**1. INTRODUCTION**

Our project is an example of **OFFLINE EMAIL NOTIFICATION** application which is basically based on Email Notifications in mobile phone. It is made of two applications. Client application which runs on user’s PC. Client application which runs on any PC on the network. To send email notification client should run the application in Client system. Client receives the Email Notification in mobile phone through SMS. After Client receiving the SMS in his phone. Receiving the Offline Email Notifications in mobile phone is a very important feature in this application. This Application is worked On Villages and Urban Areas. Companies would like to have communication software wherein they can communicate instantly within their organization. The fact that the software uses an internal network setup within the organization makes it very secure from outside attacks.

**1.1 Motivation**

**Communication:** To develop an instant messaging solution to enable users to seamlessly communicate with each other using python.

**User friendliness:** The project should be very easy to use enabling even a novice person to use it.

**GUI**: Easy to use GUI (Graphical User Interface), hence any user with minimal knowledge of operating a system can use the software. **Platform independence:** The messenger operates on any system irrelevant of the underlying operating system.

**1.2 Objective of the Project**

* These days the world is running through the emailing and which plays a very crucial role in the environment. Mailing has become a mandatory in day to day routine.
* To know what is inside the email received, at least to know whether we got the email or to know by whom we received the email, we must login and check, for that we must be online all the time.
* Where places like remote areas the access of the internet has not been developed. People from such places find difficulties to use their emails.

**1.3 Limitations of the Project**

* We can not get the emails or sometimes we are exposed to unreachable networks, so we can not connect to any of the networks.
* This is the problem that people are getting by poor networks.

**2. LITERATURE SURVEY**

**2.1 INTRODUCTION**

A feasibility study is a sophisticated case description of the complete Design Process and system analysis. This study starts by the problem definition classification. Feasibility is to come to a decision if it’s significant doing or not. One time as the acceptance problem definition has been created; a logical model of the system is developed by the analyst. A look for alternatives is analyzed suspiciously.

Questions that are going to be posed are as follows:

* If the system is developed and implemented will it be used?
* If there was adequate support from the management and from the users for the project.
* Have the users been participated in planning and expansion of the project.
* Will the system make of poorer quality result in any respect or region?

This type of system can be implemented in the association as there is enough hold up from organization and users. Being implemented in Java so that the essential operations are accepted mechanically.

Questions that are going to be posed are as follows:

* Does the required technology be present to do what is has been suggested.
* Does the projected equipment have the technical ability for using the latest system?
* Are there technical assures of accurateness, consistency and information security?
* Whether the project is implemented with intel core i5.
* The environment necessary is windows XP platform in the expansion of system.
* Observer and Factory patterns will bring up to date the results finally
* The language used is Python 2.7& Windows Environment in the development.

It is also called as the cost or budget estimated analysis. It determines if the project development was possible with given resource constraints? Are the benefits that will accrue from the new system worth the costs? What are the savings that will result from the system, including tangible and intangible ones? What is the development and operational costs? These issues are taken care by the Economic feasibility.

In this we study the acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as necessity.

**2.2 EXISTING SYSTEM:**

When the existing system was studied, it was found having some problems, existing system was very time consuming and was not very efficient. The drawback of the existing system has resulted in to the development of new system, which is very user friendly and effective. Existing system was also very low in performance.

While developing the new system all requirements of the end user was taken into consideration. These have been maximum efforts towards overcoming the drawbacks of the existing system, while the new system was designed & developed. While developing the new system all requirements of the end user was taken into consideration. These have been maximum efforts towards overcoming the drawbacks of the existing system, while the new system was designed & developed.

**2.3 DISADVANTAGES OF EXISTING SYSTEM:**

1) There is no Offline Email Notification option had been available.

2) Communication Problem.

3) The performance of existing systems very low.

4) We cannot able to send a mail notification to the mobile directly without internet connection.

**2.4 PROPOSED SYSTEM:**

* The system to be developed here is an Offline Email Notification.
* It is a centralized system.
* It is Client-Server system with server.
* All local clients are connected to the server via LAN.
* There is a two-way communication

**ADVANTAGES OF PROPOSED SYSTEM:**

1) This system is useful for those who cannot afford to have an Internet connection. For example: schools, colleges, small companies, villages, etc.

2) We can able to send a mail notification to the mobile directly without internet connection.

**3. DESIGN & ANALYSIS**

**3.1 INTRODUCTION**

A notification is a message that pops up on the user's device. Notifications can be

triggered locally by an open application, or they can be "pushed" from the server to the

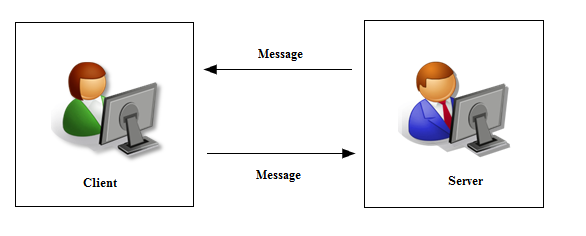
user even when the app is not running. They allow your users to opt-in to timely updates

and allow you to effectively re-engage users with customized content.

