

TEAM 3

BATCHANGARI ROHITH CHANDRA GOUD

BHARGAV VECHA

VINAY KUMAR REDDY KANDULA

PRODUCT NAME: PRIVATE CHAT BOX

### **A. Goals and Exit Criteria**

#### **i. Quality Goals that need to be met for test phase to exit.**

Quality goals of testing include Written test cases for requirements: In this phase the test cases are written for requirements that are to be met. In this project, the application the following are the expected requirements to work as expected.

1. Proper Functionality and response from the server.
2. Saving the user's id.
3. Authorizing the user with respective chat room names.
4. The text messages should be displayed as soon as they are sent.

The other quality is of written test cases that are successfully executed: Here the test cases that are written to test whether the application is working as expected without any errors and maximum efficiency.

Finally, quality of the application will also depend on the bugs that are yet to be verified, when discovered in the final phase of testing before releasing the product or at the time of user testing.

#### **ii. Robustness goals of the product.**

The application depends on a server in order to store the information of user and the messages that are being sent from one another, hence the application must be robust in doing the following tasks. The application should not halt anytime when being used. It must make sure that all the text messages are displayed properly and based on the time they are sent so that the messages are in chronological order and not in a random fashion. The application is expected to handle multiple number of users using it at the same time.

### iii. Schedule goals of the project

The schedule of the project can be depicted as different phases, the following is a table showing abstract information of Software Development Life Cycle of the project.

Project Phase	Phase Goals
Planning of project	A detailed plan on the project process
Design of application	Design of the UI and ML model architecture
Phase 1: Development of application	Initial development of Application
Phase 1: Testing the product	Testing of initial development
Phase 2: Development of application	Resolving and addressing remaining issues in the application that have been left out in the first development phase.
Phase 2: Testing	Testing the application.
Final product implementation	Final changes, improvements and deployment of the application

The software is expected to be completed within the mention time and schedule, considering any minor bugs and errors, with taking necessary measures to rectify and correct them.

### iv. Performance and efficiency goals of the product

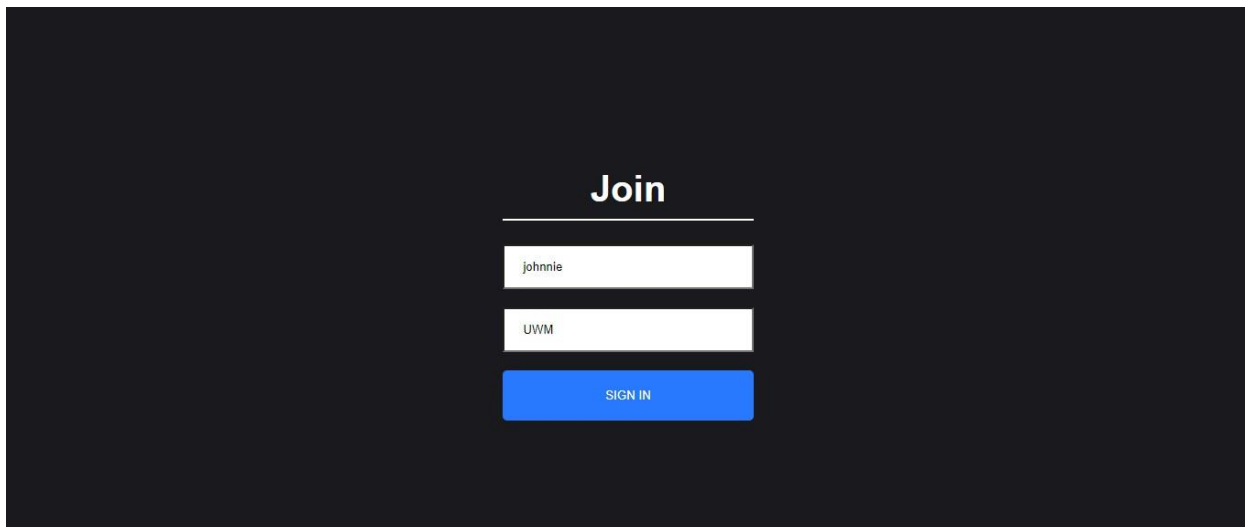
The key aspects of performance and efficiency of the product being developed are as follows:

As this software is for texting and sending information through a medium, it must do the assigned task quickly, there must not be any lag between the user and the server. The messages that are being sent must be displayed synchronously, without any delay. The information should not be misplaced or lost. The messages should not be grouped and be sent at once from a single user that has been sent over time. It must be highly responsive and should exhibit high performance. Efficiency goals of this application is that it should be able to handle multiple or a greater number of users using a smaller number of resources. It must be able to handle large number of messages that are being exchanged between the users very efficiently.

