

**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 Veenapusa |

|  |
| --- |
| **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

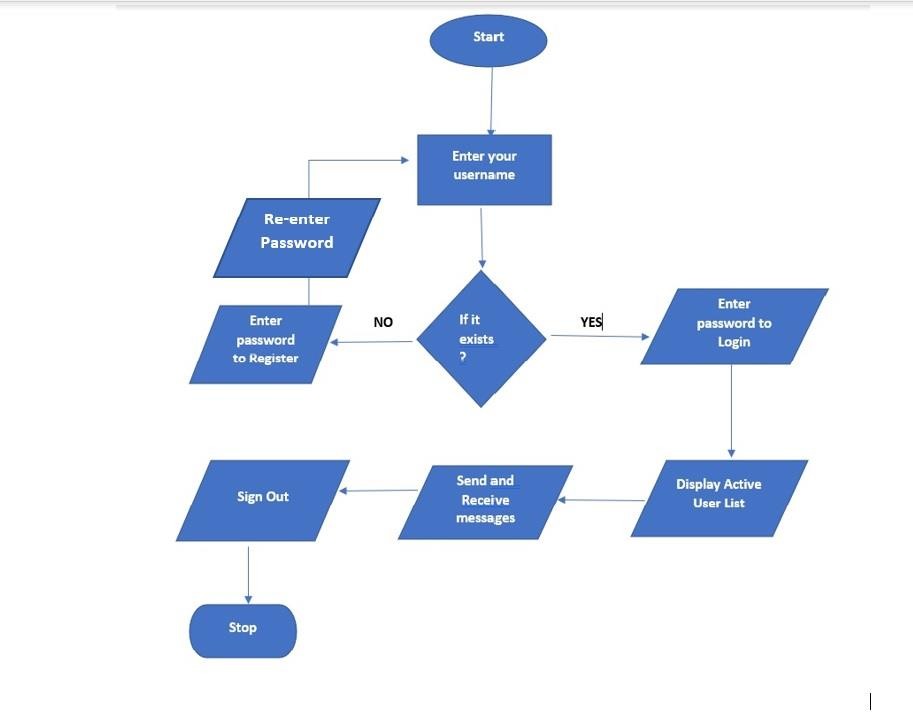
**2.1 Design Descriptions:**

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.

