Ticketing Application for Defect Management

Sprint Implementation-1

1.Introduction

The introduction of the software requirement specification provides an overview of the entire Software. The entire SRS with overview description purpose, scope, tools used and basic description. The aim of this document is to gather, analyze and give an in-depth insight into the Defect Programmer Assignments by defining the problem statement in detail. The detailed requirements of the Defect Programmer Assignments are provided in this document.

1.1. Project Purpose

Clients of software development company report defects in software they purchased. These defects are assigned to programmers depending on the functional area they are handling.

1.2. Intended Audience

This document is intended to be read by the client.

1.3. In Scope

It can take multiple files with defects as input and check the input for validity and assigns the employee to the valid defects.
Invalid defects are copied to a separate file and unassigned defects are copied to a separate file.

1.4. Out of scope

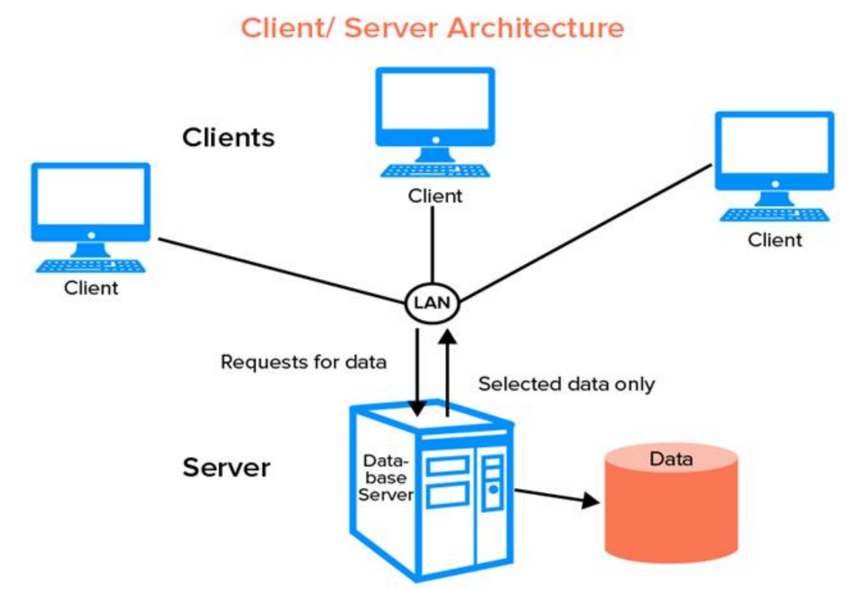
It doesn’t reassign the employees after they have been freed as their work.
It doesn't correct the invalid defects in the input file.
It doesn’t check for invalid employees
It doesn’t check for duplicates of input
It doesn’t check for incorrect categories for each parameter in the input file.

1.5. Assumptions, Dependencies & Constraints

Operating systems: It is made for windows 7 or above

End-user characteristics: It is assumed that they are not invalid employees, they are not duplicates of input, there are no incorrect categories for each parameter in the input file.

System Architecture:



1.1. Some of the characteristics of Database Architecture are:

2.Self-Describing Nature of a Database System :

One of the most fundamental characteristics of the database approach is that the database system contains not only the database itself but also an entire definition or description of the database structure and constraints also known as metadata of the database**.**

Isolation between Data, Programs and Data Abstraction:

- In a traditional file processing system, the structure of database knowledge files is embedded within the application programs, so any changes to the structure of a file may require changing all programs that access that file.

Support for Multiple Views of the Data :

- A database sometimes has many users, each of whom may require a special perspective or view of the database.

Sharing of knowledge and Multi-user Transaction Processing :

- A multi-user DBMS, as its name implies, must allow multiple users to access the database at an equivalent time or concurrently

1.2. System Use-Cases

A **system use case** displays the relationships between **consumers and providers of application services**. Application services are consumed by other application services and the application use case diagram provides added richness in describing application functionality by illustrating how and when that functionality is used. As the architecture progresses, the **use case** can evolve from functional information to include technical realization details. Architectural system use cases can also be re-used in more detailed system design work.