**Notification** – a message displayed to the user outside of the app's normal UI.

When it receives a message, the service worker wakes up just long enough to display the notification and then goes back to sleep. Because notifications are paired with a service worker, the service worker can listen for notification interactions in the background without using resources. When the user interacts with the notification, by clicking or closing it, the service worker wakes up for a brief time to handle the interaction before going back to sleep.

**3.2 ARCHITECTURE OF THE PROJECT:**

****

* 1. **HARDWARE REQUIREMENTS:**
     + System : intel core i5.
     + Hard Disk : 80 GB.
     + Ram : 2 GB.

**3.4 SOFTWARE REQUIREMENTS:**

* Operating system : Windows XP/7.
* Coding Language : Python

**3.5 SYSTEM DESIGN**

**3.5.1 DATA FLOW DIAGRAMS**

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also known as a data flow graph or a bubble chart.

**Context Diagram:**

The top-level diagram is often called a “*context diagram”*. It contains a single process, but it plays a very important role in studying the current system. The context diagram defines the system that will be studied in the sense that it determines the boundaries. Anything that is not inside the process identified in the context diagram will not be part of the system study. It represents the entire software element as a single bubble with input and output data indicated by incoming and outgoing arrows respectively.

**Types of data flow diagrams**

**1. Physical DFD**

Structured analysis states that the current system should be first understand correctly. The physical DFD is the model of the current system and is used to ensure that the current system has been clearly understood. Physical DFDs shows actual devices, departments, people etc., involved in the current system

**2. Logical DFD**

Logical DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts.

**The Basic Notation used to create a DFD’s are as follows:**

**Dataflow:** Data move in a specific direction from an origin to a

Destination.

**Process:** People, procedures, or devices that use or produce

(Transform) Data. The physical component is not identified.

**Source:** External sources or destination of data, which may be

People, programs, organizations or other entities.

**Data Store:** Here data are stored or referenced by a process in the

System

DATA FLOW DIAGRAMS

LEVEL:0

OFFLINEEMAILNOTIFICATION

Client1

Send The Notification Of Email

View of report

Client2

**LEVEL:1**

OFFLINE Email Notification

Client1Login

Client2

Check id,

and password

SendSMS

Gmail Server

SMS

Mobiledetails

Serverdetailss

**3.5.2 UML Diagrams**

The UML (Unified Modeling Language) is one of the modeling languages utilized to i) specify, ii) visualize, iii) construct and iv) document a software product.

The UML signifies a group of best engineering performances that have confirmed successful in modeling of both large systems and difficult systems. UML assists project teams to i) communicate ii) explore potential designs and iii) validate the software of the architectural design.

The major goals of the UML are as follows i) Provide users with a ready-to-use, ii) expressive visual modeling language. As a result, they can enlarge and replace significant models. It also provides extensibility and interested methods to expand the core ideas. In the case of programming languages and development processes it is independent.

**HISTORY OF UML**

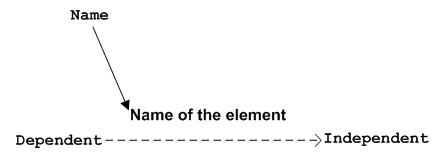
Certain object-oriented languages started to emerge between mid-1970’sto late 1980’s as different methodologists researched with dissimilar advances to object-oriented design and analysis. The no. of recognized modeling languages enlarged from fewer than 10 to more than 50 in the era of 1989-1994.

Many users of Object-Oriented methods had problem is covering total approval in anyone modeling language. In mid-1990, fresh iterations of these methods started to emerge to integrate each other practices, and a small number of evidently famous methods appeared.

UML development started in late 1994 by Jim Rumbaing and Grady Brooch when they started their work on uniting the Grady Brooch and Object Modeling Techniques.

**RELATIONSHIPS**

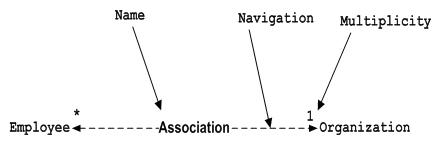
Relationships are used to show semantic links among model elements. Dependency relationship is one in which two things are connected and if one changes, it influences the other element. Dependencies can be applied to recognize connections among the selection of model elements and packages. These relationships are unidirectional relationships and mainly called as “is-a relationship”



A generalization is a connection between two elements in which one element is more general form of the other one. Class inheritance is characterized in this way, but we generally use generalization. Example is packages.



An association is a structural relationship which is used to map one object to other objects or another set of objects. It is also used to recognize the message path between a use case and actor. The association is mainly called as “the glue that ties systems together”.



A realization is a kind of dependency relationship that recognizes a contractual connection between elements of a realizing element. Consider an example as a class executes the behavior of a specifying element in such situation it is considered as an interface. A realization is also used link collaborations and use cases.

**ASSOCIATIONS**

UML treats composites and aggregations to be unique forms of association with unique notations. A skilled association is a plain association with a hint of what information to use when recognizing the target object over a set of associate objects.

Aggregation stands for whole-part relationship. In the aggregation relationship it treats one element as the whole and the remaining elements as its parts.

**DEPENDENCIES**

Dependency relationships are often stereotyped to carry the requirements of particular types of diagrams. The below are few examples:

Extends offers a way of managing performance that is not obligatory in a use case. The not obligatory actions are packaged in an extending use case and associated via <<extends>> dependency.

