High Level Design

Project Management System

Last date of revision: 28/7/22

Contents

1. Introduction

1.1. Why this High Level Design Document?

The purpose of this High Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

1.2. Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3. Definitions

- PMS-Project Management System
- Fair Share – An administratively set data rate per time frame that is considered fair.
- Throttling – A reduction in maximum transfer rate of data.
- Open Pipe – A connection to the Internet with no throttling.
- Slow Pipe – A throttled version of an open pipe where all users of that gateway share that reduced pipe's rate.
- Firewall –Functionality that can allow or block certain ports and addresses.
- IPTables – A firewall built into the Linux kernel.

1.4. Overview

The HLD will:

- present all of the design aspects and define them in detail
- describe the user interface being implemented
- describe the hardware and software interfaces
- describe the performance requirements
- include design features and the architecture of the project
- list and describe the non-functional attributes like:
    - o security
    - o reliability
    - o maintainability
    - o portability
    - o reusability
    - o application compatibility
    - o resource utilization
    - o serviceability

2. General Description

2.1. Product Perspective

The Employee Project Management system Goals of the project are: The employee-project management system has been designed and developed to manage the process project details like project id, project name, start date, end date, no of resources required, no of resources allocated, project status and employee details like emp id, emp name, emp status, gender, basic salary, bonus, project details. In the employee object, the number of projects being allocated are a maximum of three projects with allocation information like project id, allocation start date, allocation end date, role, percentage allocated.

In order to add / delete the employee, the administrator has a built-in login id and password set. There may be any number of administrators. Here, in the employee details entry, the employee id's auto generated. All employee details should be captured. The administrator is allowed to add projects. The project id is auto generated. All project details should be captured. The end date should be at least one month after the start date.

The administrator is allowed to allocate the projects to the resource (employee). Once a resource is allocated to a project, update the count. The project status is by default active. It will be set as "closed" manually. Resources cannot be allocated to closed projects.

An employee can be allocated to a max of 3 projects. The sum of allocation percentage of all the projects should be 100. As soon as an employee is allocated to 3 projects or allocation percentage sum is 100, the employee is marked as allocated and bonus also calculated.

The employee is given a user id and password. The employee can login and see his/her details including employment and project.

2.2. Tools

1. Visual studio.

2. Git

3. linux environment for running programs
   gcc *.c –o main

4. Thread concepts
   gcc *.c –o main –lpthread

5. gdb-debugging the line by line code
   gcc *.c –o main –lpthread
   gdb main

   or

   make build

   gdb outputfilename
   ex:gdb main

6. valgrind—memory check
   valgrind -s --leak-check=full ./main

7. splint-checking uninitialized variables
   make lint

8. to check memory leak in gdb
   make memcheck

9. gprof-profile creation

```
gcc -g -pg *.c -o $(EXE)
gprof ./$(EXE)
gprof ./$(EXE) gmon.out
gprof ./$(EXE) gmon.out --brief
gprof ./$(EXE) gmon.out --brief --flat-profile
gprof ./$(EXE) gmon.out --brief --graph
```

2.3. General Constraints

Employee Project Management System mainly works on resource to be allocated on the project or not.

If The resource is not allocated then status should be 0(not active).

If The Resource is allocated the status should be 1(active).

Admin allocates the projects to the resources.

2.4. Assumptions:

In order to add / delete the employee, the administrator has a built-in login id and password set. There may be any number of administrators. Here, in the employee details entry, the employee id's auto generated. All employee details should be captured. The administrator is allowed to add projects. The project id is auto generated. All project details should be captured. The end date should be at least one month after the start date.