### B. Items to be Tested/ Inspected:

Initially, we check whether the key is generated and stored in the server. When a user wanted to join a chat room, he enters the room key. If the key already exists, then the user is added to the chat room. We check for the name of the user whether it is saved correctly or not. In the chat room, we check whether the messages sent are displayed to every user and the emoji's whether they are displayed in graphic representation or in symbols. We also investigated the server issues. We checked if the server is getting timed out and fixed it.

## User Interface:

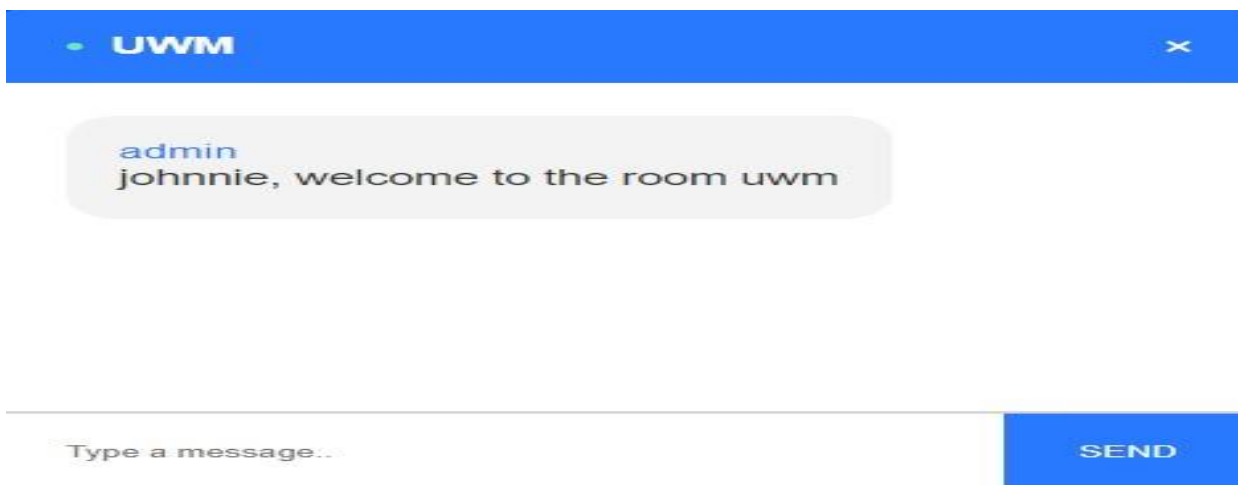
A dark-themed user interface for joining a room. It features a central 'Join' title, followed by two input fields: the first contains 'johnnie' and the second contains 'UWM'. Below these is a blue 'SIGN IN' button.

Join

johnnie

UWM

SIGN IN

A chat room interface. At the top is a blue header bar with 'UWM' and a close button. Below is a message bubble from 'admin' saying 'johnnie, welcome to the room uwm'. At the bottom is a text input field with the placeholder 'Type a message...' and a blue 'SEND' button.

UWM

admin  
johnnie, welcome to the room uwm

Type a message...

SEND

## C. Test Process/ Methodologies:

### i. Unit Testing

In this type of testing, individual modules of the software are tested. Its main goal is to verify each module is working as per the requirements. Unit testing is done during the development phase of each module.

### ii. Inspection

The testing when done during development phase, it is verified by other team members. They go through the module's execution and verify whether it is working as per the requirements and look for any bugs which might affect during the integration of the modules.

### **iii. Black Box Testing**

It is also called as Behavioral/Specification-Based/Input-Output Testing. Black Box Testing is a software testing method in which testers evaluate the functionality of the software under test without looking at the internal code structure. When the user enters an existing key, he is redirected to the chat room. If the key entered is not available in the server, then a new chat room is created, and the user is considered the admin.

