Online Flight Ticket

System

Bug report



|  |  |
| --- | --- |
| **Name** | **Student Number** |
| ZHAO Yiwei | 56641836 |
| LI Xiangyu | 56641559 |
| WANG Chenchen | 56344983 |
| WANG Yian | 56641105 |
| XIE Yifei | 56641547 |
| YANG Jingxian | 56644285 |

**TABLE OF CONTENTS**

1. **Description**
2. **Template**
3. **Explanation** 
   1. **Describe the Bug**
   2. **Possible Related Components**
   3. **To Reproduce**
   4. **Expected Behavior**
   5. **Screenshots**
   6. **Environment**

**3.7 Additional Context**

**3.8 Assignees**

**3.9 Communication Between Members**

1. **Issues**

**4.1 Issue 1**

**4.2 Issue 2**

**4.3 Issue 3**

**4.4 Issue 4**

**4.5 Issue 5**

**4.6 Issue 6**

**4.7 Issue 7**

**1. Description**

Bugzilla is a useful website which we use to post our bugs. It is also recommended by our teacher. Firstly, I create our Group 5 product category on the “CS3343 Bugzilla Home Page”. Then, a template is created for our IT testers to post their own bugs. When a bug report is received, our project manager will assign a developer to fix the bug. And our developer will handle the bugs with higher priority first. After a bug is fixed, we can close it manually or automatically close it in a commit.

**2. Template**

|  |  |  |  |
| --- | --- | --- | --- |
| **[CCCS-1]** | | | |
| **Status:** | solved | | |
| **Reporter:** | Yiwei Zhao | **Assignee:** | Yiwei Zhao |
| **Environment:** | Java 17, windows 10 | | |
| **Resolution:** | We should pass the variable “destination” to function computeDoubleRoute(). | | |
| **Steps to Reproduce:**   1. Run the software.exe 2. Open the buy ticket page. 3. Find all the flights from Beijing to Shanghai which pass by the transfer station Chongqing. 4. The system returns flights including not only Beijing to Shanghai but also Beijing to Guangzhou or other cities.   **Expected output:**  Because the Chongqing is a transfer station, so the number of results should be only one.  **Actual output:** the program returns two results. | | | |

**3. Explanation**

Here is the detailed explanation of each part of our bug report.

3.1 Describe the Bug

Testers have to provide detailed description of the issue. Apart from giving an overall description which describe the summary more detailed, the description should include reproduce steps, expected result and actual result. Reproduce steps required testers to clearly mention the minimal sets of necessary steps to trigger the bug. Expected result is how the application should behave on the mentioned steps without bugs. For the actual result, it is what the application actually done on running the mentioned steps.

3.2 Possible Related Components

The reporter is expected to predict where the bug might be in our project structure. This is optional because sometimes it is difficult to predict, and sometimes the reporter may not be responsible for the area where the bug occurs.

e.g., src/TicketSystem.Database

3.3 To Reproduce

The reporter must give detailed steps on how to reproduce the bug, as this is important for the developer to locate the bug.

3.4 Expected Behavior

The reporter can explain what the program will behave if no bug is present or what is their “ideal” behavior. This is optional.

e.g. When method iUserDAO.queryUser() or iUserDAO.changePwd() is true, it will return a new constructor, otherwise it will return null.

3.6 Environment

This part is to state the environment in which the issue occurs. The environment should include the hardware platform like PC. Also the testers should mention the operating system with the OS version where they found the bug, such as Window 8, Window XP, etc.

3.7 Additional Context

This is for the reporter to explain any additional information and is optional.

3.8 Assignees

This is shown in the sidebar on the right. Once a bug is reported, it will be assigned to one ormore developers to fix.

3.9 Communication Between Members

This is a great feature of GitHub Issues. After a bug is reported, members can communicate on the bottom of this bug to express their own ideas or provide additional information.

