_id,train\_station,arrival\_time,driving\_time,residence\_time)
3. VALUES(%s,%s,%s,%s,%s,%s)
4. """
6. **def** insert\_to\_table\_train():
7. **try**:
8. with conn.cursor() as cursor:
9. with open("G:\软件工程课程设计\全国高铁信息\高铁具体信息.json","r",encoding="utf-8") as jf:
10. results = json.load(jf)
12. **for** train **in** results.keys():
13. **for** station **in** results[train]:
14. sql = insert\_to\_train\_table.replace("example",train)
15. list = []
16. list.append(int(station["train\_time"]))
17. list.append(station["station\_id"])
18. list.append(station["train\_station"])
19. list.append(station["arrival\_time"])
20. list.append(station["driving\_time"])
21. list.append(station["residence\_time"])
22. cursor.execute(sql,list)
23. **print**(list)
25. conn.commit()
27. **finally**:
28. conn.close()

向所有列车表中添加数据

1. **def** center(self):
2. qr = self.frameGeometry()
3. cp = QDesktopWidget().availableGeometry().center()
4. qr.moveCenter(cp)
5. self.move(qr.topLeft())

将主界面防放在屏幕中间

1. **def** search(self):
2. # print(1)
3. start = self.textbox1.text()
4. stop = self.textbox2.text()
5. **print**("正在搜索")
6. **print**(start)
7. **print**(stop)
8. **import** json
9. with conn.cursor() as cursor:
10. with open("G:\软件工程课程设计\全国高铁信息\高铁车站出发信息.json","r",encoding="utf-8") as jf:
11. results = json.load(jf)
12. stations = list(results.keys())
13. # print(stations)
14. **if** start **in** stations:
15. sql = search\_table\_station.replace("example", start)
16. cursor.execute(sql)
17. results = cursor.fetchall()
18. train\_list = []
19. **for** trains **in** results:
20. # print(trains)
21. **if** stop **in** trains:
22. train\_list.append(trains[0])
23. **if** len(train\_list) == 0:
24. # print("没有从" + start + "到" + stop + "的高铁动车")
25. self.stop\_no\_train(start,stop)
26. # pass
27. # for station\_train in trains:
28. #     print(station\_train)
29. #     if stop in station\_train:
30. #         train\_list.append(trains[0])
31. travels = []
32. **for** train **in** train\_list:
33. sql = search\_table\_train.replace("example",train)
34. cursor.execute(sql)
35. results = cursor.fetchall()
37. travel = {}
38. **for** train\_station **in** results:
39. **if** start **in** train\_station:
40. train\_id = train\_station[1]
41. start\_station = train\_station[2]
42. start\_time = train\_station[4]
43. **if** stop **in** train\_station:
44. arrival\_time = train\_station[3]
45. time\_start = start\_time.split(":")
46. time\_arrival = arrival\_time.split(":")
47. # print(time\_start)
48. # print(time\_arrival)
49. time\_total = []
51. **if** int(time\_start[0])<=int(time\_arrival[0]) **and** int(time\_start[1])<int(time\_arrival[1]):
53. time\_total.append(int(time\_arrival[0])-int(time\_start[0]))
54. time\_total.append(int(time\_arrival[1])-int(time\_start[1]))
55. **if** int(time\_start[0])<=int(time\_arrival[0]) **and** int(time\_start[1])>int(time\_arrival[1]):
57. time\_total.append(int(time\_arrival[0]) - int(time\_start[0]) - 1)
58. time\_total.append(int(time\_arrival[1]) - int(time\_start[1]) + 60)
60. **if** int(time\_start[0])>int(time\_arrival[0]) **and** int(time\_start[1])<int(time\_arrival[1]):
61. time\_total.append(int(time\_arrival[0]) - int(time\_start[0]) + 24)
62. time\_total.append(int(time\_arrival[1]) - int(time\_start[1]))
63. **if** int(time\_start[0])>int(time\_arrival[0]) **and** int(time\_start[1])>int(time\_arrival[1]):
64. time\_total.append(int(time\_arrival[0]) - int(time\_start[0]) + 24 -1)
65. time\_total.append(int(time\_arrival[1]) - int(time\_start[1]) + 60)
66. **else**:**pass**
67. total\_time = "{}:{}".format(time\_total[0],time\_total[1])
68. travel = {
69. "train\_id":train\_id,
70. "start\_station":start\_station,
71. "start\_time":start\_time,
72. "arrival\_time":arrival\_time,
73. "stop":stop,
74. "total\_time":total\_time
75. }
76. travels.append(travel)
78. self.show\_trains(travels)
80. **else**:
81. self.station\_no\_train(start)

