

**Client-Server Database**

Low-Level Design Version Draft v7

**Document Control:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Project Revision History** | | | | | | |
|  |  |  | |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | |
|  |  |  |  | | | |
|  |  |  |  | | | |

|  |
| --- |
| **Team Members** |

|  |  |
| --- | --- |
| **Employee ID** | **Name** |
| **46290158** | **Muskan Pathan Allabaksh** |
| **46282451** | **Sri Harshita Talari** |
| **46282452** | **Renuka Gujjala** |
| **46282098** | **Bhagyalakshmi Daliparthi** |
| **46282102** | **Sarika Pathiputturu** |
| **46290159** | **Vinitha Devalla** |

|  |
| --- |
| **Table of Contents** |

**I) Low-Level Design**

|  |  |
| --- | --- |
| 1. Introduction |  |
| 1.1 Purpose | 4 |
| 1.2 Document Conventions | 4 |
| 1.3 Intended Audience and Reading Suggestions | 4 |
| 1.4 References | 4 |
| 1. Detailed system design | 5 |
| 2.1 Design Description | 5 |
| 2.2 Flowchart | 6 |
| 2.3 Modules | 7 |
| 2.4 Use Case Diagram | 15 |

|  |
| --- |
| **Low-Level Design** |

**1. Introduction**

The aim of this document is to gather, analyze and give an in-depth overview of the Project i.e., Data-Base Server. It is an application to store and retrieve different types of data (e.g., Employee data, School data, etc..). A list of supported data types and format is published by the server. All clients connecting to server will send data to store in the server and they can retrieve the data from Server. The Database Server is based on the concept of communication between client and server and managing the data in database. The server will maintain the different type and formats of data in database and will receive the relative data from client and will add to the database. The server can connect to single client at a time. No two clients can access or modify the same database at a time. The server should employ proper connection of database in case two clients attempt to modify the database values at same time.

### 1.1 Purpose

The purpose of this document is to describe the low-level design flow of the current project description to represent a suitable model for coding.

### 1.2 Document Conventions

TBD (To be continued).

### 1.3 Intended Audience and Reading Suggestions

This is general-purpose Software Thus any one Can Access it.

1.Client

2.Development Team

3.Maintenance Team

The document is primarily intended for team members, which consists of trainees under the **Capgemini** Training Program.

### 1.4 Refers

The references are:

1. System Requirements Specification Document
2. System Specification Requirement

## 2. Detailed System Design

**2.1 Design Descriptions:**

This project design mainly focuses on implementing of Client Server model. User will be able to choose the based on reequipment, which they want to implement. In these projects after the successful connection of Server and Client, the Server will display the Database format to user. Users have to select a choice of Database in which, he wants to add, delete, retrieve and display the data. Based on the choice the user has entered the algorithm will continue the process.

Here, the Client-Server and Database will help in understanding the storage of data in database and commination between them.

**Main menu:**

User gets options selection menu or to exit the program.

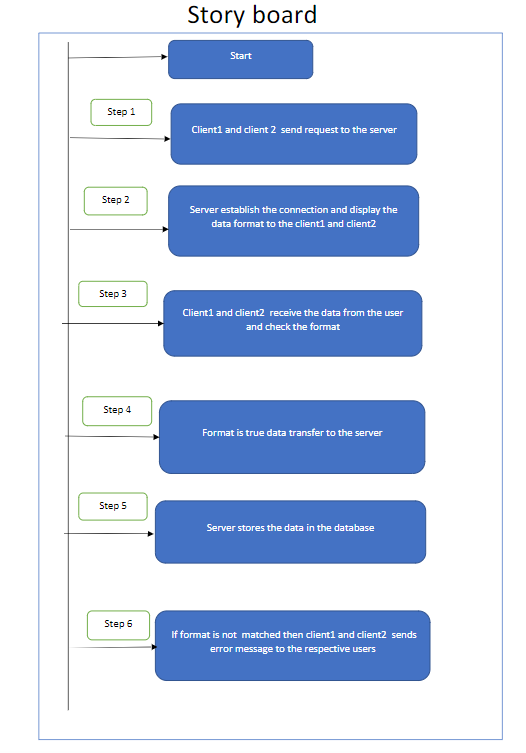
**2.2 Flowchart**

**2.2.2 Menu level 2**

**Diagram

Description automatically generated**

**2.2.3 Storyboard**

****

**2.3 Modules**

**2.3.1 ui.c**

**main()**

| **Name** | main | | | |
| --- | --- | --- | --- | --- |
| **Input** | Parameter Name | NA | Initial value:NA | - |
| **Output** | Return value type | int | - | - |
| **Description** | Main User interface file for interaction with user | | | |
| **Pseudo Code** | 1. Display the main menu  2. Wait for user options  3. Based on user input create a ready-to-run queue with process names | | | |

**2.3.2 ui\_val()**

| **Name** | ui\_val | | | |
| --- | --- | --- | --- | --- |
| **Input** | Parameter Name | int, int |  | To check whether user input is valid or not. |
| **Output** | Return value type | int |  | NA |
| **Description** | This program is used to set limits to the input values of the UI. | | | |
| **Pseudo Code** | 1. Validate the inputs based on the menu level.  2. If invalid return -1, else return 0 | | | |