1. **Issues**

|  |  |  |  |
| --- | --- | --- | --- |
| **[CCCS-1]** | | | |
| **Status:** | solved | | |
| **Reporter:** | Yiwei Zhao | **Assignee:** | Yiwei Zhao |
| **Environment:** | Java 19, windows 10 | | |
| **Resolution:** | We should pass the variable “destination” to function computeDoubleRoute(). | | |
| **Steps to Reproduce:**   1. Run the software.exe 2. Open the buy ticket page. 3. Find all the flights from Beijing to Shanghai which pass by the transfer station Chongqing. 4. The system returns flights including not only Beijing to Shanghai but also Beijing to Guangzhou or other cities.   **Expected output:**  Because the Chongqing is a transfer station, so the number of results should be only one.  **Actual output:** the program returns two results. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **[CCCS-2]** | | | |
| **Status:** | solved | | |
| **Reporter:** | Yiwei Zhao | **Assignee:** | Yiwei Zhao |
| **Environment:** | Java 19, windows 10 | | |
| **Resolution:** | Correct the getFlightIndex() in Flight.java. | | |
| **Steps to Reproduce:** 1. Run the software.exe  2. Open the buy ticket page  3. Find all the flights from Beijing to Shanghai.  4. The system returns flights that every flight has a Fid at the front.  **Expected output:**  The form of the returned Fid should be like “MU2456”.  **Actual output:** the program returns the Fid instead of FlightIndex. So it returns the first flight with Fid 1, the second flight with Fid 2. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **[CCCS-3]** | | | |
| **Status:** | solved | | |
| **Reporter:** | Yiwei Zhao | **Assignee:** | Yiwei Zhao |
| **Environment:** | Java 19, windows 10 | | |
| **Resolution:** | Add the sort by time function in FlightDAO.java. | | |
| **Steps to Reproduce:** 1. Run the software.exe  2. Open the buy ticket page  3. Find all the tickets from Beijing to Shanghai  4. Sort them by time.  5. The system return the new list of flights in a new order.  **Expected output:**  The order should be sorted by time.  **Actual output:** however, it is sorted by cost instead of time. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **[CCCS-4]** | | | |
| **Status:** | solved | | |
| **Reporter:** | Jingxian Yang | **Assignee:** | Jingxian Yang |
| **Environment:** | Java 19, windows 10 | | |
| **Resolution:** | We should add a branch to return null in the Login() and changePwd() in the if conditional sentences. | | |
| Steps to Reproduce:  In User.java, run the function login() and changePwd(), we can see the return results of if conditional sentences.  Expected output:  When method iUserDAO.queryUser() or iUserDAO.changePwd() is true, it will return a new constructor, otherwise it will return null.  Actual output:  It only can return new constructor when if conditional sentences are true. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **[CCCS-5]** | | | |
| **Status:** | solved | | |
| **Reporter:** | Jingxian Yang | **Assignee:** | Jingxian Yang |
| **Environment:** | Java 19, windows 10 | | |
| **Resolution:** | We should add if conditional sentences if(rs==null) in queryOrderByUsername() and addOrder() to judge whether the result is null. | | |
| Steps to Reproduce:  In Order.java, run the function queryOrderByUsername() and addOrder(), we can see the return results of executions.  Expected output:  When executing the queryOrderByUsername() and addOrder(), when the result is not null, we will see rsToAl(rs), otherwise it will return null.  Actual output:  We can only get the rsToAl(rs) and when the rs==null, it fails. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **[CCCS-6]** | | | |
| **Status:** | solved | | |
| **Reporter:** | Yiwei Zhao | **Assignee:** | Yiwei Zhao |
| **Environment:** | Java 19, windows 10 | | |
| **Resolution:** | Correct the getFlightIndex() in Flight.java. | | |
| **Steps to Reproduce:** 1. Run the software.exe  2. Open the buy ticket page.  3. Find all the flights from Beijing to Shanghai at 9 o’clock on December 1st.  4. Sort the flights by time.  5. The system returns all the flights from Beijing to Shanghai.  **Expected output:**  The flights should only contain flights at least 1 hour later than the search time.  **Actual output:** the program returns flights including those already taking off. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **[CCCS-7]** | | | |
| **Status:** | solved | | |
| **Reporter:** | Yiwei Zhao | **Assignee:** | Yiwei Zhao |
| **Environment:** | Java 19, windows 10 | | |
| **Resolution:** | Correct the addOrder() function in Order.java. | | |
| **Steps to Reproduce:** 1. Run the software.exe  2. Open the buy ticket page.  3. Find all the flights from Beijing to Shanghai at 9 o’clock on December 1st.  4. Choose one flight ticket and buy it.  5. Finish the purchase procedure.  **Expected output:**  After the purchase procedure, the number of remaining tickets should be reduced by the purchased number.  **Actual output:** the number of tickets left remains the same. | | | |