# Software Project

CECS 544

Spring 2024

# Project Teams

- No student may do the project solo.
- A team of at least 2 people and no more than 3 is required.
- The project is likely to required many hours of cooperative work.
- You may pick your own teammates. An assignment for this task follows.
- If you desire, a teammate will be assigned for you by me.

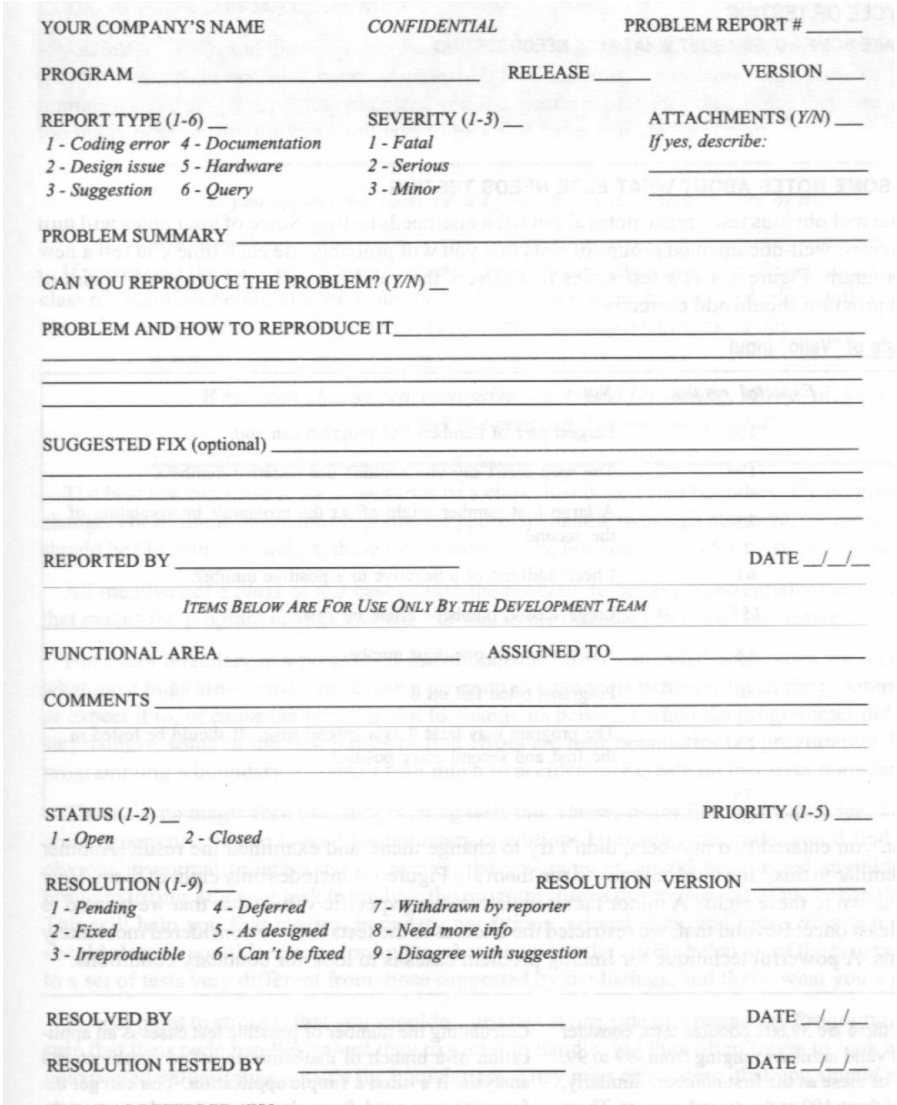
# Bughound Bug Tracking Software

- Our project is a scaled back version of a product I developed a long time ago for a little company (now defunct) in Bellevue, WA.
- The main point of the project is to have a common basis for testing on a reasonable scale of a product you are familiar with.
- Our testing will primarily be “black-box” with focus on correctly delivered functionality.
- We will also conduct testing on security, structure, usability, navigation, and compatibility with multiple host configurations. And some technical testing we will learn in Unit 4 (data flows and anomalies)

# Bughound Bug Tracking Software

- Bughound is a secure (authorized users and login required) web-based bug recording and tracking software product
- Key Features:
  - Using web browser, create, edit and update “bug” reports on multiple products
  - Store error report content in relational tables
  - Access error report content via SQL
  - Search for bugs on multiple fields
  - Facilities to add, delete or update information on program, releases, functional areas, employees, more

# Bughound Content



YOUR COMPANY'S NAME *CONFIDENTIAL* PROBLEM REPORT # _____

PROGRAM _____ RELEASE _____ VERSION ____

REPORT TYPE (*1-6*) __
*1 - Coding error 4 - Documentation*
*2 - Design issue 5 - Hardware*
*3 - Suggestion 6 - Query*