3. Design Details

3.1. Main Design Features

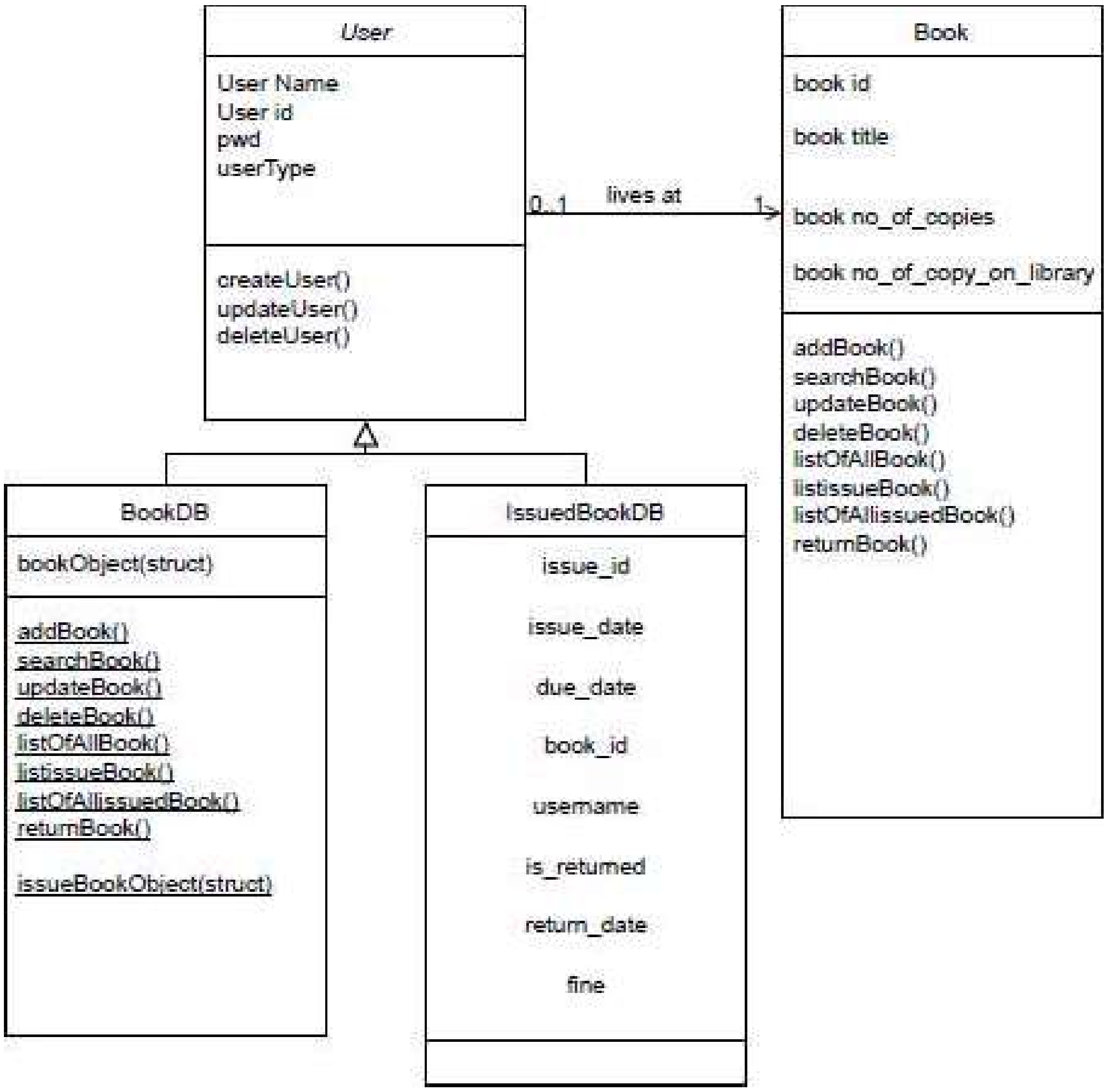
The main design features include five major parts: the architecture, the user interface design, external interface, the database, process relation, and automation. In order to make these designs easier to understand, the design has been illustrated in attached diagrams (Use Case, and Screen Shots).