**2.3.3 fcfs()**

| **Name** | fcfs | | | |
| --- | --- | --- | --- | --- |
| **Input** | Parameter Name | NA | Initial value:NA | - |
| **Output** | Return value type | int |  | - |
| **Description** | This is a fcfs algorithm which will take input as binary files of a process and output the simulation between input processes. | | | |
| **Pseud**  **Code** | 1. Asks the user to choose the number of binary files to simulate  2. Takes path of binary of processes in order  3. Creates user-entered queue, ready to run queue.  4. Executes the binary of processes according to fcfs algorithm | | | |

**2.3.4 ljf ()**

| **Name** | ljf | | | |
| --- | --- | --- | --- | --- |
| **Input** | Parameter Name | NA | Initial value:NA | - |
| **Output** | Return value type | int |  | - |
| **Description** | It will take input as binary files of process and burst time and provides the simulation for entered processes. | | | |
| **Pseud**  **Code** | 1. Asks users to choose the number of binary files to simulate.  2. Takes path of binary of processes in order and burst time for each process.  3. Creates user-entered queue, ready to run queue.  4. Executes the binary of processes according to the Ljf algorithm. | | | |

**2.3.5 prio()**

| **Name** | prio | | | |
| --- | --- | --- | --- | --- |
| **Input** | Parameter Name | NA | Initial value:NA | - |
| **Output** | Return value type | int |  | - |
| **Description** | This is a priority algorithm which will take input as binary files of process and priority of each process and output the simulation between input processes. | | | |
| **Pseud**  **Code** | 1. Asks the user to choose the number of binary files to simulate  2. Takes path of binary of processes in order and priority for each process.  3. Creates user-entered queue, ready to run queue.  4. Executes the binary of processes according to priority algorithm | | | |

**2.3.6 rr()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | rr | | | |
| **Input** | Parameter Name | NA | Initial value:NA | - |
| **output** | Return value type | int |  | - |
| **Description** | This is a round robin algorithm which will take input as binary files of process, quantum time and the total time of the processes and output the simulation between input processes. | | | |
| **Pseud**  **Code** | 1. Asks the user to choose the number of binary files to simulate  2. Takes path of binary of processes in order, burst time for each process and time quantum.  3. Creates user-entered queue, ready to run queue.  4. Executes the binary of processes according to RR algorithm | | | |

**2.3.7 sjf()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | sjf | | | |
| **Input** | Parameter Name | NA | Initial value:NA | - |
| **Output** | Return value type | int |  | - |
| **Description** | This is a SJF algorithm which will take input as binary files of process and burst time and provides the simulation for entered processes. | | | |
| **Pseud**  **Code** | 1. Asks the user to choose the number of binary files to simulate  2. Takes path of binary of processes in order and burst time for each process  3. Creates user-entered queue, ready to run queue.  4. Executes the binary of processes according to SJF algorithm | | | |

**2.3.8 sort()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | sort | | | |
| **Input** | wait\_arr | proc | Initial value:NA | The array of structures containing the data of processes |
|  | time/priority | int | Initial value:NA | Time or Priority of the processes |
|  | cnt | int | Initial value:NA | Number of processes |
| **Output** | None | - | - | - |
| **Description** | This function sorts the classes in ascending or descending according to time or priority | | | |

**2.3.9 swap()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | swap | | | |
| **Input** | xp | int\* | Initial value:NA | - |
|  | yp | int\* | Initial value:NA |  |
| **Output** | Return type: | NA |  | - |
| **Description** | function used to swap the values of the memory pointed by passed integer pointers | | | |

**2.3.10 only\_name()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | only\_name | | | |
| **Input** | full path of the process entered by the user | char\* | Initial value:NA | - |
| **Output** | Return type | NA | - | - |
| **Description** | function used by simulator program to extract the process binary file name from the entered path | | | |
| **Pseud**  **Code** | 1. Take the complete path of binary of processes  2. extract the name of the process and return it | | | |

**2.3.11 file\_checker()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | file\_checker | | | |
| **Input** | path of the process | char\* | Initial value:NA | - |
| **Output** | returns success or failure macro after checking the availability of process binary in the given path | int | - | - |
| **Description** | function used by simulator program for checking the existence of process binary | | | |
| **Pseud**  **Code** | 1. Validate the binary of process entered by the user  2. if present return 0, else -1 | | | |

**2.5 Structure used**

* **Proc structure was created to store parameters for each entered binary.**

typedef struct

{ char name[MAX\_SIZE]; //For storing name of entered binary

char path[MAX\_SIZE]; //For storing path of the entered binary

int time\_req; //For storing burst\_time of entered binary (if applicable)

int prio; //For storing priority of entered binary (if applicable)

} proc;

**2.5 Use Case Diagram**

**2.6 Design and Implementation Constraints**

The system is built using the C language.

### 2.7 Security

TCP itself an secure network connection.