1.1. Subsystem Architecture

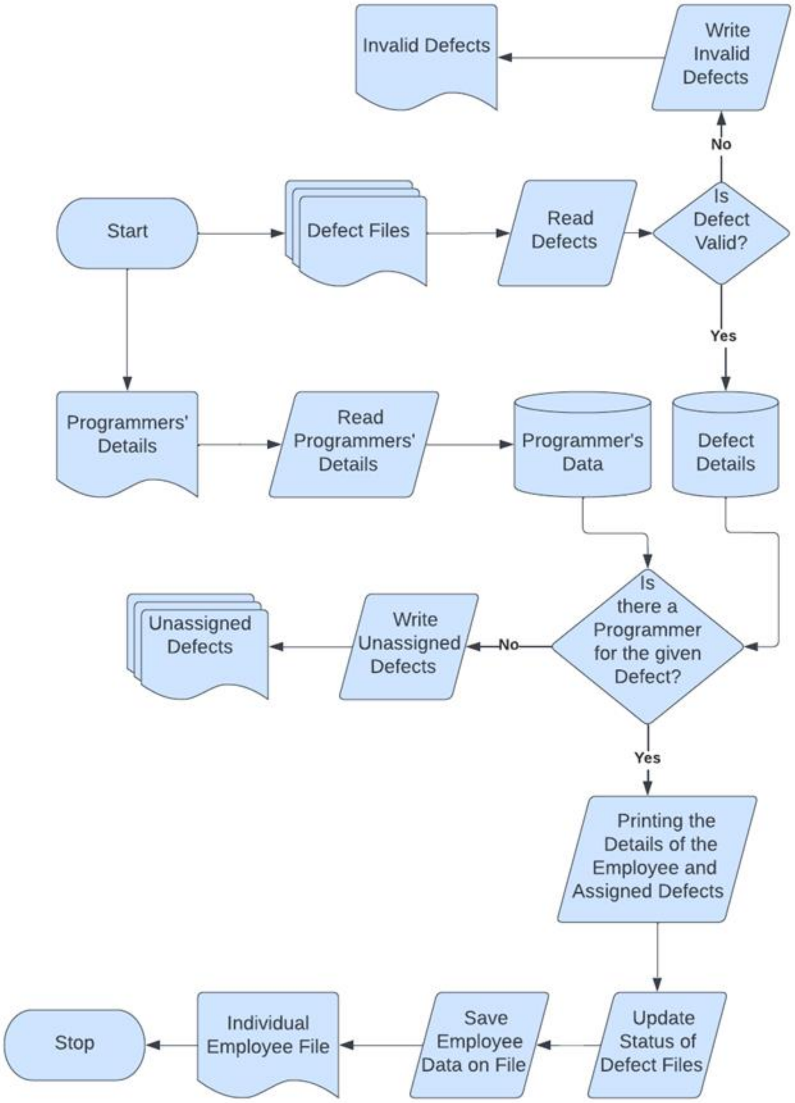
The Model Subsystem has the responsibility of handling the data in the system. The Subsystems can contain other Subsystems or they can be split into separate Subsystems. breaking the overall functionality and implementation down into well segmented subsystems with cleanly specified interfaces is one of the principle challenges of a software architect. Success in this endeavor is key in ensuring that the source code can be efficiently developed, enhanced, tested and maintained. But such success is not always achieved. And even when the initial system architecture is well structured, the software structure can degrade over the life of the system.

1.2. System Interfaces

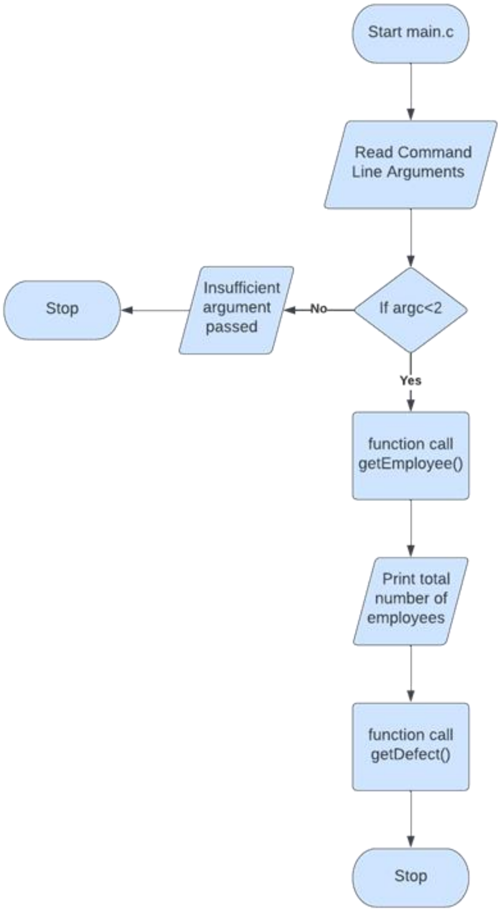
The logical characteristics of each interface between the software product and the hardware components of the system ' 'the logical characteristics of each interface between the software product and its users '