Includes offers a way of managing performance that is general to a no. of use cases. The optional activities are privileged away, wrapped up in a built-in use case, and linked through an <<includes>> dependency.

**‘Imports’** is a kind of dependency among packages. A receiving package can right to use publicly visible essentials of the package being significant.

|  |  |  |  |
| --- | --- | --- | --- |
| **Relationship** | **Symbol** | **Line Style** | **Arrow Tip** |
| Inheritance |  | Solid | Closed |
| Realization |  | Dotted | Closed |
| Association |  | Solid | Open |
| Dependency | ------------------- | Dotted | Open |

Table: 3.5. - UML Relationships

**3.5.2.1 USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

**USE CASE DIAGRAM:**

**Fig. 5.2.2 Use case diagram showing Client-side cases.**

System

**Client**

**Login.**

**ViewMessage**

**ReceiveMailNotificationMobile**

**ReceiveSMSinMobile**

**ViewMail**

**Logout.**

**3.5.2.2 CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

**CLASS DIAGRAM for Client:**

**Email**

+id

+eid

+ename

+conductedby

+sendEmail ()

+Viewemail ()

**Client**

+email

+sms

+mobileno

+sendNotification ()

+sendMail ()

+ViewMail ()

**Mobile**

+number

+smsgateway

+ReceiveTheSMS ()

**3.5.2.3 SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

**SEQUENCE DIAGRAM:**

: Client

Login

Notification

Viewsms

Email

1: UserId ()

2: Password ()

3: view ()

4: viewsms ()

5: Email ()

**3.5.2.4 ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**ACTIVITY DIAGRAM FOR CLIENT**

Client

ConnectToserver

SendSMS

SendEmailNotifications

ViewNotification

ViewEmail

**4. IMPLEMENTATION**

**4.1 MODULES:**

* Server Module
* Client Module
* Notification Module
* Email Module

**4.1.1 MODULES DESCRIPTION:**

**1) Server Modules:**

A server is a specially equipped computer connected to the internet that allows users with chat client to converse with one another via typed messages in real time.

**2) Client Module:**

A client is a software program that allows users to connect to chat servers and communicate with other user via a chat room.

**3) Notification Module:**

In This Module send a mail notification to the mobile directly without internet connection using TWILIO. Here we will see the subject of email.

**4) Email Module:**

After receiving the notification in mobile phone, we will check the mail box. If it is important, we will check mail otherwise not subject of email.

### 4.2 TESTING & VALIDATION

The purpose of testing is to find out errors. Testing is the procedure of trying to find out every possible fault or fault in a work product. It provides a way to test the functionality of the following i) components, ii) sub-assemblies, iii) assemblies and/or final product. It is the procedure of implementing software with plan of ensuring that the software system attains its requirements and user chance does not fail in an inappropriate way.

We have different types of testing. Every test type addresses a detailed testing requirement. Software testing is a main feature of software quality assurance and represents the essential reuse of requirement. Design and code testing represent attractive differently for the software throughout earlier description and development phase, it was attempted to construct software from a conceptual concept to real execution.

The testing phase contains, testing of the improvement of the system which using various testing techniques such as i) white box testing, ii) control structure Testing.

**TESTING TECHNIQUES**

It contains two types of techniques. One is white box testing and second one is Black box testing.

**WHITE BOX TESTING**

This white box testing is one among the test case blue print techniques that uses the control arrangement of the practical design to attain test cases. After applying white box testing it was recognized that.

Leave Recording System (LRS) software assures that all autonomous paths with in the modules have been exercised at least one time.

1. It has exercised all logical decisions on either 0 (true)
2. or1 (false) sides.
3. It was tested to perform all loops at their margins and within their operational limits.
4. It was tested for the inner data structures to make sure their authority.