搜索所有的符合条件的信息，排序并把它显示，传递给显示函数

1. **def** show\_trains(self,travels):
3. self.tableWidget = QTableWidget(self)  # 创建一个表格
4. self.tableWidget.setEditTriggers(QTableWidget.NoEditTriggers)
5. self.tableWidget.move(50, 100)
6. self.tableWidget.resize(700, 250)
8. self.tableWidget.setRowCount(len(travels))
9. self.tableWidget.setColumnCount(6)
11. self.tableWidget.setHorizontalHeaderLabels(['列车', '出发站', '出发时间', '到达时间', '到达站','总时长'])
13. **if** self.combobox.currentText() == "出发最早":
14. start\_times = []
15. **for** travel **in** travels:
16. start\_time = travel["start\_time"].split(":")
17. start\_time = int(start\_time[0] + start\_time[1])
18. start\_times.append(start\_time)
19. time\_time = dict(zip(range(len(travels)), start\_times))
20. # start\_times = list(time\_time.values())
21. # start\_times.sort()
22. results = sorted(time\_time.items(), key=**lambda** item: item[1],reverse=False)
23. sort\_time = []
24. **for** result **in** results:
25. sort\_time.append(result[0])
27. **for** result **in** results:
28. list = []
29. list.append(travels[result[0]]["train\_id"])
30. list.append(travels[result[0]]["start\_station"])
31. list.append(travels[result[0]]["start\_time"])
32. list.append(travels[result[0]]["arrival\_time"])
33. list.append(travels[result[0]]["stop"])
34. list.append(travels[result[0]]["total\_time"])
35. **for** info **in** list:
36. self.tableWidget.setItem(sort\_time.index(result[0]), list.index(info), QTableWidgetItem(info))


40. **elif** self.combobox.currentText() == "出发最晚":
41. start\_times = []
42. **for** travel **in** travels:
43. start\_time = travel["start\_time"].split(":")
44. start\_time = int(start\_time[0] + start\_time[1])
45. start\_times.append(start\_time)
46. time\_time = dict(zip(range(len(travels)), start\_times))
48. results = sorted(time\_time.items(), key=**lambda** item: item[1], reverse=True)
49. sort\_time = []
50. **for** result **in** results:
51. sort\_time.append(result[0])
53. **for** result **in** results:
54. list = []
55. list.append(travels[result[0]]["train\_id"])
56. list.append(travels[result[0]]["start\_station"])
57. list.append(travels[result[0]]["start\_time"])
58. list.append(travels[result[0]]["arrival\_time"])
59. list.append(travels[result[0]]["stop"])
60. list.append(travels[result[0]]["total\_time"])
61. **for** info **in** list:
62. self.tableWidget.setItem(sort\_time.index(result[0]), list.index(info), QTableWidgetItem(info))
64. **elif** self.combobox.currentText() == "时长最短":
65. start\_times = []
66. **for** travel **in** travels:
67. start\_time = travel["total\_time"].split(":")
68. start\_time = int(start\_time[0] + start\_time[1])
69. start\_times.append(start\_time)
70. time\_time = dict(zip(range(len(travels)), start\_times))
72. results = sorted(time\_time.items(), key=**lambda** item: item[1], reverse=False)
73. sort\_time = []
74. **for** result **in** results:
75. sort\_time.append(result[0])
77. **for** result **in** results:
78. list = []
79. list.append(travels[result[0]]["train\_id"])
80. list.append(travels[result[0]]["start\_station"])
81. list.append(travels[result[0]]["start\_time"])
82. list.append(travels[result[0]]["arrival\_time"])
83. list.append(travels[result[0]]["stop"])
84. list.append(travels[result[0]]["total\_time"])
85. **for** info **in** list:
86. self.tableWidget.setItem(sort\_time.index(result[0]), list.index(info), QTableWidgetItem(info))
87. self.combobox.show()
88. self.tableWidget.show()