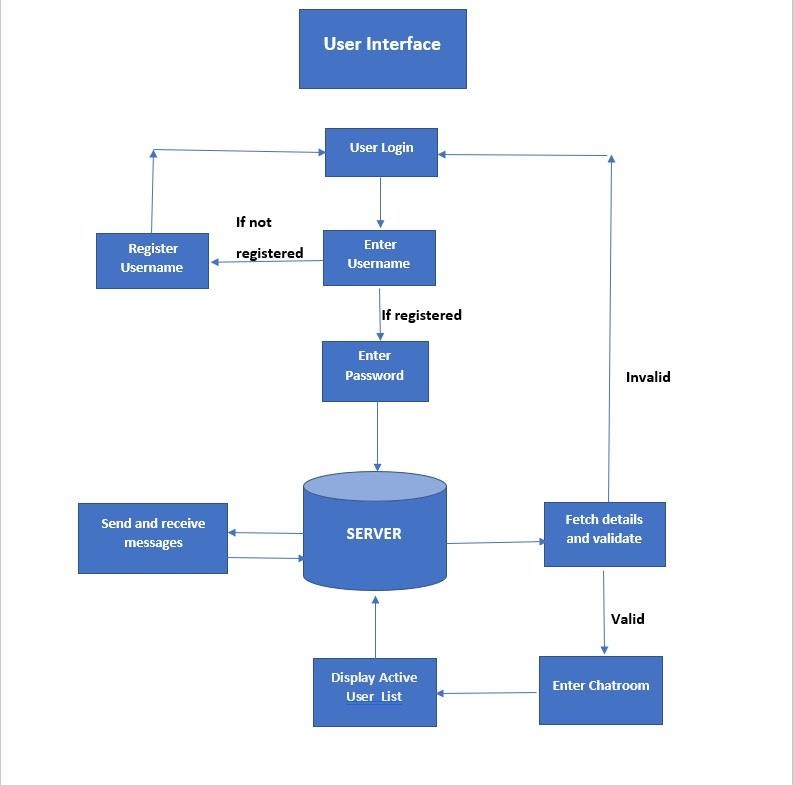
**Main menu:**

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

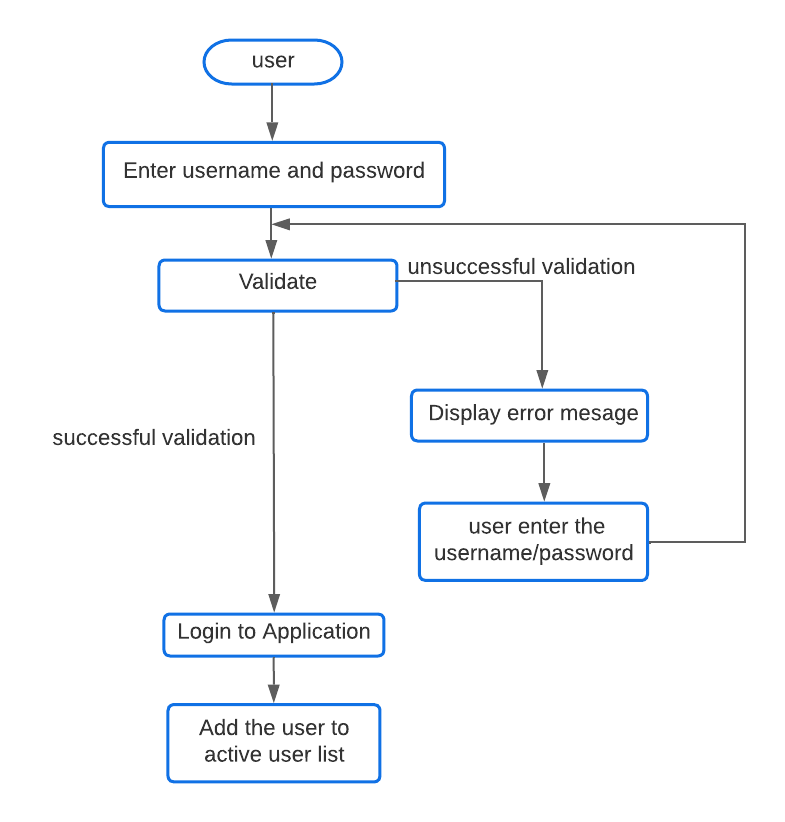
**2.2 Flowchart**



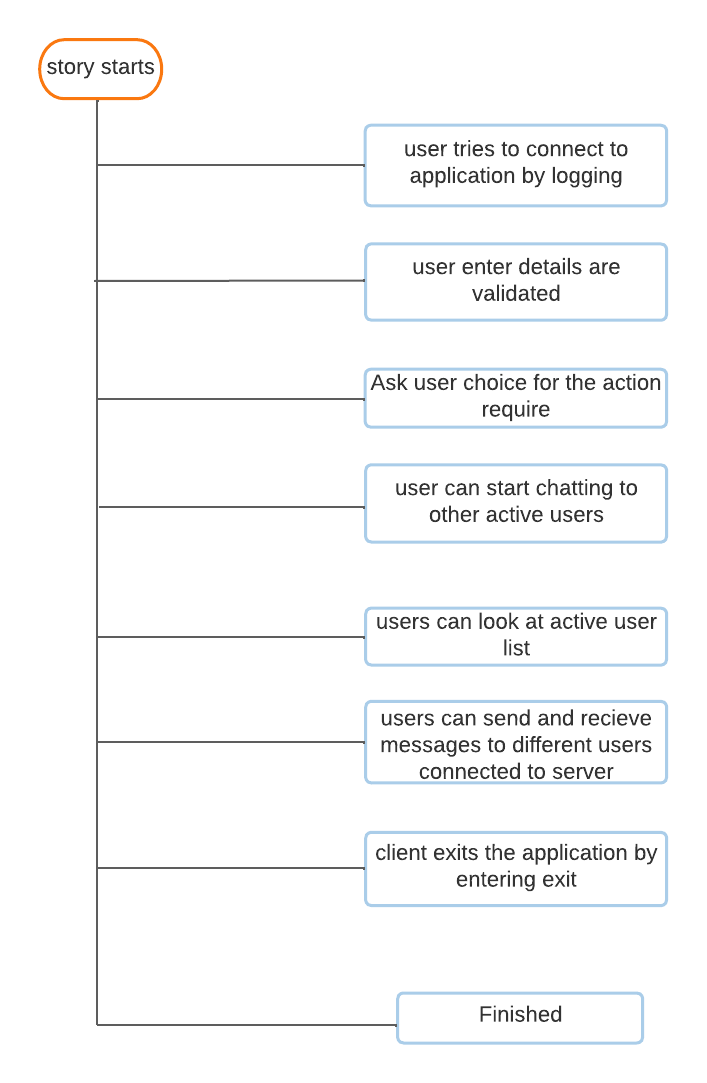
# User/Client Interface Detailed Data Flow



**Flow Chart for User Registration:**



**2.3 Storyboard**



**2.3 Modules**

**2.3.1 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 ready to run the required functions | | | |

**2.3.2 enter()**

| **Name** | enter | | | |
| --- | --- | --- | --- | --- |
| **Input** | Parameter Name | int socfkd |  | Take the username |
| **Output** | Return value type | int |  | NA |
| **Description** | This program is used to take username | | | |
| **Pseudo Code** | 1. Validate the inputs based on the menu level.  2. If invalid return , exit | | | |