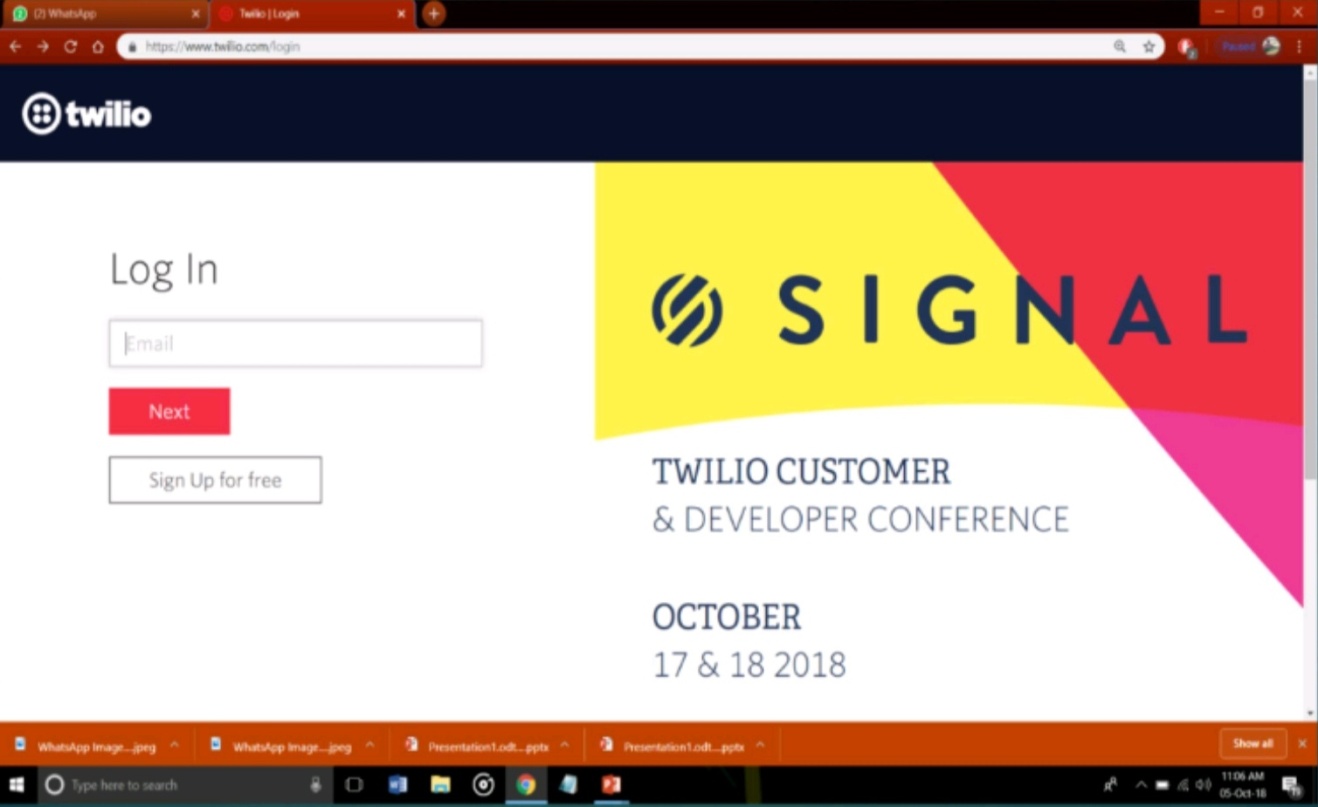
**BLACK BOX TESTING**

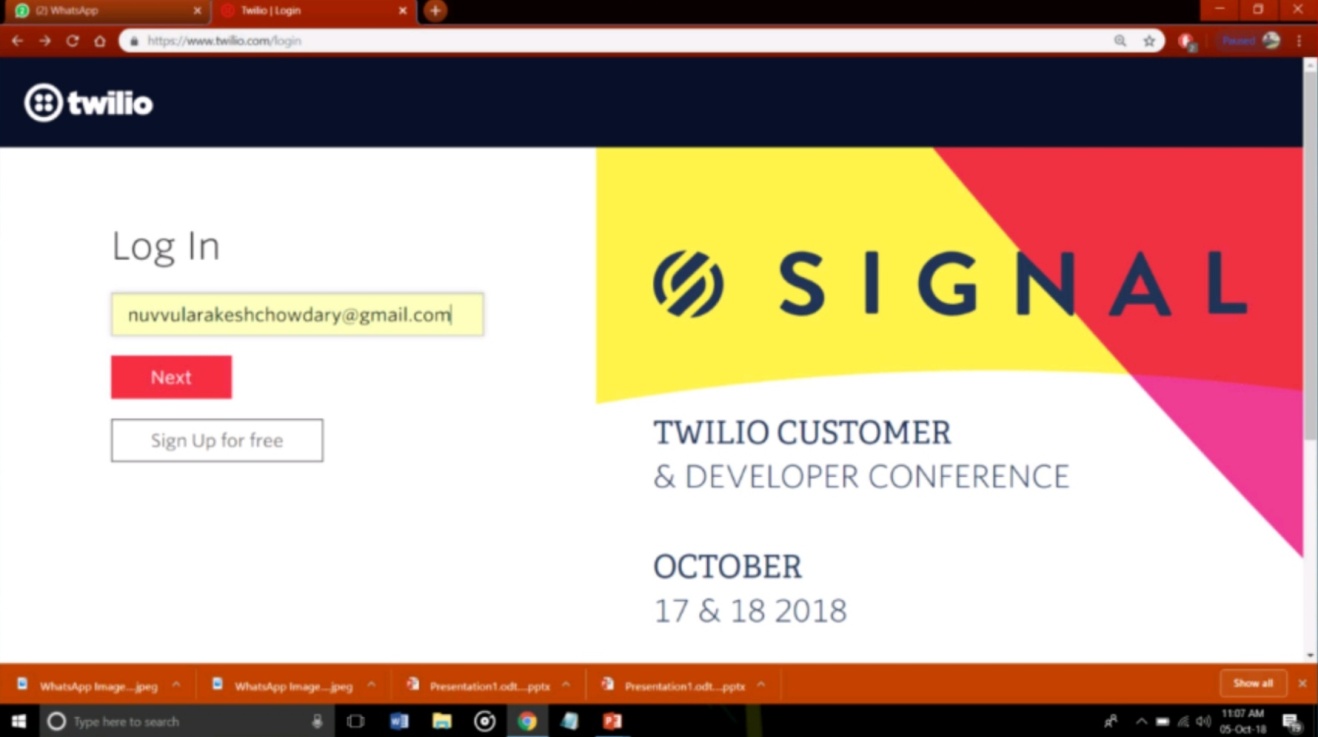
Black- box testing concentrates on functional requirement of the software. It enables to obtain sets of input conditions that will totally work out all functional necessities of a program.

