Module 8: Group Assignment

# JMeter load testing on To Do Web UI:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Code smell table:

|  |  |  |
| --- | --- | --- |
| Name | Category | Severity (1-10) |
| Data Class | Class | 1 |
| Dead Code | Logic | 3 |
| Temporary Fields | Class | 4 |
| Switch Statements | Class | 5 |
| Feature Envy | Class | 6 |
| Long Method | Class | 7 |
| Divergent Change | Architectural | 7 |
| Shotgun Surgery | Application | 8 |
| Large Class | Class | 8 |
| Duplicate Code | Application | 8 |

|  |  |  |
| --- | --- | --- |
| Name | Category | Severity (1-10) |
| Wolf Tickets | Project Management | 4 |
| Dead End | Application | 5 |
| Corncob | Code | 6 |
| Jumble | Code | 6 |
| Stovepipe System | Architectural | 7 |
| Golden Hammer | Developer | 7 |
| Analysis Paralysis | Project Management | 7 |
| Cut and Paste | Application | 8 |
| Death by Planning | Project Management | 8 |
| Lava Flow | Project Management | 8 |
| Boat Anchor | Architectural | 9 |

# How are code maintainability and simplicity related?

Code maintenance and simplicity are related since having the latter will make the former easier to do. Simpler code is easier for developers to understand because developers will have an easier time deciphering what the code does. When fixing any issues that arise or adding new features, developers will have a better understanding of how the program works and know that they can maintain the software without the fear of breaking everything.

Simple code is the opposite of spaghetti code by being organized in a modular way. By breaking down the code into smaller modules, it becomes easier to maintain since the maintainer can take their time to understand how the code works bit by bit instead of having to rely on other developers with experience. This also decreases the maintainer’s reliance on documentation which may be poor if the code is complicated. Any changes made to maintain the code won’t require extensive new documentation since the simplicity of the code will make it speak for itself.

# Runtime efficiency and code simplicity are often competing goals. How can you deal with this problem? Is it possible to have code that is both simple and efficient?