**PROGRAM**: Add Client Order **RELEASE**: 1 **VERSION**: 1

**REPORT** **TYPE** (1 TO 6): 1 **SEVERITY** (1 to 3): 1 **DATE: 03-04-16**

*1 – Coding Error 4 – Documentation 1 - Fatal*

*2 – Design Issue 5 – Hardware 2 - Serious*

*3 – Suggestion 6 – Query 3 – Minor*

***Attachment Included****? (Y/N): N* ***REPORTED BY: Wayenard Sey***

*If yes, describe:*

***PROBLEM SUMMARY: Can you reproduce the problem? Y***

|  |
| --- |
| ***Order is accepted even if no day is selected for Due Date, but no data is added to the database*** |

***PROBLEM AND HOW TO REPRODUCE IT***

|  |
| --- |
| 1. Enter all valid inputs for adding an order, but do not select a valid day  2. Click “Add Order”  3. Notice that the order is still accepted but nothing else happens  4. Order is not added to the database |

**SUGGESTED FIX (optional):**

**Check if error checking on the ‘left side’ of the Add Order window is correctly implemented since it also occurs with the Client Name field, Order Receiver field, and Month field.**

ITEMS BELOW ARE FOR USE ONLY BY THE DEVELOPERS

**STATUS:2** 1 – Open 2 – Closed

**RESOLUTION (1 TO 9): 2 RESOLUTION VERSION: 1.1**

1 – Pending 4 – Deferred 7 – Withdrawn by reporter

2 – Fixed 5 – As designed 8 – Need more info

3 – Irreproducible 6 – Can’t be fixed 9 – Disagree with suggestion

**RESOLVED BY: Rod Arceo III DATE: 03-04-16**

**RESOLUTION TESTED BY: Wayenard Sey DATE: 03-05-16**

**TREAT AS DEFERRED (Y/N): N**

**Report Type**

* **Coding Error** - program behaves in an unexpected or wrong way. Programmer can respond to a coding error by saying program works as designed.
* **Design Issue –** program works as intended, but you disagree with the design. User interface errors are usually design issues. Programmer can disagree with a report classified as design issue.
* **Suggestion –** Nothing is wrong but your idea can improve the program
* **Documentation** – Program doesn’t behave as described in the manual or in specifications. Identify which document and page. You’re not saying which should be changed: document or software. This is the programmer’s decision to make.
* **Hardware –** Faulty interactions between the program and some type of hardware.
* **Query –** Program does something you don’t understand or don’t expect. Though you doubt that the program should work this way, if you aren’t sure, choose Query. If you’ve found a problem, the programmer will still fix it. If she doesn’t or if you don’t like her rationale for keeping the program this way, you can always submit a design issue report later.

**SEVERITY**

Bugs rated as minor tend not to be fixed. If there are lots of minor errors, write a follow-up report with Serious drawing attention to their quantity.

**ATTACHMENTS**

Test data, screenshots, etc. to support the report

**PROBLEM SUMMARY**

Summaries help everyone quickly review outstanding problems and find individual reports. This is the most carefully read part of the report! If your problem sounds less severe, managers will likely defer it. If you make the problems more severe than it is, you will gain a reputation for alarmism.

NEVER use the same problem summary for two different reports even if they are similar. DO NOT include replication steps. Treat the summary and description as separate.

**Can you reproduce the problem?**

Yes/ No/ Sometimes. If you say sometimes, be careful in describing what you tried, what you think triggers the bug and what you checked that is not triggering the bug. If you cannot reproduce a bug, you will waste everyone’s time and lose credibility! If you say NO, the programmer will likely ignore it.

**PROBLEM AND HOW TO REPRODUCE IT**

It is much better to spoon feed the programmer that to say too little. Programmers dismiss many legitimate bugs because they don’t know how to reproduce them. They postpone dealing with bugs they can’t immediately reproduce. And they waste a lot of time trying to reproduce a bug that’s not described fully.

Identify the conditions needed to recreate the bug. If you cannot reproduce it, EXPLAIN what you have tried. This is still helpful for the programmer. Describe all error messages FULLY.

**PRIORITY – Assigned by the project manager**

1. Fix Immediately – this is holding up other work.
2. Fix as soon as possible
3. Must fix before next milestone
4. Must fix before final
5. Fix if possible
6. Optional – use your own judgment

**RESOLUTION AND RESOLUTION VERSION**

* **Pending** – reports start as pending; this tells the manager to look at this report and classify and assign it.
* **Fixed** – programmer mark bugs fixed, they must indicate which version it was fixed in
* **Irreproducible** – Programmer cannot make the problem happen.
* **Deferred** – Manager acknowledges there’s a problem but chooses not to fix it in this release. Deferred is appropriate whether the bug reflects an error in coding or design.
* **As designed -** The problem reported is not an error. Behavior described is as intended.
* **Withdrawn by reporter**  - Reporter can be withdrawn.
* **Need more info** – programmer has a question
* **Disagree with suggestion –** No change will be made.
* **Duplicate**  - Close bugs that are identical NOT similar.

**TREAT as Deferred**

A bug is deferred if the project manager agrees that it's a software error but has decided that it won't be fixed in this release. Both coding errors and design errors can be deferred.

As with the PRIORITY field and the extended COMMENTS, this field reflects our belief that disagreements between project managers and testers are healthy and normal.

If you dispute a RESOLUTION of As designed, leave it alone. But answer Yes to TREAT AS DEFERRED. Thereafter this report will be included with the deferred bugs in all reports. This is almost the same as changing the programmer’s resolution, but not quite. The difference is that the Testing Group is saying, "Fine, that's your opinion and we'll leave it on record. But we get to choose what problems we show to senior management and this one's on our list." This is much more sensible than changing the Resolut i on Code.