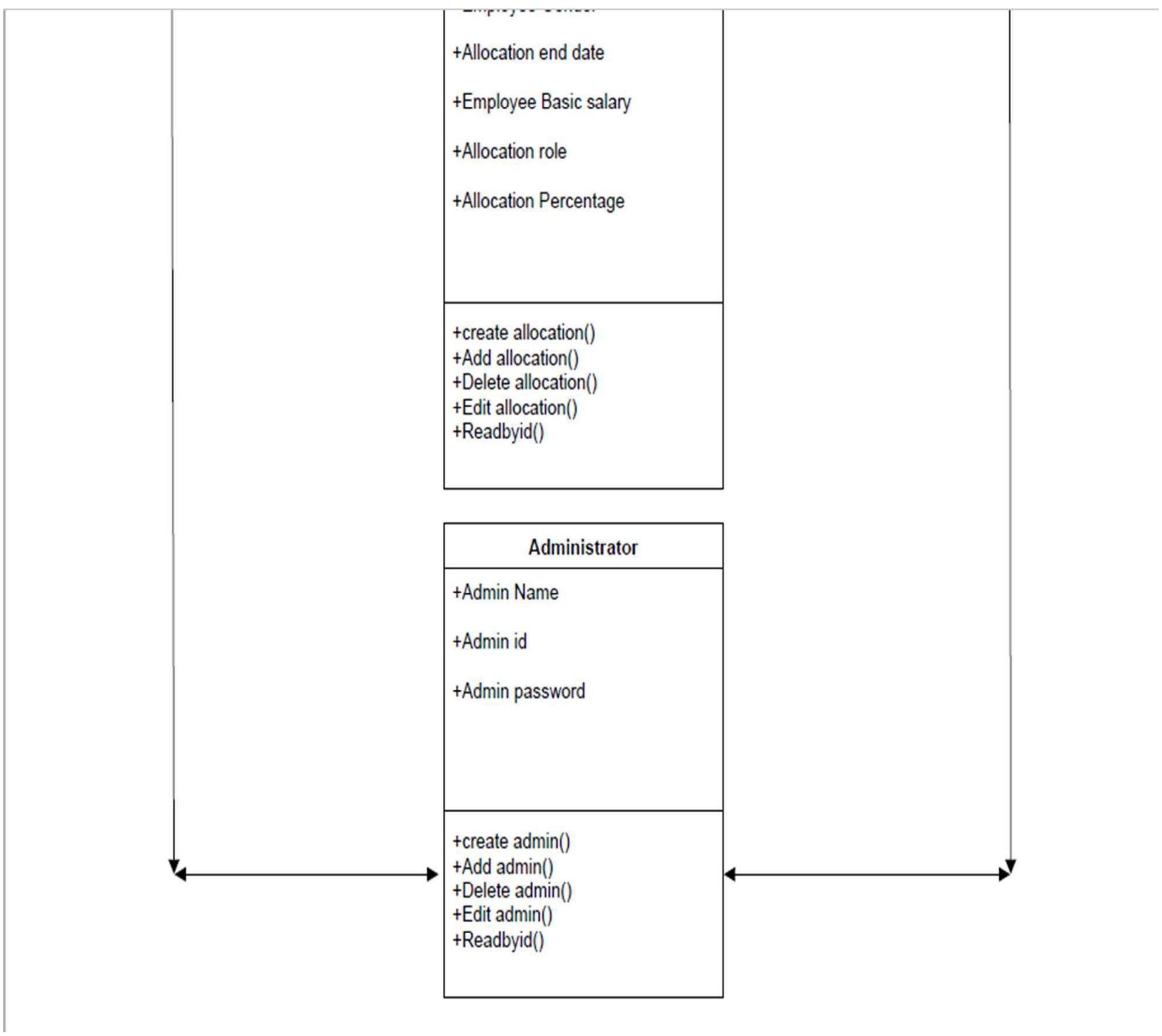
3.2. Application Diagrams:

--class diagram
--sequence diagram
