

# **SOCIAL MEDIA BROADCAST CLIENT**

A Project Report Submitted to

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY**

In partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND ENGINEERING**

SUBMITTED BY

**NAJMA NAAZ (10BD1A0561)**

**SHARANYA VR (10BD1A05A6)**

**VAIBHAV BHALLA (10BD1A05A8)**

Under the Guidance of

**(Dr. Ramakanta Mohanty, Professor and HOD,IT Dept.)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY**

(Approved by AICTE, Affiliated to JNTUH)

Narayanaguda, Hyderabad

Year of submission: 2014

---

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **CERTIFICATE**

This is to certify that this Project Report is the bonafide work of the following students:

**NAJMA NAAZ** (10BD1A0561)

**SHARANYA VR** (10BD1A05A6)

**VAIBHAV BHALLA** (10BD1A05A8)

carried out the Project entitled “**SOCIAL MEDIA BROADCAST CLIENT**” under our supervision.

**Mentor**

**Mr.Ramakanta Mohanty**

Professor and HOD,IT Dept

**Branch Co-ordinator**

**Mr K.Siva Rama Krishna**

Associate Professor,CSE

**Head of the Department(CSE)**

**Dr.Siddhartha Ghosh**

Professor ,CSE

**Submitted for Viva Voce Examination held on** \_\_\_\_\_

**External Examiner**

---

## **DECLARATION**

We hereby declare that the results embodied in this dissertation entitled “**SOCIAL MEDIA BROADCAST CLIENT**” has been carried out by us together during the academic year 2013-2014 as a partial fulfillment of the award of the B.Tech degree in Computer Science and Engineering from JNTU-H. We have not copied the same from any source and have not submitted this report to any other university or organization for award of any other degree.

**NAJMA NAAZ** (10BD1A0561)

**SHARANYA VR** (10BD1A05A6)

**VAIBHAV BHALLA** (10BD1A05A8)

## ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance have been a source of inspiration throughout the course of the project work. We are glad to have the opportunity of expressing our gratitude to all of them.

We are thankful to our Mentor **Dr.Ramakanta Mohanty**, Professor and Head of Department Information Technology, Keshav Memorial Institute of Technology. We thank him for his support and guidance throughout the project. . He is a source of inspiration for innovative ideas and his kind support is well known to all his students and colleagues.

We also like to thank the Computer Science Branch Coordinator, **Mr. Siva Rama Krishna** Assoc. Professor, Computer Science and Engineering Dept., Keshav Memorial Institute of Technology. We thank him for providing us timely advice throughout the project work.

We also express our sincere gratitude to the Management and Principal of Keshav Memorial Institute of Technology for their encouragement, facilities provided and support.

Finally, we would like to make a special mention of all our family members and friends who helped us for the successful completion of the seminar report.

**NAJMA NAAZ** (10BD1A0561)

**SHARANYA VR**

(10BD1A05A6)

**VAIBHAV BHALLA**

(10BD1A05A8)

---

## ABSTRACT

*Having a great social media profile is a must for many companies these days. Brand Promotion becomes an easy task, through a social media profile. Companies can notify people about their products and other updates just by posting on these social networking sites. They need to have pages on major social networking sites like Facebook, Twitter, Pinterest, Google+, Reddit, Stumbleupon, LinkedIn etc.*

*Content posting and Content engagement reports is a very important factor for companies to get a feedback about their products and features. This will act as a statistical tool for them to reframe their business strategies. User engagement is also very critical for these pages. For instance, Facebook penalizes pages with low user engagement and any stories shared on those pages will get very less visibility on Facebook. It's a big loss to companies as they are losing on the opportunity to reach more and more customers.*

*But managing these multiple profiles is a mundane task - uploading the same content across accounts, reaching out to all the people who have subscribed to their products from different sites, and analyzing the feedback from different sites, etc.*

*To fulfill these use cases, we need a tool to broadcast content across different social networks and also manage the pages on different accounts. Managing pages can include viewing the Posts, Messages, Content engagement Reports and reaching out to all subscribers of the pages on different sites through a client. Additional features can include Content Scheduling, etc.*

---

# CONTENTS

Certificate	I
Declaration	II
Acknowledgement	III
Abstract	IV
Contents	V
List Abbreviations	VI
List of Figures	VII

## CHAPTER 1

1. Introduction	1
1.1. Motivation	1
1.2. Problem Definition	1
1.3. Objective of Project	2
1.4. Limitations of the Project	2
1.5. Scope of the Project	2

## CHAPTER 2

2. Software Requirement Specifications	3
2.1. Requirements Specification Document	3
2.1.1. Functional Requirements	3
2.1.2. Non-Functional Requirements	4
2.2 Software Requirements	4
2.3 Hardware Requirements	4

## CHAPTER 3

3. Technologies/Tools Used for Used for implementation	5
3.1 Java	5
3.2 Play Framework	5
3.3 HBase	6
3.4 JUnit	6
3.5 Eclipse IDE	6
3.6 HTML	6
3.7 JSON	6