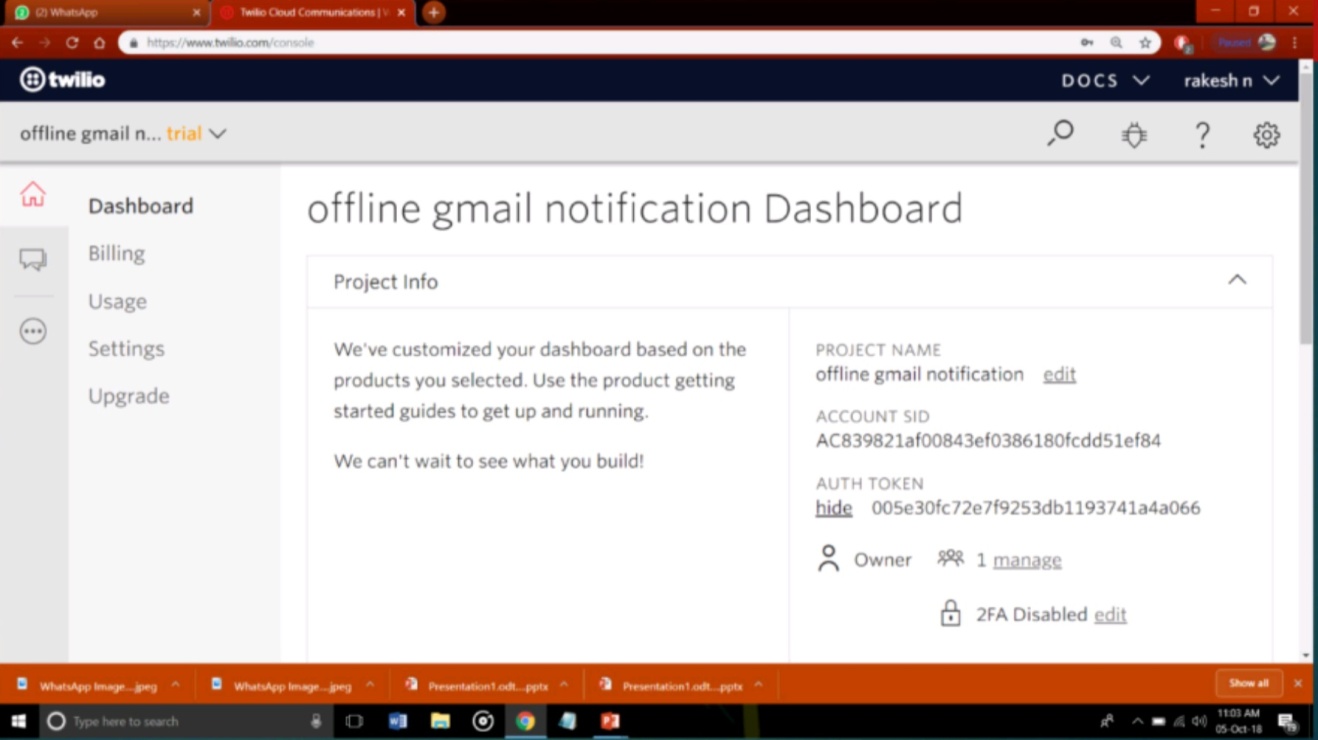
Black box testing tries to locate error in the following group:

