# Defect Tracking Policy

## About

This document is a comprehensive guide for the tracking and resolution of software defects for the ECE 1140 class ‘Train’ Project. This guide allows us to detect issues early, deal with them efficiently, and help with ease of integration

## Procedure

1. **Identification** 
   1. **How to find defects?**

It's important within the context of software design to find issues early and often. This requires frequent testing of issues, even when you believe your module is functioning correctly. An important piece of this is *edge case testing.* In order to identify issues, we want to create a bank of tests that test for cases of unusual/out of normal behavior from the system.

Secondarily, issues are often blind to the creator. Cross collaborations of modules is a key component of throughout testing

* 1. **What to do when an defect is identified?**

When a person/s identifies a defect within the system. They must note to which module it originates, whom it might also affect, and who wrote the code causing the defect. The following steps will identify in more detail how to deal with defects.

1. **Testing**

After the identification of a defect. Before committing an formal report to Jira, first, we must test to understand how this defect will affect our system. Formally, testing means to explore how this defect is affecting the system.

* 1. **Extent testing**

The first kind of testing is ‘Extent testing’. This is to measure how many sub-modules will be affected by this defect. In order to properly test this, you must work with other modules to test the flow of data. Its important to identify whether or not the defect only exists locally, and thus can be resolved by a sub-module creator, or extends beyond and requires further research and collaboration to solve.

*Extent Types:*

1. Local Extent - The defect affects only the module in which it exists
2. Dual Extent - The defect affects the module in whihc it exists and one other module
3. Left/Right Extent - The defect affects either the ‘Left System’ (CTC, Wayside, Track Model) or ‘Right System’ (Track Controller, Track Model)
4. Full Extent - The defect affects all or most modules
   1. **Severity Testing**

After seeing the extent to which the program is affected by the defect. It is up to the tester to identify the severity of the defect. It can fall into multiple categories based on how it affects the different functionalities of the system

*Severity Types:*

1. Minor/No Severity - The defect has little to no affect on the functionality of the system
2. Non-Vital Severity - The defect affects only non-vital tasks in the system
3. Vital Severity- The defect affects vital tasks in the system
4. Full Severity - The defects breaks the entire functionality of the program
5. **Jira Procedure**

Once the testing is complete, it is time to notate and assign work related to the defect.

* 1. **Creating Jira Task**
     1. The task shall be created on Jira and assigned based on Severity.
        1. Minor/No Severity → Backlog Category
        2. Non-Vital Severity → Final Work Package
        3. Vital Severity → Most upcoming Work Package
        4. Full Severity → Most important task // It's own category
  2. **Allocation of Task**

1. The allocation of tasks shall be based on the extent of which the defect affects.
2. Local Extent → Assigned to only module affected
3. Dual Extent → Assigned to module with defect creation but may require counsel from other module
4. Left/Right Extent → Assigned to module with defect creation but must require counsel from additional modules
5. Full Extent → All hands on deck to fix issue