## CHAPTER 4

4. Design	8
4.1 System Architecture	8
4.1.1 Architecture Diagram	8
4.1.2 Modules of the System	8

---

4.2 UML Diagrams	11
4.2.1 Class Diagram	12
4.2.2 Use case Diagram	14
4.2.3 Sequence Diagram	16
4.3 Database Design	18
4.3.1 ER Diagrams	18
 <b>CHAPTER 5</b>	
5. Implementation	19
5.1. Classes used	19
 <b>CHAPTER 6</b>	
6. Integration and Testing	20
6.1 Integration	20
6.2 Testing	25
6.2.1 Test Cases	35
 <b>CHAPTER 7</b>	
7.1. Output Screens	36
7.2. Future Enhancement	43
7.3. Conclusion	43
7.4. References	44

---

## LIST OF ABBREVIATIONS

Acronym	Expansion
JS	Java Script
XML	Extensible Markup Language
JSON	JavaScript Object Notation
API	Application Program Interface
HTML	Hyper Text Markup Language
UML	Unified Modeling Language
SDLC	Software Development Life Cycle
CSS	Cascading Style Sheet

---

## LIST OF FIGURES

S.No.	Fig. No	Name	Page Number
1	Fig 4.1	Architecture Diagram	8
2	Fig 4.2	Class Diagram	13
3	Fig 4.3	Use Case Diagram	15
4	Fig 4.4	Sequence Diagram for Posting	16
5	Fig 4.5	Sequence Diagram for Retrieving all Messages	17
6	Fig 4.6	Sequence Diagram for Replying to a Message	17
7	Fig 4.7	ER Diagram	18
8	Fig 7.1	First Screen	36
9	Fig 7.2	Registration Screen	37
10	Fig 7.3	Index Screen	38
11	Fig 7.4	Broadcast screen	39
12	Fig 7.5	Reach Screen	40
13	Fig 7.6	Accounts Screen	41
14	Fig 7.7	Messages Screen	42

---

# CHAPTER 1

## INTRODUCTION

### 1.1 MOTIVATION

#### **The growing popularity of Social Networking Sites**

Social Networking sites have become a major part of every person's life today. One is easily able to connect to any other person in any corner of the world and hence has become a major means of communication. Social networking websites are, for the most part, easy to use. Most sites are easy to navigate. In fact, many require little knowledge of the internet. In addition to being easy to navigate, social networking websites also make it easier to meet new people online. Also, social networking sites are popular is because many are free to use. In fact, the majority of social networking sites, such as Facebook and Twitter, are free to use.

Social networking sites allow you to learn information about so many things that are shared by other users and provide a platform to reach the masses within few seconds. Most Companies are targeting social media today, for the promotion of their brands. Having a great social media profile is a must for many companies these days. They get to constantly notify people about their latest products through the PAGES provided by these social networking sites.

#### **Brand Management across Social Media through Pages.**

Social Networking site 'Pages' help businesses, organizations and brands share their stories and connect with people. Profiles represent individuals and must be held under an individual name, while Pages allow an organization, business, celebrity, or band to maintain a professional presence on Facebook.

Two really crucial advantages of administering a Page are user analytics and easily-embedded Like buttons and widgets. With the recently overhauled Facebook Insights, you can view all kinds of useful data about user activity on your page, including how many likes and comments you received each day, demographic break-downs and much more. This data can be really useful for understanding the characteristics of your customer base, and knowing which wall posts get the best reaction. This data is not available to individual profiles.

Another thing profiles don't offer is the means to embed a Like button and other Open Graph plugins into your own site. This is a major setback for businesses hoping to grow their audience on Facebook.

### 1.2 PROBLEM DEFINITION

The existing trend for posting content onto the pages involves logging into multiple accounts and posting the same content in different pages across multiple sites which is a mundane task. To view the page's insights such as user activities, likes and comments for each post, it is required to log in to each account separately. It is not easy to compare the statistics across the sites as there is no single interface to view insights. Another drawback with the existing system

is that, organizations have to connect to their subscribers separately for each of their accounts which becomes difficult with increase in the number of subscribers. User Engagement is very critical for these pages. If it is low, then the stories shared on such pages will get less visibility on the site which will turn into a huge loss for the companies.

### **1.3 OBJECTIVE OF THE PROJECT**

This project provides a tool using which organisations and companies can :

- Broadcast post/content onto the pages across different social networks
- Manage content and subscribers
- View Insights and Statistics of different across different accounts
- Direct interaction with page subscribers/customers when they raise an issue on one of the social media channels

### **1.4 LIMITATIONS OF THE PROJECT**

The following are the limitations of the project :

- This project lets the user connect to their Facebook and Twitter profile only.
- This project lets the user connect to only one Facebook profile at a time.
- This project allows the user only to reply to a message but cannot start the conversation.
- The user cannot upload any media files such as pictures or video, they can only post the URL of the files.