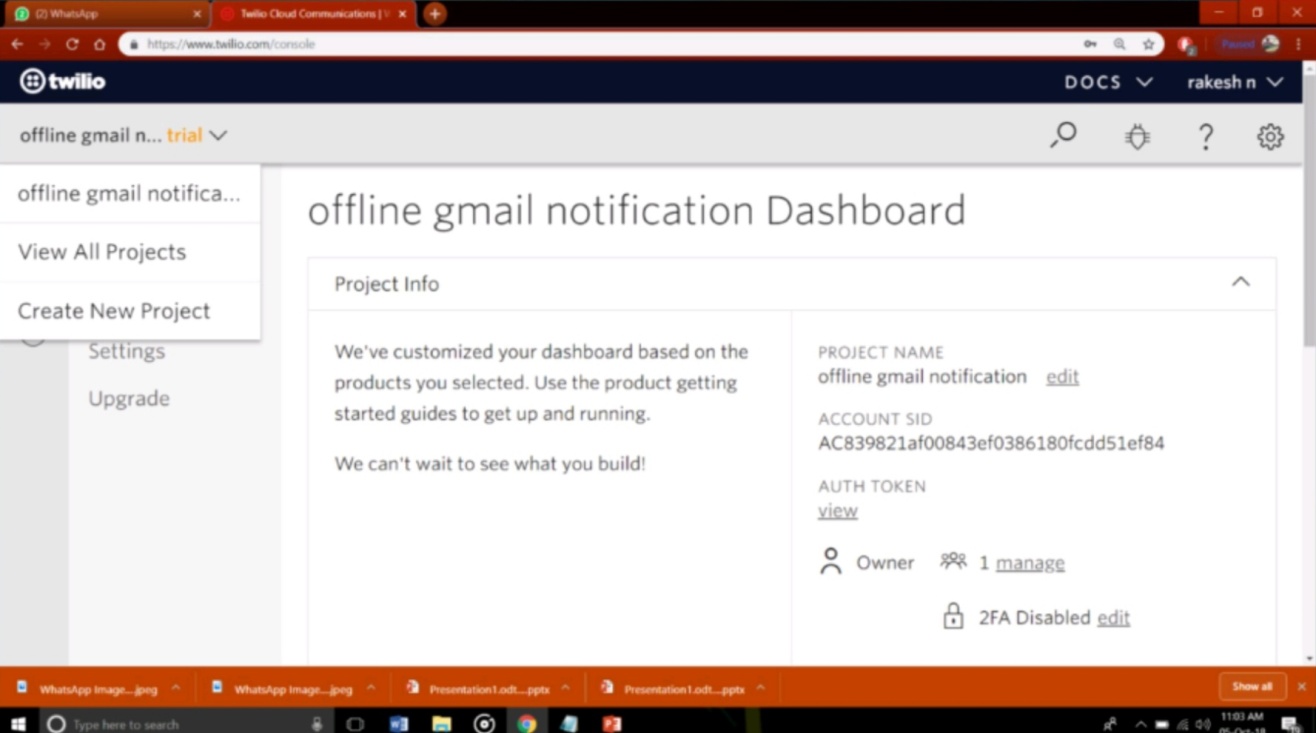
1. Wrong or missing function
2. Interface errors
3. Errors in data structures or outside database access and presentation errors

