# Programmer's guide of messenger_simulator application

## Introduction to program functionality:

This program creates and environment for the user to communicate with each other

## Problem of I/O:

We need to store the messages and account information somewhere.

## Solutions for I/O:

Using two .txt files one for account information and other for saving the messages log as database.

## Data structure:

1D dynamic array

## Exception handling:

Is used for opening the files and checking for errors

## OOP:

Used for storing member objects and log objects

## IDE used:

Code blocks

## Program data flow:

First the program will check the existence of two files ("Data.txt","message_logs.txt").

If they do not exist the program will create them.

After having these two files available than **menu()** function will be called. In this function

we first send a char buffer to check how many lines is in "Data.txt" file are there, Every line is a member object, than allocate an dynamic array and fill each object with each line.

We call **read_member()** function to read the data from "Data.txt" file

Each line in Data.txt file is formatted in this form:

*(Username)' '(Password)' '(Nickname)'\n'*

1. If user selects login:
   program calls **login()** function and we check the user input and compare it with every member object attributes to see if they are the same or not we use method called **u_p_is_match()** from member class for this problem.
   We call **Dashboard()** function after a successful login, we receive the index of username and password match as member_id as parameter in this function. Now **read_member()** function is called to read the "message_logs.txt" file and we do the same technique as we did for reading members.

   Each line in message_logs.txt is formatted in this form:
   *(Local time)' '(Sender Username)' '(Target Username)' '(Message)'\n'*

   Program will print the messages that was sent to them(Inbox),And call the function **sendmessage()** If user select it,
   Program will print all available users, the program will take the local time of system using **currentDateTime()** function and the message etc. ,than add it to "message_logs.txt" file.

2. If user selects create new account:
   The function **addaccount()** will be called and a new line will be added to Data.txt file.