

**Custom Messaging Application**

Low-Level Design Version Draft v0.2

**Document Control:**

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

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

|  |  |
| --- | --- |
| **Employee ID:** | **Name** |
| 46290140 | Sasumalli Durga Bhavani |
| 46290142 | Harshitha Manduri |
| 46287669 | Chaithra J Shetty |
| 46291225 | Satrasala Supraja |
| 46290062 | Shireesha Vennapusa |

|  |
| --- |
| **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 insight into the Custom Messaging Application. It is a system in which the user can register and afterwards login to the system using the valid username and password. After successful login attempt, the user will be able to view rest of the users that are also logged in at that time. Once the user is registered, their data i.e., username and password are stored on the server in a structured format.

**1.1 Purpose**

The purpose of this document is to describe the low-level design flow of the Custom Messaging Application

### 1.2 Document Conventions

TBD (To be continued).

### 1.3 Intended Audience and Reading Suggestions

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

**1.4 References**

The references are:

1. System Requirements Specification Document

## 2. Detailed System Design

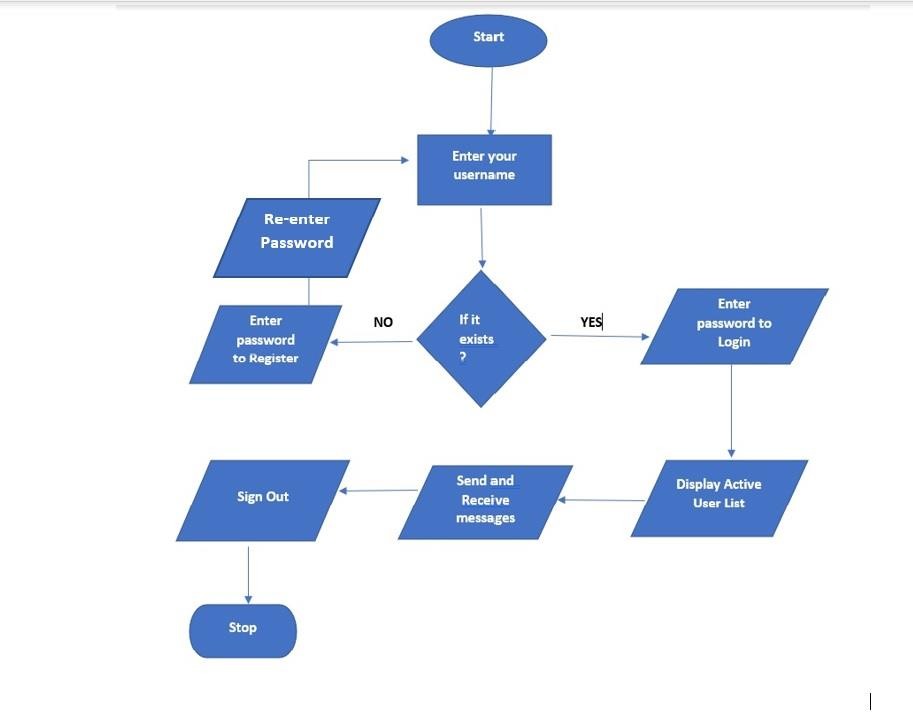
**2.1 Design Descriptions:**

The project aims to create and develop a server to client chat application. While registration it asks for the username and password from the user along with other valuable information like name etc. It stores all this data into the server for validation part later. The messages can very efficiently be transferred among the users into this system.

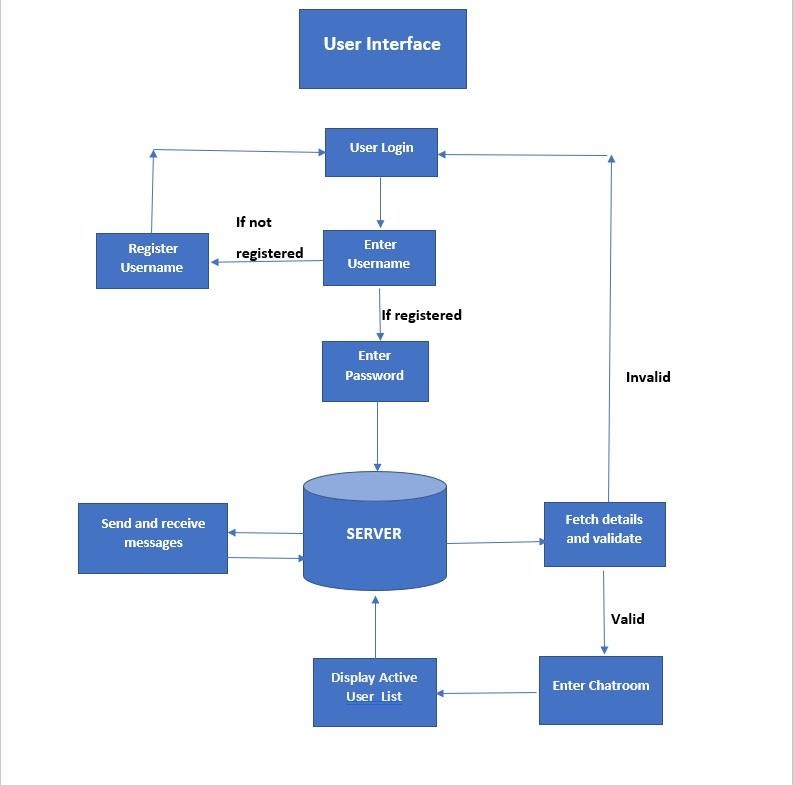
**Main menu:**

User gets options to Register, Login or to Exit the Application.

**2.2 Flowchart**



# User/Client Interface Detailed Data Flow



**Flow Chart for User Registration:**

**Diagram

Description automatically generated**

**2.3 Storyboard**

**2.3 Modules**

**2.3.1 main\_menu.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 login()**

| **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 register()**

| **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 group\_chat ()**

| **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 private\_chat()**

| **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 ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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

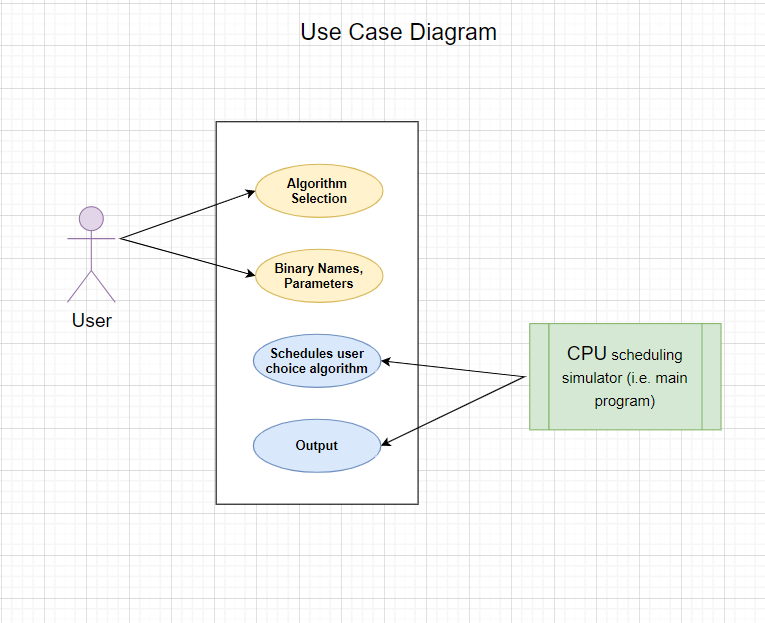
char path[MAX\_SIZE]; //For storing path of the entered binary

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

NA