向用户显示图表，及排序下拉框

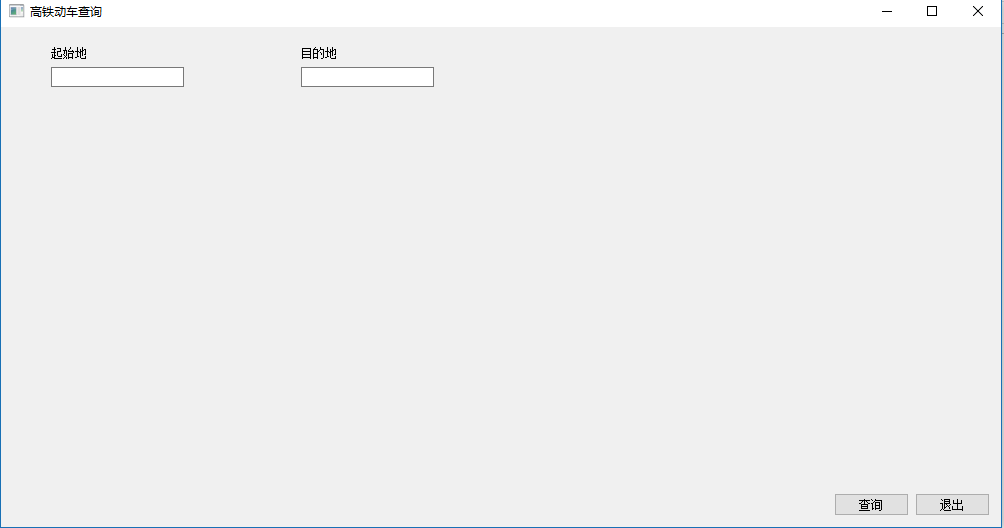


图 4-8 运行截图 – 1

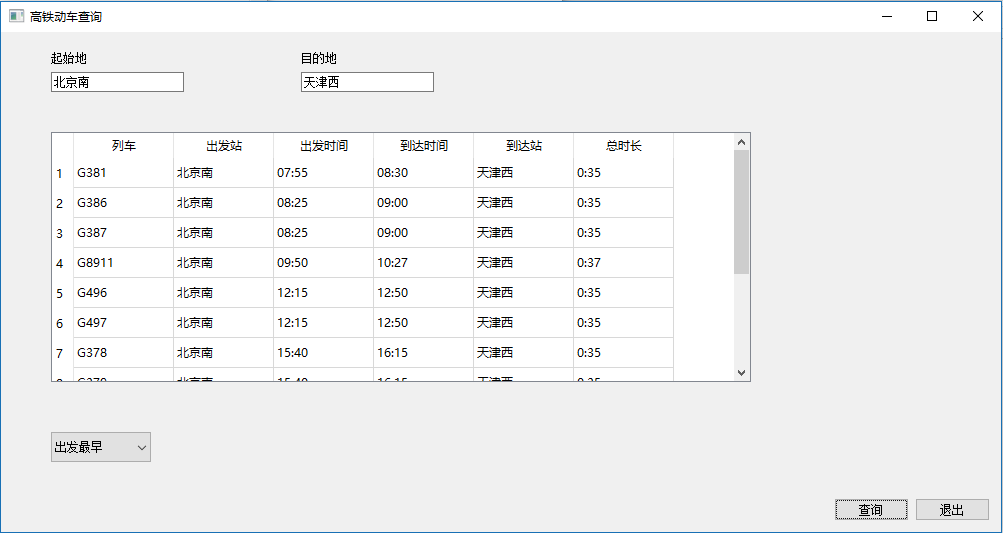


图 4-8 运行截图 – 2

**5 总结**

本次实现了一个高铁列车时刻查询，用户输入出发地和目的地后可以查询到两点之间的所有列车班次和班次的详细信息，并进行用时，出发点，到达点的排序。一开始有这个想法的时候没觉得多困难，应该是把数据拿到手，然后做基本的查询和列出信息就可以了。但是现实并没有这么简单，甚至在拿数据上都有困难，最后用了爬虫的方式拿到了数据。一共有4000辆左右的列车。一开始打算直接将数据放进数据库中直接进行查询。但是数据太多可能回导致查询数据太慢，最后决定将数据进行一系列处理。我是提取出所有的车站并找出经过的列车及后续停靠车站。一共有600多辆列车，最后一共在数据库里面放入4000多张列车表，600多张车站表。这次的课设让我加深了爬虫的技术，增强了对数据和数据库处理能力。并对软件工程的总体流程更加熟悉。写我这个系统，最重要的是逻辑一定要清晰，对数据的处理逻辑很重要，所以写程序之前一定要做一个思维导图，这样会让写程序时思路更加清晰。