2. Detailed System Design

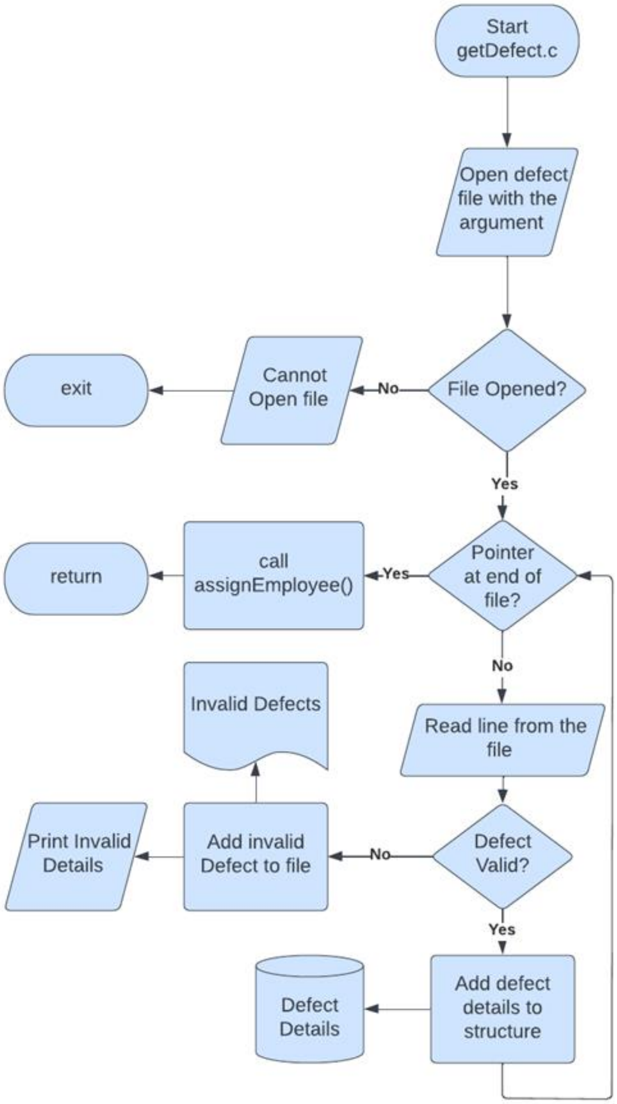
Overall Flowchart



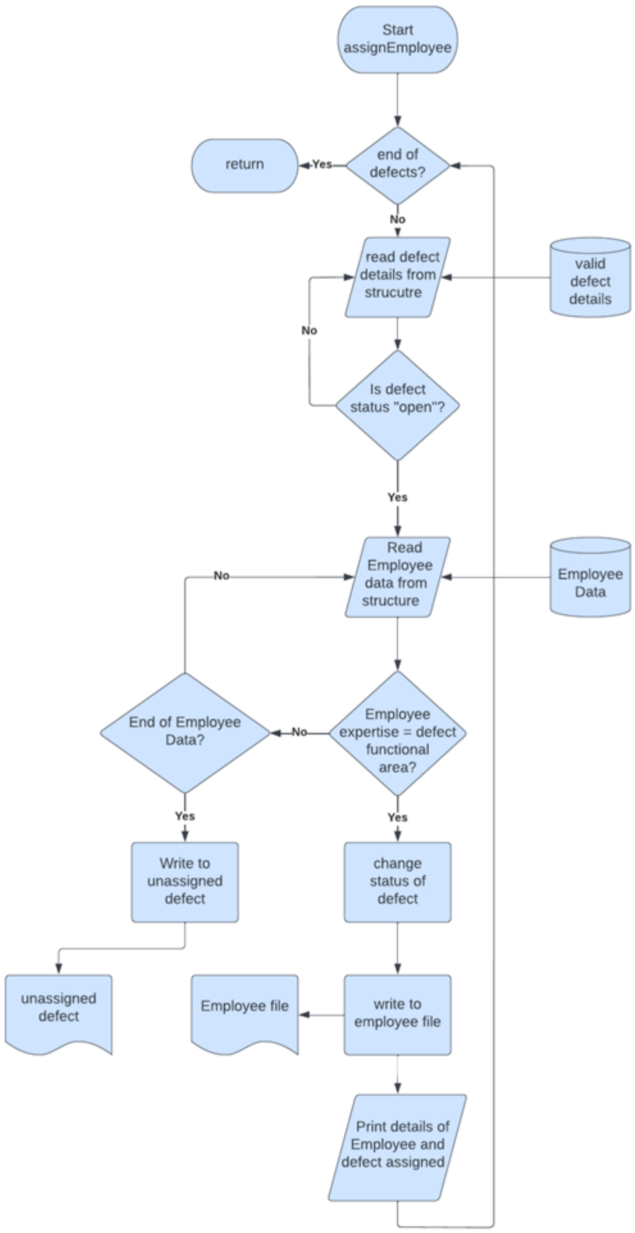
Flowchart of main.c


Start main.c
Read Command Line Arguments
If argc<2
No
Insufficient argument passed
Stop
Yes
function call getEmployee()
Print total number of employees
function call getDefect()
Stop

Flowchart of getdefect.c



Flowchart of assign.c



4.Structures details:

4.1 Defect structure:

It stores details of a particular defect.