### **iv. White Box Testing**

It is also called as Glass Box, Clear Box, and Structural Testing. White Box Testing is based on the application's internal code structure. In white-box testing, an internal perspective of the system, as well as programming skills, is used to design test cases. This testing is usually done at the unit level. This type of testing is done when the tester has knowledge regarding the code structure and designs test cases based on his knowledge.

### **v. Test Metrics**

Code coverage is the percentage of code which is covered by automated tests. Code coverage measurement simply determines which statements in a body of code have been executed through a test run, and which statements have not. The number of bugs or errors occurred is also one of the test metrics. In our project we were able to identify two errors with respect to text boxes which were not accepting numbers, then made necessary changes to the code to fix problem.

### **vi. Test-bug report-fix-retest process**

We have used spreadsheet with multiple access to report the bugs occurred. They are reported so that all the team members can view them and perform the necessary action. We made necessary changes to the code and to the UI design in order to fix the problems and performed testing again, to check if the issue was resolved. We followed this process after every development process until all the errors that have been identified and reported were solved and obtained the results that were expected from the application.

## **D. Resources**

### **i. People**

This project is being developed by a group of three people having knowledge and skills in Java, JavaScript, ReactJS, NodeJS, Express, and CSS. Basic knowledge of software testing is required to perform testing to rectify bugs and errors.

### **ii. Tools**

We will be using GitHub as our management tool as it can store all the previous versions of the software that will help in management, testing and developing as it supports version control. Microsoft visual studio will also be used as it supports a vast number of programming languages and allows debugging which is user friendly.

### **iii. Systems**

For test case development we will be using multi access spread sheets which can be accessed by all the group members to develop and implement the test cases. Subsequent changes and updates can be reported into these spread sheets.

## E. Schedule

### i. Test-case development.

The testing for this project can be performed in multiple phases as development is done in multiple phases. The testing can be carried after each development phase has completed to highlight any bugs or errors occurred after the initial or intermediate development. The team members will create multiple test cases depending on their knowledge and what they have developed for the application, which are updated to a spread sheet that can be access by all the team members and can perform those test cases. Development of good number of test cases that are efficient in testing the software will result in correctness of the product with less time and budget. For this multiple test cases can be developed for example: Response from the server, messages that are being received and sent, short and very long messages should be equally important and be given same priority. All the text information should be sent spontaneously when sent by the user. The emojis entered by the user should be displayed in graphical format rather than symbols.

### ii. Test Execution.

All the test cases that are developed must be executed in order to obtain good results. There might be a few bugs and errors that might be missed if these test cases are ignored. After implementing the test cases, errors and bugs need to be addressed and then perform testing again to measure their success.

### iii. Problem reporting and fixing.

All the bugs and errors that have discovered must be informed to other team members. They will be uploaded into the test case development spread sheet which can be accessed by all the team developers. The member who has developed the specific functionality, in which an error or bug has occurred, will be notified and will be asked to solve it with the help of other team members, by considering all the inputs and selecting the optimal solution.

## F. Risks

**i. Missing Goals:** The product is working as expected and satisfies the requirements that were established during the initial stage of the development.

**ii. Backup Resources Needed:** As the product is a web application, we have made a copy of the application on our personal computers, and on the cloud storage so that it can be deployed easily if any loss happens.

## G. Major Test Scenarios and Test Cases

### i. Boundary value and input domain test cases.

BVT is used to test the boundaries between the equivalence partitions. BVT is most appropriate where the input is a continuous range of values, simply because this is where so many defects exist.