SEVERITY (*1-3*) __
*1 - Fatal*
*2 - Serious*
*3 - Minor*

ATTACHMENTS (*Y/N*) ___
*If yes, describe:*
_____
_____

PROBLEM SUMMARY _____

CAN YOU REPRODUCE THE PROBLEM? (*Y/N*) __

PROBLEM AND HOW TO REPRODUCE IT_____
_____
_____
_____

SUGGESTED FIX (optional) _____
_____
_____
_____

REPORTED BY _____ DATE __/__/__

*ITEMS BELOW ARE FOR USE ONLY BY THE DEVELOPMENT TEAM*

FUNCTIONAL AREA _____ ASSIGNED TO_____

COMMENTS _____
_____
_____
_____

STATUS (*1-2*) __
*1 - Open 2 - Closed*

PRIORITY (*1-5*) ____

RESOLUTION (*1-9*) ___
*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*

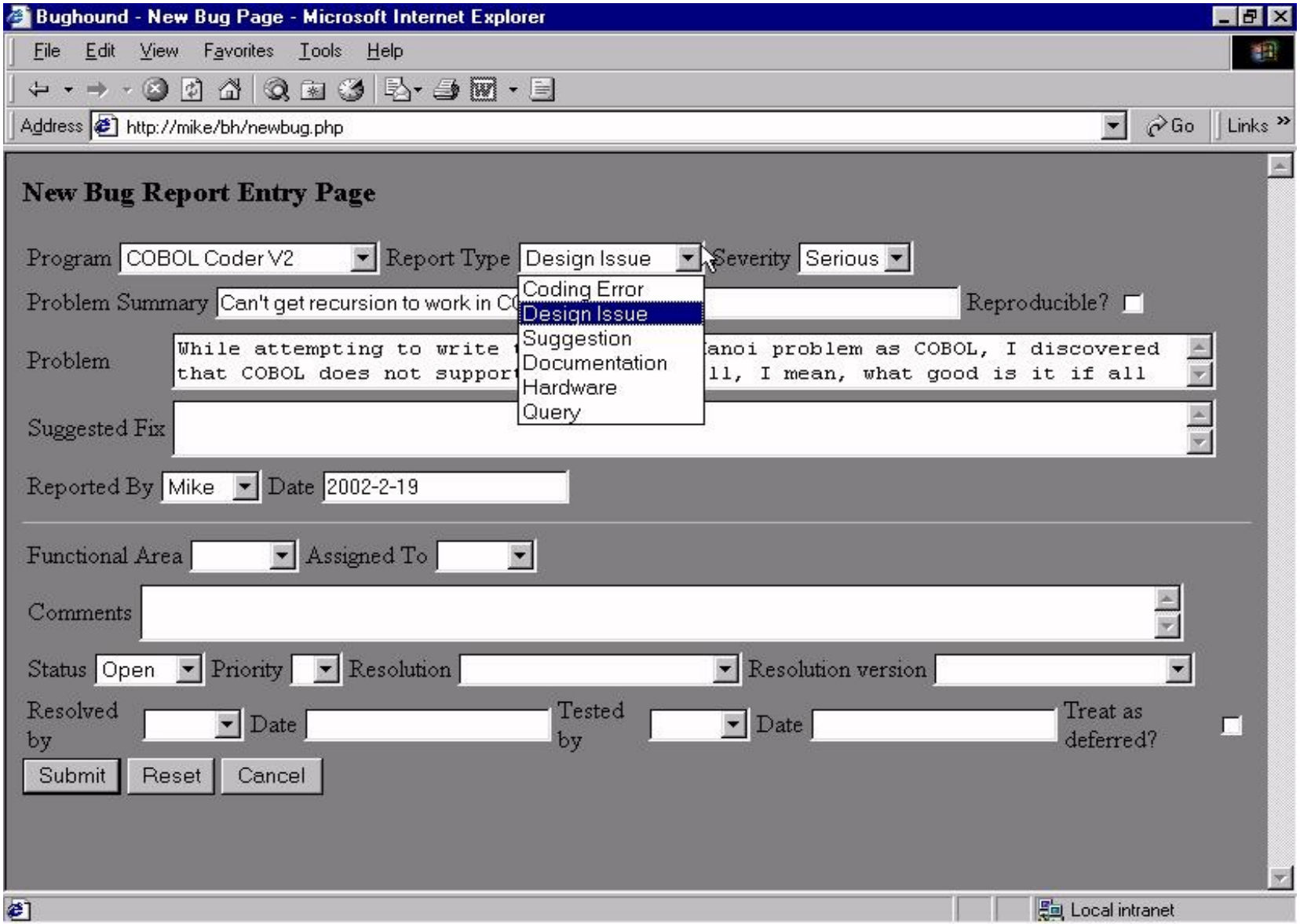
RESOLUTION VERSION _____

RESOLVED BY _____ DATE __/__/__

RESOLUTION TESTED BY _____ DATE __/__/__

- Bughound content is based on this form from Testing Computer Software, 2nd Edition, Kaner, Falk and Nguyen (see Syllabus)
- As with most forms, it translates easily to a web page or pages.
- Some fields, for example REPORT TYPE can be simple listboxes.
- Some fields are plain test (PROBLEM SUMMARY)
- Some require links to external files (ATTACHMENTS)
- For initial submission of a bug report, only fields up to REPORTED BY and DATE are used.