All the properties of this structure are self – exploratory except status

Status refers whether it is assigned or not, If the status is open or closed.

```
Description

defectId

module Name
functional Area
date
status
Type
```

4.2. Employee Structure:

It stores details of a particular employee.

BUnit refers to the bussiness unit of the employee.

Emplock is a parameter which is used for mutex locking of certain code

n_defect refers to no. Of defects assigned to this employee.

assigned_arrr stores details of all the assigned errors.

Remaining parameters are self-exploratory

```
Id
Name
BUnit
Expertise
Designation
n_defect
assigned_arr[MAXDEFECT]
```

5.Environment Description

The system environment is primarily the set of variables that define or control certain aspects of process execution.
They are set or reset each time a shell is started. From the system-management point of view, it is important to ensure the user is set up with the correct values at log in. Most of these variables are set during system initialization. Windows uses environment variables to store valuable information about system processes, resource usage, file path, and more.

5.1. Time Zone Support

The time functions access and reformat the current system date and time. You do not need to specify any special flag to the compiler to use the time functions. Include the header file for these functions in the program.

5.2. Language Support

English is only supported language.

5.3. User Desktop Requirements

- **Operating systems**: Windows 10/11, Windows 8.1, Windows Server 2019, Windows Server 2016, Windows Server 2012 R2
- **CPU**: 1vCPU with 1 GHz or faster processor
- **RAM**: 1024 MB
- **Hard drive**: 100 MB or more
- **.NET Framework version**: 4.6.1 or later
- **Video**: DirectX 9 or later with WDDM 1.0 driver

5.4.Server-Side Requirements

server-side code has full access to the server operating system and the developer can choose what programming language (and specific version) they wish to use. Developers typically write their code using web frameworks.

5.5.1.Deployment Considerations

There are new deployment options in Windows 10 that help you simplify the deployment process and automate migration of existing settings and applications.
For many years, organizations have deployed new versions of Windows using a “wipe and load” deployment process. At a high level, this process captures existing data and settings from the existing device, deploys a new custom-built Windows image to a PC, injects hardware drivers, reinstalls applications, and finally restores the data and settings. With Windows 10, this process is still fully supported, and for some deployment scenarios it is still necessary.

5.5.2.Application Server Disk Space

Windows Server does not allow ATA/PATA/IDE/EIDE for boot, page, or data drives. The following are the estimated minimum disk space requirements for the system partition. Minimum: 32 GB

5.5.3.Database Server Disk Space

Database server (Windows)	Windows Server 2012 R2, 2016, 2019 MSSQL 2016 and up, or Oracle 11g and up, or PostgreSQL 11 and up
Database server (Linux)	Linux Debian 8-9 PostgreSQL 11 and up

5.5.4.Integration Requirements

system integration involves both the integration of components and the integration of functions. High-level functions can be obtained by integrating components into a comprehensive working system instead of individual disconnected subsystems

5.5.5.Network

As a cloud SaaS application, Oracle Fusion Cloud Warehouse Management, formerly Log Fire, is accessed over the internet. Adequate bandwidth is required to provide a responsive experience. Bandwidth requirements vary based on the volume at a client site. Bandwidth is not the only determining factor for application responsiveness.

5.5.Configuration

Only configurations that directly apply for use with Configuration Manager are included here.

5.6.1.Operating System

Operating system management is the task of an individual who is usually referred to, in UNIX literature, as the system administrator. Unfortunately, only a few system administrator activities are straightforward enough to be correctly called administration. This and related guides are intended to help system administrators with their numerous duties.
This operating system provides its own version of system-management support in order to promote ease of use and to improve security and integrity.
In most cases, running 32-bit applications on 64-bit hardware is not a problem, because 64-bit hardware can run both 64-bit and 32-bit software. However, 32-bit hardware cannot run 64-bit software.

5.6.2.Database

Database configuration is to set up the connection details required to connect to your database from Visual Paradigm. Database configuration is required because: Select Tools > DB > Database Configuration… from the toolbar. At the top left corner of the Database Configuration window, set the (Programming) Language of your project.

5.6.3.Network

Network configuration is the process of assigning network settings, policies, flows, and controls. In a virtual network, it's easier to make network configuration changes because physical network devices appliances are replaced by software, removing the need for extensive manual configuration.

- Automated data tracking and reporting, allowing administrators to spot any configuration changes and potential threats or issues
- An easy way to make bulk changes, such as a blanket password change in a situation where passwords are compromised

5.6.4.Desktop

Configuration represents the way in which a system is set up, related to the assortment of components that make up the system, it can refer to hardware, software, or a combination of both.