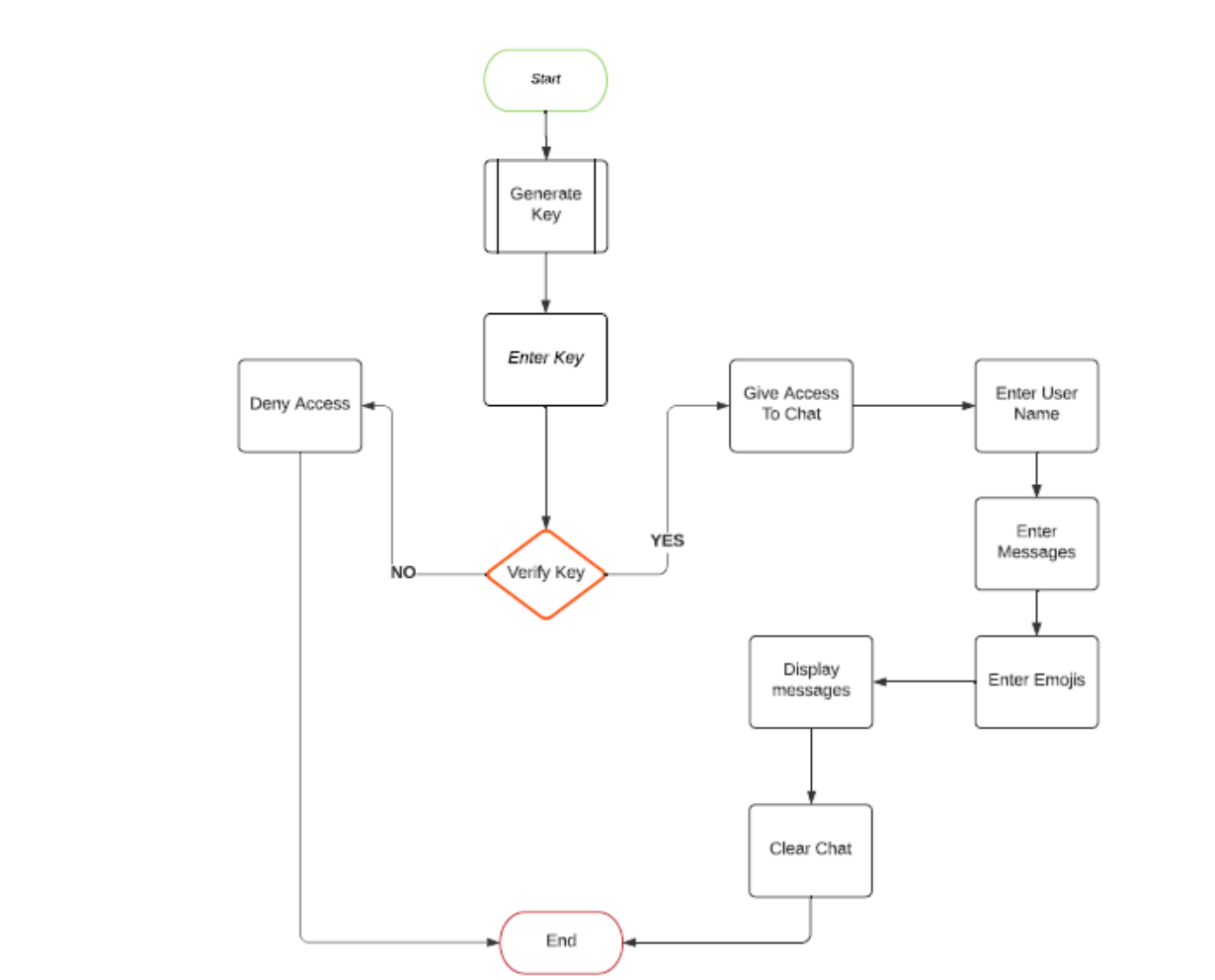
#### Test Scenario

Consider the following situation where a text box which is used to enter the room key allows a character to be entered:

If the user A enters @UWM to create a chat room, then @UWM becomes the unique key which will be used by other users to join that chat room.

If another user B enters @UWm in the text box (room key) then they will not be able to join the chat room created by User A as there is a mismatch of a character “m” as they are case sensitive. Case sensitivity completely depends on the preference of the client, but it will allow more unique key or chat room generations.

## ii. Control path and data flow test cases



Control path and data flow test cases will allow us to check the flow of control and data flow which make sure that the data is provided to next task correctly.

### Test cases

1. Consider the following test case where the chat room key that has been created must be verified before giving access to the other user.
2. The messages that are being entered by the user should only be displayed after pressing the enter key, which can be made sure by using action listener, which will check if the user has pressed enter then only the message or text will be sent through the server and then it will be visible to the other users that are present in that chat room.

### **iii. Integration and intermodular test cases.**

Integration testing will make sure that the different modules of the software application are integrated correctly so that there are no gaps between the modules and can perform as a single system. Modularity will allow the developers to bind multiple functionalities of the system that can be developed by multiple people. In our project we made sure that the User Interface is perfectly integrated with the backend system(server) so that there are no delays or errors while communicating or data transfer.