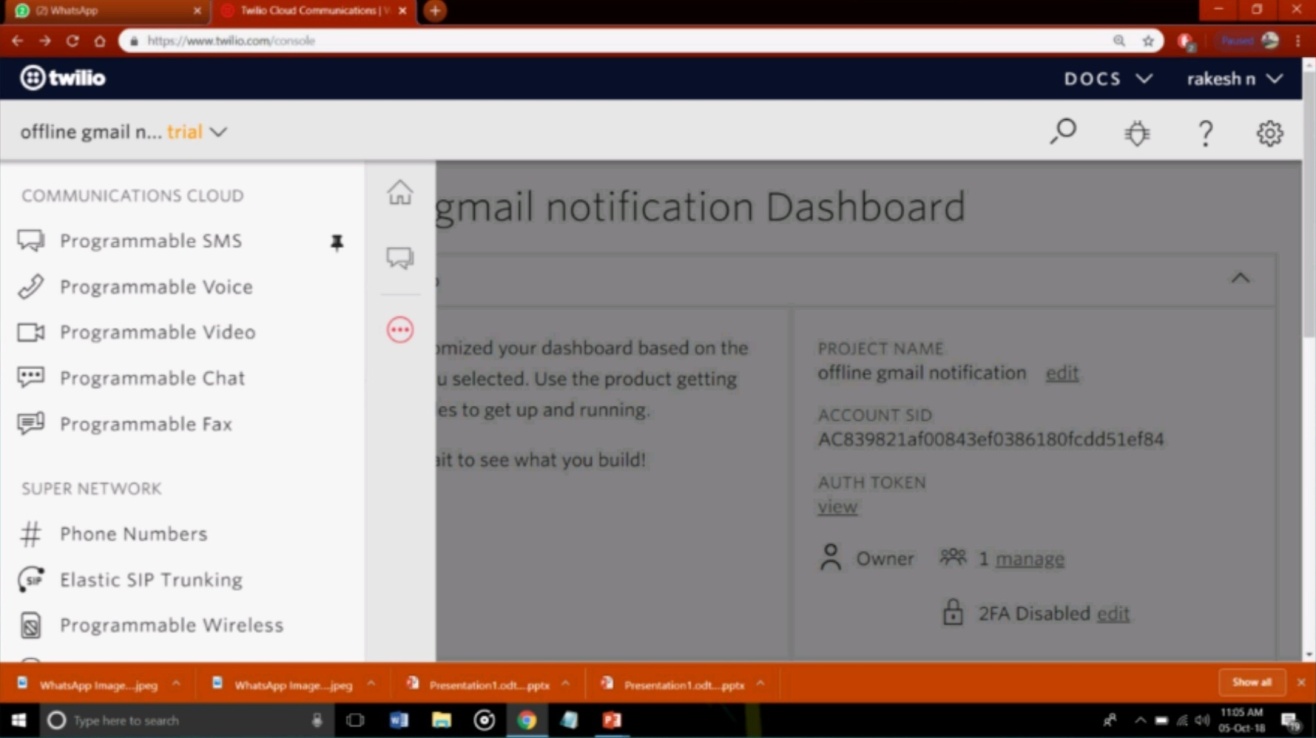
**5. OUTPUT SCREENS**

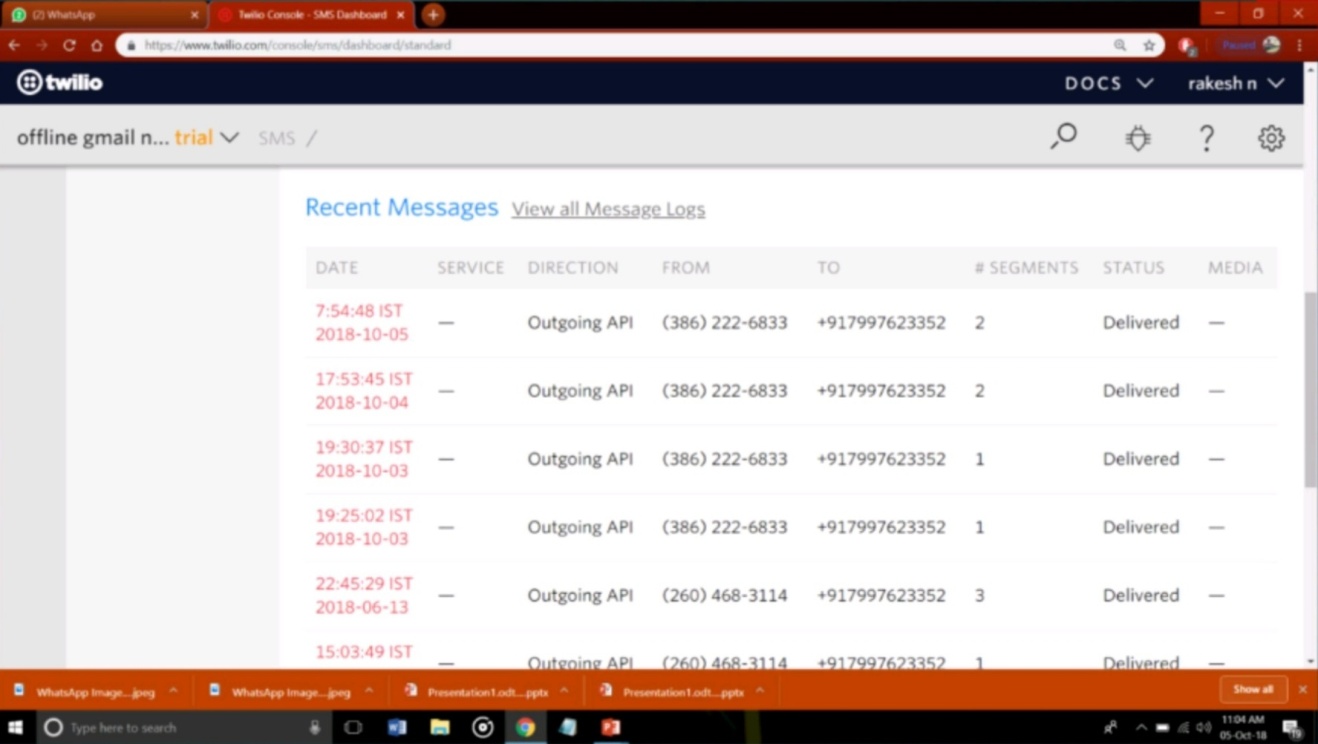


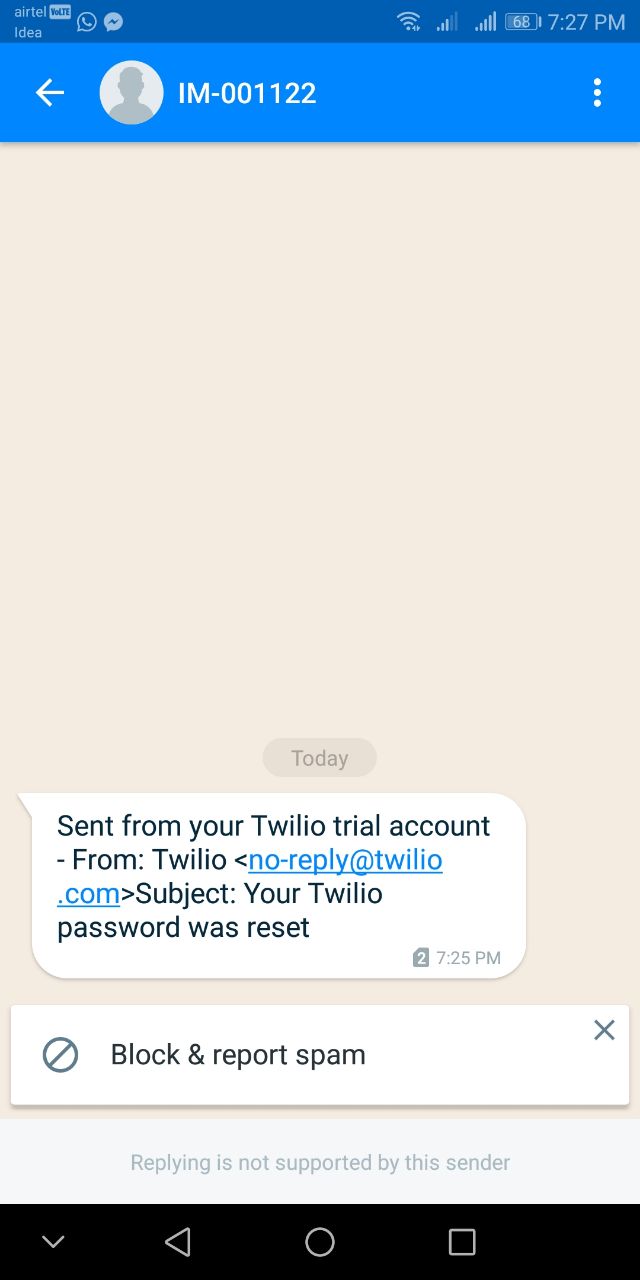


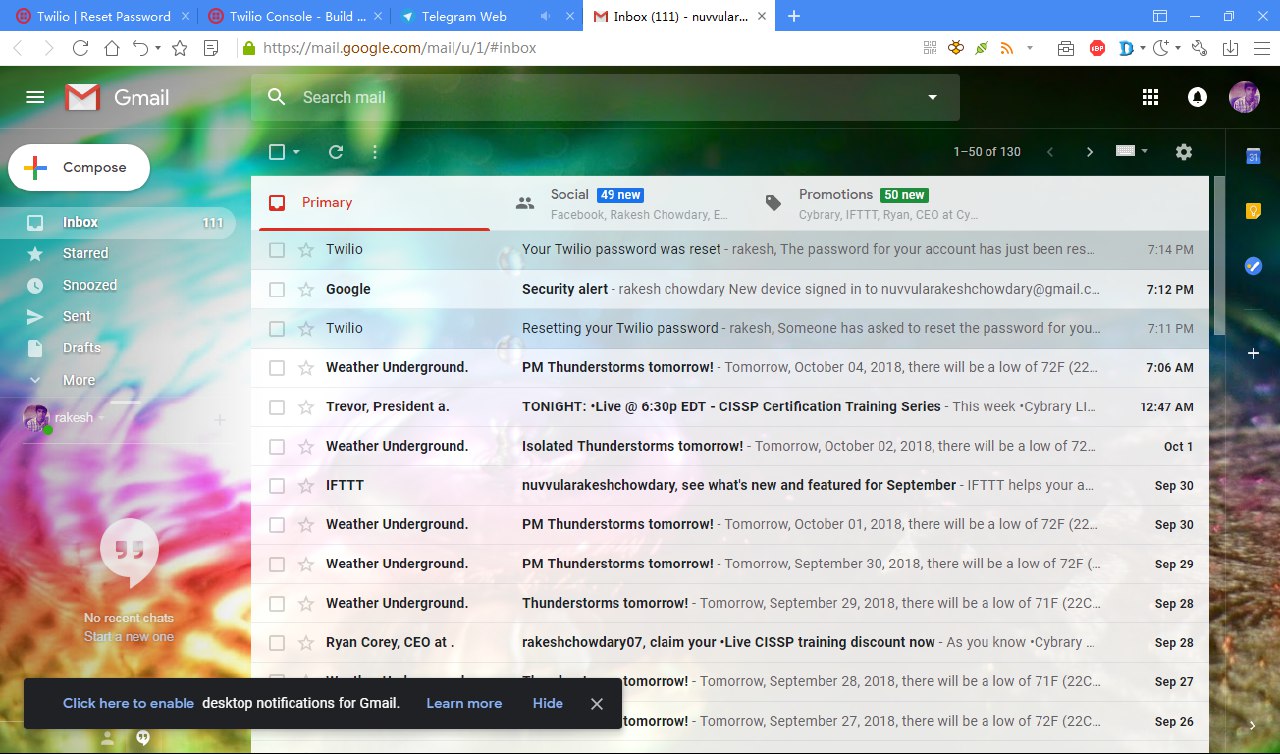












**6. CONCLUSION**

Communication over a network is one field where this tool finds wide ranging application. this project we can able to send a mail notification to the mobile directly. This tool can be used for large scale communication and conferencing in an organization or campus of vast size, thus increasing the standard of co-operation. In addition, it converts the complex concept of sockets to a user-friendly environment. This software can have further potentials, such as file transfer and voice chatting options that can be worked upon later.

**REFERENCES**

[1] Tanenbaum, “Computer Networks”, Second edition

[2] A. Raju,” Advanced Java”

[3] Introduction to Sockets, web.njit.edu/ ~gblank/cis604/Lectures/604Sockets.ppt

[4] Q. Charatan and A. Kans,” Java in two semesters”, 2nd edition McGraw Hills publication,2006

[5] Introduction to Socket Programming, [http://www.cs.rutgers.edu/~pxk/rutgers/notes/sockets/index .html](http://www.cs.rutgers.edu/~pxk/rutgers/notes/sockets/index%20.html)

[6] J. F. Kurose and K. W Ross, “Computer Networking- A Top-Down Approach featuring the Internet”, 2nd Edition (Addison Wesley World Student Edition)

[7] Author P. Burden,” Socket Programming”

[8] Author G. McMillan,” Socket Programming HOWTO”

[9] Socket Concepts: <http://publib.boulder.ibm.com/infocenter/iseries>

[10] W3SCHOOLS: www.w3schools.com