# The Report in Software



# Bughound Details

- Searchable fields
  - Program
  - Report Type
  - Severity
  - Functional Area
  - Assigned To
  - Status
  - Priority
  - Resolution
  - Reported By
  - Report Date
  - Resolved By
- Autogenerate bug ID
- Attach to report
  - Images
  - Memory dumps
  - Text files
- Dynamic fields
  - Release and version data correspond to program change
- One bug per report
- Validate input
- Export bug data to ASCII or XML

# Implementation

- Must be web browser based
- Must use a sql database for all bug data, program data, employee data, area data
- You must HAND CODE the project
  - You may use netbeans, eclipse, or Visual Studio
- It will require server and client scripting such as php or asp and javascript or something from the 21st century.

# Due Dates

- March 1 –
  - ER diagram for all DB elements
  - UML deployment diagram
- March 15
  - use cases for
    - create a new bug report
    - updating an existing bug report
    - management functions for the DB including
    - adding, updating, or deleting programs, areas, employees,
- April 17 – finished product ready for testing

# Contents of the Problem Report

- **Problem Report Number** - unique
- **Program** - your company may write many
- **Version** - *Release and Version*
  - necessary because the problem may not be reproducible in other versions
  - prevents confusion as to whether the bug was from a fixed version or the fix failed

# Contents of the Problem Report

- **Report Type**
  - *Coding Error* - program may work as designed
  - *Design Issue* - program works as intended, but design is in question
    - So? AS long as it works, what's the problem?
  - *Suggestion* - just an idea to improve the program
  - *Documentation* - doesn't behave as described in the manual or feature is not documented
  - *Hardware* - when the program fails on some specific type of hardware
  - *Query* - program behavior is unexpected

# Contents of the Problem Report

- **Severity** - indicates the opinion of the reporter as to the seriousness of the problem
  - Beizer rates from 1 to 10 (*Mild* to *Infectious*)
  - Fewer levels the better (*Minor, Serious, Fatal*)
  - Minor errors tend not to be fixed. But many minor errors affect software quality as seen by users.
  - Annoyances (too many steps, menus too deep) may be minor to the programmer, but major to users

# Contents of the Problem Report

- **Attachments** - printouts, memory dumps, memos, macros
- **Problem Summary** - one or two line summary. “Program crashes when commission less than 1%.”
- **Can You Reproduce the Problem? -** Yes, No, Sometimes. If the answer is No, many programmers ignore the report. If Yes, you may have to demo for the coders

# Contents of the Problem Report

- **Problem and How to Reproduce It**
  - explain why it is a problem. Describe all steps and symptoms including error messages
  - Be careful to describe how to reproduce the problem. Programmers ignore bugs they can't reproduce.
  - Better to fail reproducing now, than after you've reported the bug
  - Even if you can't reproduce it, describe all the steps taken to do so, and report the problem anyway

# Contents of the Problem Report

- **Suggested Fix** - optional. Programmers are not always able to imagine the fix (especially GUI issues)
- **Reported By** - Programmer must be able to contact the reporter if can't understand the issue. Also used for tacking tester productivity.
- **Date** - date the problem was discovered, not the date of the report. Helps to identify the version

# Contents Problem Report

- **The following are required in Bughound but not required to submit a report. That is, they may be left blank upon report submission.***
  - **Functional Area -** Everyone uses the same list
  - **Assigned To -** group or manager responsible for fixing the problem
  - **Comments** - in paper systems, a brief statement on how the problem was fixed. In software trackers, can be a running commentary, discussion, feedback from testers, coders, managers

# *More* Contents Problem Report

- **As previous, required but optional***
- **Status** - Open, Closed or Open, Closed, Resolved
- **Priority** - assigned by manager
  1. Fix immediately
  2. Fix as soon as possible
  3. Fix before next milestone
  4. Fix before release
  5. Fix if possible
  6. Optional

# *More* Contents Problem Report

- **As before, required but optional***
- **Resolution and Resolution Version** - current status
  - Pending
  - Fixed
  - Cannot be reproduced
  - Deferred
  - As designed
  - Withdrawn by reporter
  - Need more info
  - Disagree with suggestion
  - Duplicate