--use case diagram respectively

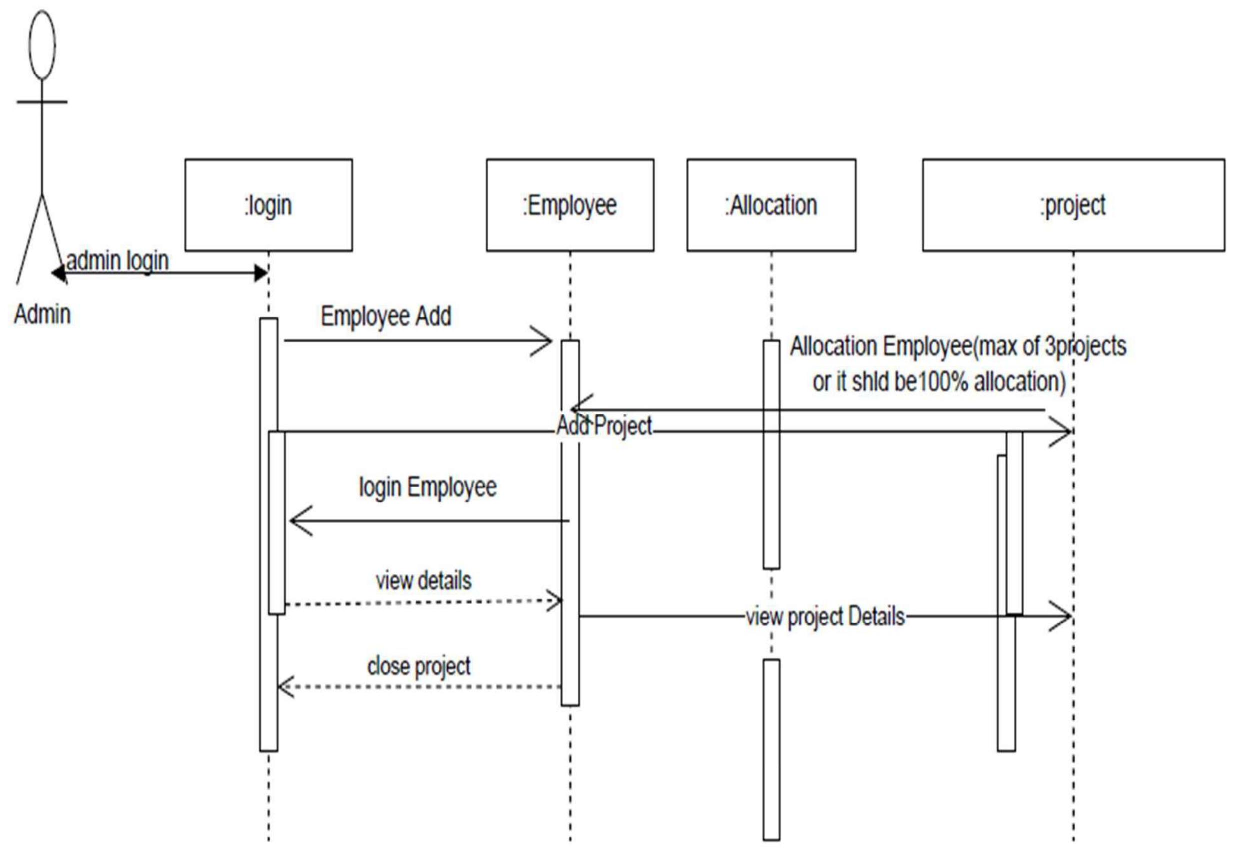
Class Diagram:

