**IMPLEMENTATION AND TESTING:**

**Implementation:**

**Implementation is the stage of the project where the theoretical design is turned in to a working system. The implementation state is a system project in its own right.**

**It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, training of staff in the changeover procedure and evaluation of change over methods.**

**Once the planning has been completed, the major efforts are to ensure that the program in the system is working properly.**

**At the same time concentrate on training user staff. When the staff has been trained a full system can carry out.**

**The various activities involved while implementing a project :-**

* **End user education and training**
* **Training on application software**
* **System testing**
* **Parallel run and change over to new system**
* **Post implementation review**

**Testing:**

**System testing is a critical aspect of Software Quality Assurance and represents the ultimate review of specification, design and coding. Testing is a process of executing a program with the intent of finding an error.**

**A good test is one that has a probability of finding an yet undiscovered error. The purpose of testing is to identify and correct bugs in the developed system. Nothing is complete without testing. Testing is vital in the success of the system.**

**In the code testing the logic of the developed system is tested. For this every module of the program is executed to find an error. To perform specification test, the examination of the specification stating what the program should do and how it should perform under various conditions.**

**System testing does not test the software as a whole, but rather the integration of each module in the system. The primary concern is the compatibility of individual module. One has to find areas where modules have been design with different specifications of data lengths, type and data element name.**

**CODE:**

The GET TO GATHER (G2G) application is developed used awt (Abstract Window Toolkit).

The **java.awt** package is much useful for creating user interfaces and for painting graphics and images. A user interface object such as a button or a scrollbar is called, in AWT terminology, a component. The Component class is the root of all AWT components. Some components fire events when a user interacts with the components. A container is a component that can obtain components and other containers. A container can also have a layout manager that controls the visual placement of components in the container.

The **java.awt** package implements different interfaces like **LayoutManager**, which defines the interface for classes that know how to layout Containers.

**Paint** interface defines how color patterns can be generated for Graphics2D operations. A class implementing the Paint interface is added to the Graphics2D context in order to define the color pattern used by the draw and fill methods.

The **java.net package** provides the classes for implementing networking applications. Using the socket classes, one can communicate with any server on the Internet or implement their own Internet server. A number of classes are provided to make it convenient to user Universal Resource Locators (URLs) to retrieve data on the Internet.

The java.net package implements different interfaces like **DatagramSocketImplFactor*y*** for implementing data gram socket implementations. Classes DatagramSocket to create actual socket implementation use it.

**SocketImplFacotry** interface defines a factory for Socket implementations. It is used by the classes socket and ServerSocket to create actual socket implementations.

**SocketOptions** interface of methods to get/set socket options. Is implemented by SocketImpl and DatagramSocketImpl.

**TESTING APPROACH:**

##### **COMPILING TEST**

It was a good idea to do our stress testing early on, because it gave us time to fix some of the unexpected deadlocks and stability problems that only occurred when components were exposed to very high transaction volumes.

##### **EXECUTION TEST**

This program was successfully loaded and executed. Because of good programming there were no execution errors. The complete performance of the project “INTRANET CHATTING” was good.

##### **OUTPUT TEST**

The successful output screens are placed in the output screens section above with brief explanation about each screen.

**STORAGE TESTING**

**This determines the capacity of the new system to store transaction data on a disk or on other files. The proposed software has the required storage space available, because of the use of a number of hard disks.**

**PERORMANE TIME TESTING**

**This determines the amount of the time used by the system to process transaction data.**

**In this phase the software developed Testing is exercising the software to uncover the error and ensure the system.**

**UNIT TESTING:**

**Unit testing focuses first on the modules in the proposed system to locate errors. This enables to detect errors in the coding and logic that are contained within that module alone. Those resulting from the interaction between modules are initially avoided. In unit testing step each module has to be checked separately.**

**A Unit corresponds to a screen/form in the package. Unit testing focuses on verification of the corresponding class or Screen. This testing includes testing of control paths, interfaces, local data structures, logical decisions, boundary conditions, and error handling. Unit testing may use Test Drives, which are control programs to co-ordinate test case inputs and outputs, and Test stubs, which replace low-level modules.**

**INTEGRATION TESTING:**

**Integration testing is used to verify the combining of the software modules. Integration testing addresses the issues associated with the dual problem of verification and program construction. System testing is used to verify, whether the developed system meets the requirement.**

**WHITE BOX TESTING:**

**White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.**

**Types of White Box Testing:**

* **API testing :- testing of the application using public and private APIs**
* **Code coverage :- creating tests to satisfy some criteria of code coverage**
* **Fault injection methods :- improving the coverage of a test by introducing faults to test code paths**
* **Mutation testing methods**
* **Static testing :- White box testing includes all static testing**

**BLACK BOX TESTING:**

**Black box test design treats the system as a “black-box”, so it doesn’t**

**explicitly use knowledge of the internal structure.**

**Black box test design is usually described as focusing on testing functional requirements. Synonyms for black box includes: Behavioural, functional, opaque-box and closed-box.**

**A Simple Black Box Specification:**

**Black Box Testing is testing technique having no knowledge of the internal functionality of the system. This testing technique treats the system as black box or closed box. Tester will only know the formal inputs and projects results. Tester does not know how the formal inputs and projected results. Tester does not know how the program actually arrives at those results.**

**Hence tester tests the system based on the functional specification given to him. That is the reason black box testing is also considered as functional testing.**

**This testing technique is also called as behavioural testing or opaque box testing or simply closed box testing. Although black box testing is a behavioural testing, Behavioural test design is slightly different from black box test design because the use of internal knowledge is not illegal in behavioural testing.**

**Test Case:**

**Case 1 : For Login**

* **Cursor position in username field**
* **Validations for wrong entry of username**
* **Cursor back to username field and re-enter the username**
* **Cursor position to Password field on pressing Tab key**
* **Validations for wrong entry of password**
* **Cursor back to password field and re-enter the password**

**Case 2 : For Registration**

* **Cursor position in username field**
* **Validations for wrong entry of username**
* **Cursor back to username field and re-enter the username**
* **Cursor position to Password field on pressing Tab key**
* **Validations for wrong entry of password**
* **Cursor back to password field and re-enter the password**
* **Cursor position in Server name**
* **Enter the valid Server name**
* **Cursor position in Port no**
* **Enter the correct Port no**