**2.3.3 send\_msg\_handler ()**

| **Name** | send\_msg\_handler | | | |
| --- | --- | --- | --- | --- |
| **Input** | Parameter Name | NA | Initial value:NA | - |
| **Output** | Return value type | void |  | - |
| **Description** | This is a send message function used to send messages to users | | | |
| **Pseud**  **Code** | 1. Takes the user input message and send to the respective user | | | |

**2.3.4 recv\_message\_handler ()**

| **Name** | recv\_message\_handler | | | |
| --- | --- | --- | --- | --- |
| **Input** | Parameter Name | NA | Initial value:NA | - |
| **Output** | Return value type | void |  | - |
| **Description** | It takes the receive message from the send function to respective user | | | |
| **Pseud**  **Code** | The user recieves the other user message | | | |

**2.3.5 choice()**

| **Name** | choice | | | |
| --- | --- | --- | --- | --- |
| **Input** | Parameter Name | NA | Initial value:NA | - |
| **Output** | Return value type | void |  | - |
| **Description** | it handles the which choice does the client have selected and also prints error messages | | | |
| **Pseud**  **Code** | 1.It takes the user choice for which action to perform | | | |

**2.3.6 catch\_ctrl\_c\_and\_exit ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | catch\_ctrl\_c\_and\_exit | | | |
| **Input** | Parameter Name | sig | Initial value:NA | - |
| **output** | Return value type | int |  | - |
| **Description** | When the user type the message exit it exit from application | | | |
| **Pseud**  **Code** | When user enter exit it passes signal to exit from application | | | |

**2.3.7 store\_msg ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | store\_msg | | | |
| **Input** | Parameter Name | arr | Initial value:NA | - |
| **Output** | Return value type | void |  | - |
| **Description** | the recieved message will be stored in a file in a certain format with timestamp | | | |
| **Pseud**  **Code** | the message will be stored | | | |

**2.3.8 send\_message ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | send\_message | | | |
| **Input** | s | char | Initial value:NA | - |
|  | uid | int | Initial value:NA | - |
|  | status | char | Initial value:NA | - |
| **Output** | None | - | - | - |
| **Description** | Send message to all clients except sender | | | |

**2.3.9 p\_chat ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | p\_chat | | | |
| **Input** | cli | client\_t\* | Initial value:NA | - |
|  | sockfd | int\* | Initial value:NA |  |
| **Output** | Return type: | NA |  | - |
| **Description** | makes the private chat users BUSY and send some messages to client | | | |

**2.3.10 g\_chat ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | g\_chat | | | |
| **Input** | cli | Client\_t\* | Initial value:NA | - |
| **Output** | Return type | NA | - | - |
| **Description** | sends active groups data to the client and makes server ready for group chat | | | |
| **Pseud**  **Code** |  | | | |

**2.3.11 handle\_client ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | handle\_client | | | |
| **Input** | arg | Void \* | Initial value:NA | - |
| **Output** | Return type | NA | - | - |
| **Description** | uses all the functions and performs all communication between client and server | | | |
| **Pseud**  **Code** |  | | | |

**2.3.12 receive\_file()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | receive\_file | | | |
| **Input** | sockfd | int | Initial value:NA | - |
| **Output** | Return type | NA | - | - |
| **Description** | it serves the sending option in the menu,  it opens a file and writes the data coming from client | | | |
| **Pseud**  **Code** |  | | | |

**2.3.13 send\_file ()**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | send\_file | | | |
| **Input** | sockfd | int | Initial value:NA | - |
| **Output** | Return type | NA | - | - |
| **Description** | it serves the recieving option in the menu, it sends file data and takes the file choice from client and provide the file to client | | | |
| **Pseud**  **Code** |  | | | |

**2.5 Structure used**

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

typedef struct{

struct sockaddr\_in address;

int sockfd;

int uid;

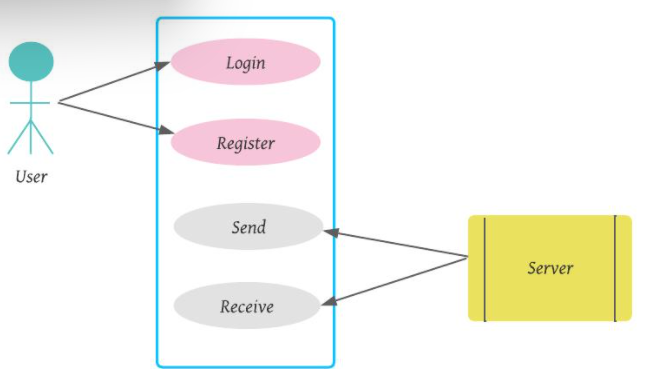
char name[32];

char password[8];

char status[10];

} client\_t;

**2.5 Use Case Diagram**



**2.6 Design and Implementation Constraints**

The system is built using the C language.

### 2.7 Security

NA