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

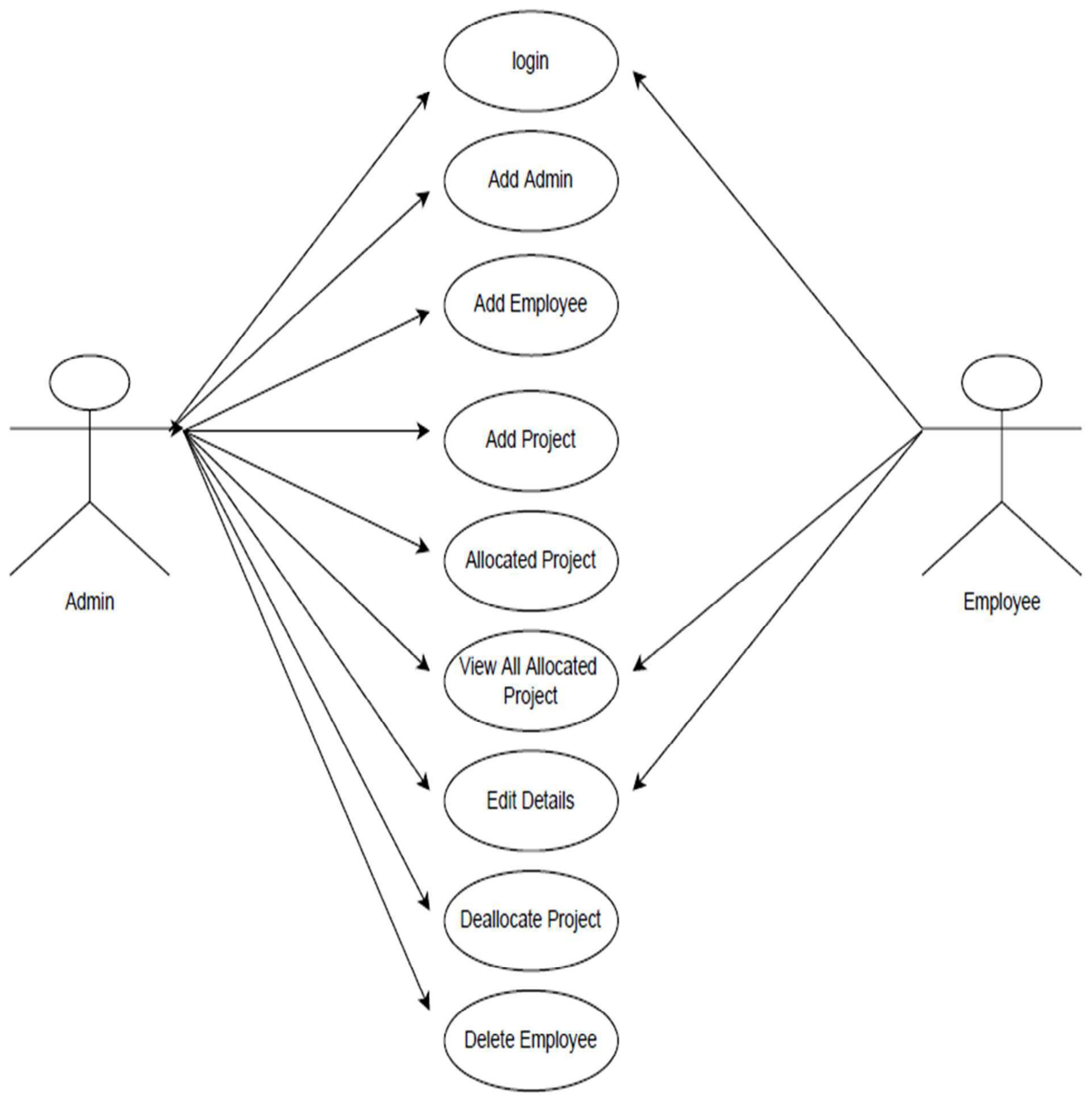



+Allocation end date
+Employee Basic salary
+Allocation role
+Allocation Percentage
+create allocation()
+Add allocation()
+Delete allocation()
+Edit allocation()
+Readbyid()
Administrator
+Admin Name
+Admin id
+Admin password
+create admin()
+Add admin()
+Delete admin()
+Edit admin()
+Readbyid()

Sequence Diagram:
A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.



UseCase Diagram:
Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.



4.Technology Architecture

4.1 Files:

In Employee Project management System Files we used to store the data and read file from database and edit , delete, update all operation is based on the file.

4.2 User Interface:

In this Employee Project Management System Admin is the User Interface.
Admin can add the employee and add the project and no of resources to the projects.
Resources allocated to the projects done by admin only.

4.3 Reports:

The report displays the The no of employess allocated to the projects and no of employees Not allocated the projects and Total Resources required for the projects.

4.4 Error Handling:

In Employee Project management system we have resolved all the issues comes when we are executing the files or compilation of the process.

4.5 Database Design:

In Employee Project Management system we are creating admin.dat for admin files and project details for project.dat and employee details for emp.dat for store and update the db. we can get the information from database.

4.6 Interfaces:

Admin:

- In order to add / delete the employee, the administrator has a built-in login id and password set
- The administrator is allowed to add projects
- The administrator is allowed to allocate the projects to the resource (employee). Once a resource is allocated to a project.

Employee:

- The employee is given a user id and password.
- The employee can login and see his/her details including employment and project.

4.7 Performance:

Employee Performance Management is about aligning the organisational objectives with the employees' agreed measures, skills, competency requirements, development plans and the delivery of results.

4.8 Security:

Security, in the context of project management, can therefore be defined as the identification of potential risks and implementation of strategies which will protect or preserve the confidentiality, integrity, and availability of project resources.

4.9 Reliability:

Reliability is an aspect of performance that refers to how consistently the project management software does what it's supposed to. For example, if your project management software takes an hour to run a basic report, there's a performance issue

5.0 Portability:

This system should have the ability that, once it is together, the entire system should be able to be physically moved to any location. Code and program portability should be possible between kernel-recompiled Linux distributions. For everything to work properly, all components should be compiled from source.

5.1 Reusability:

The code written and the components used should have the ability to be reused with no problems. Should time allow, and detailed instructions are written on how to create this project, everything will be completely reusable to anyone.

5.2 Application Compatibility:

In this Library Management System we are using the C language.
we can compile code and debug using linux tools.

1. linux environment for running programs
   ```
   gcc *.c –o main
   ```
2. Thread concepts
   ```
   gcc *.c –o main –lpthread
   ```
3. gdb-debugging the line by line code
   ```
   gcc *.c –o main –lpthread
   gdb main
   ```
   or

   ```
   make build
   ```
   gdb outputfilename
   ex:gdb main
4. valgrind—memory check
   ```
   valgrind -s --leak-check=full ./main
   ```
5. splint-checking uninitialized variables
   ```
   make lint
   ```
6. to check memory leak in gdb
   ```
   make memcheck
   ```
7. gprof-profile creation
   ```
   gcc -g -pg *.c -o $(EXE)
   gprof ./$(EXE)
   gprof ./$(EXE) gmon.out
   gprof ./$(EXE) gmon.out --brief
   gprof ./$(EXE) gmon.out --brief --flat-profile
   prof ./$(EXE) gmon.out --brief --graph
   ```