1. Test selection problem:

My test selection problem is this following function

public void FiredEmployeeID(int ID) {

        for (int i = 0; i < EmployeeList.size(); i++) {

            if (ID == EmployeeList.get(i).getID()) {

                continue;

            }

        }

    }

By adding this **Junit** test case

@Test

    public void testFiredEmployeeID\_Found() {

        factory.addEmployee("John", 123, 1000.0);

        assertTrue(factory.CheckEmployeeID(123));

        factory.FiredEmployeeID(123);

        assertFalse(factory.CheckEmployeeID(123));

    }

By asserting false to see if the deleted employee still exists, test fails due to an (Trivial mutant) (syntax based testing pdf page 8)

Due to an error in the if condition in FiredEmployeeID function in the body of the statement the continue skips the main purpose of the function (removing the employee from the list)   
Here is the solution

public void FiredEmployeeID(int ID) {

        for (int i = 0; i < EmployeeList.size(); i++) {

            if (ID == EmployeeList.get(i).getID()) {

                EmployeeList.remove(i);

                break;

            }

        }

    }

I just added [remove()](https://www.scaler.com/topics/remove-in-java/) function on the element that need to be moved by id

note: remove is built in java function

1. I those the addEmployee function as it appears clear as crystal

public void addEmployee(String Name, int ID, double baseSalary) {

        Employee emp = new Employee(Name, ID, baseSalary);

        EmployeeList.add(emp);

        numberOfEmployee++;

    }

WP: EmployeeList != null && !EmployeeList.contains(new Employee(Name, ID, baseSalary))  
  
It will satisfy the assumed post condition :   
  
**EmployeeList**.contains(new **Employee**(**Name**, **ID**, **baseSalary**))   
  
This means that the **EmployeeList** is not null and does not already contain a new **Employee** object with the specified **Name**, **ID**, and **baseSalary**. This ensures that a new **Employee** object is added to the **EmployeeList** only if the list is not null and the new **Employee** object is not already present in the list

1. My suggestion is to assign a PM(Architect) with a technical background(DDD) to the product to clarify the retirements By adding ACs to the user stories, it would help devs to know the edge cases in the product and they can also work in TDD by adding the tests before implementing any requirements, Maybe they can also add a SDET to ensure that the project is working as needed from E2E perspective
2. It seems like an employee’s payroll app (Karim’s OOP course 12th project)

Firstly, I would prefer to do a heavy **functional testing** and static analysis

This will be divided into two sections

|  |  |
| --- | --- |
| internally (inside the company) | Externally |
| Ensure that every employee is on the expected payroll (for example 15kEGP as a base salary for senior devs) | It is very… very critical to check **Egyptian Tax Authority** updates frequently |
| (For reputable Companies ☺) check COLA is added under certain conditions (inflation) | [Taxes APIs](https://sdk.invoicing.eta.gov.eg/release-notes/Feb-2023-release/) For ERP systems |
| It includes testing the various features of the application such as calculating salaries, deducting taxes, and generating pay stubs |  |

After finishing the functional testing we can run a **load testing** cycles by using JMeter or Gatling to see how the system will act under certain workloads (running Threads)