### **1.5 SCOPE OF PROJECT**

This project focuses mainly focuses on connecting to a user's Facebook and Twitter profile. Anyone with a valid twitter or Facebook account can register on this web-application. This project provides a single interface to handle multiple accounts of different social networking websites. It allows the user to post on multiple Facebook pages and multiple Twitter profile at a same time. This project also allows the user to manage their accounts, view statistics, receive feedback and view insights of their post across different social networking profiles. User can add multiple Twitter accounts and multiple Facebook pages, through which they can get connected with their audience. User can view number of likes, view comments and also reply to the messages. The user also has an option to deactivate their account on this application.

## CHAPTER 2

### SOFTWARE REQUIREMENT SPECIFICATION

**Purpose :** The main purpose of this project is to implement a tool to broadcast content across different social networks and also manage the pages on different accounts which includes viewing Posts, Messages, content engagement reports and reaching out to all subscribers of the pages on different sites through a client.

**Scope:** This application can be easily accessed by any one with an internet connection. It is used to connect to user's twitter and facebook profile. It helps the user to post the same content on multiple accounts and pages with just one click. User can view number of likes, view comments and also reply to the messages.

#### **Users of the system are:**

##### **A. Business Organizations**

- They can manage and access all their pages across different sites.
- They can broadcast content onto selected pages across different sites simultaneously.
- They can directly interact with page subscribers/customers when they raise an issue on one of the social media channels
- They can manage and schedule content of their pages.

#### **User Requirements :**

- Username and password for the website for the purpose of registering their accounts on multiple social networking sites.
- Access to Internet.

### **2.1 Requirements Specification Document**

#### **2.1.1. Functional Requirements**

- User should easily be able access this web-application on the internet with a predefined URL.
- User should be able to register with this application by authenticating either their Twitter or Facebook profile.
- User should be able to post text and URL, on different pages/accounts associated with social sites.
- User should be able to direct communicate with any of their subscriber via messages.
- User should be able to view insights of each post on their page i.e view likes and also comments .
- Used to check the delays errors inconsistencies in records and access to historical records.

### **2.1.2. Non-Functional Requirements:**

- All modules must be clearly separate to allow different user interfaces to be developed in future. Through thoughtful and effective software engineering, all steps of software development process will be well documented to ensure maintainability of the product throughout its lifetime. All developments will be provided with good documentation.
- The response time, utilization and throughput of the system must be satisfactory, to give good performance. Care is taken so as to ensure a system with comparatively high performance.
- The ease of the use and training of the end users of the system is usability. System should have qualities like efficiency, control. The main aim of the project is to provide comfort to the chefs and cooks by updating various recipes and easy search options.
- The ease with which a software system can accommodate changes to its software is modifiability. Our project is easily adaptable for changes that is useful for the application to withstand the needs of the users.
- The ability of the system to run under different computing environments is called portability. The environment types can be either hardware or software, but is usually a combination of two.
- The extent to which, the existing application can be re-used in a new application. Our application can be reused a number of times without any technical difficulties.

## **2.2 Software Requirements**

- Language: JAVA
- Front End: J2EE
- Framework: Play Framework
- Database: Hbase
- Testing: JUnit

## **2.3 Hardware Requirements**

- Pentium 4 Processor
- Minimum 1GB RAM
- 80GB HDD
- Internet Connectivity
- Operating system- Any Operating System

## **CHAPTER 3**

### **TECHNOLOGIES/TOOLS USED FOR IMPLEMENTATION**

This section includes all the technologies and tools used for developing the application Social Media Broadcast Client.

#### **3.1 JAVA**

Java is an object-oriented programming language developed by Sun Microsystems a company best known for its high end UNIX workstations. Java language was designed to be small, simple and portable across platforms, operating systems, both at the source and at the binary level, which means that Java programs (applet and application) can run on any machine that has the Java virtual machine (JVM) installed.

#### **3.2 PLAY FRAMEWORK**

Play is an open source web application framework, written in Scala and Java, which follows the model–view–controller (MVC) architectural pattern. It aims to optimize developer productivity by using convention over configuration, hot code reloading and display of errors in the browser.

Support for the Scala programming language has been available since version 1.1 of the framework. In version 2.0, the framework core was rewritten in Scala. Build and deployment was migrated to SBT, and templates use Scala instead of Groovy.

Major differences from other framework

From other Java frameworks:

- Stateless: Play 2 is fully RESTful - there is no Java EE session per connection.
- Integrated unit testing: JUnit and Selenium support is included in the core.
- API comes with most required elements built-in.
- Static methods: all controller entry points are declared as static (or equivalently, in Scala, functions on Scala objects). After requests were made for this to be customisable, Play 2.1 now supports other styles of controllers, so controllers need not be static/Scala objects; however, this is still the default.
- Asynchronous I/O: due to using JBoss Netty as its web server, Play can service long requests asynchronously rather than tying up HTTP threads doing business logic like Java EE frameworks that don't use the asynchronous support offered by Servlet 3.0.<sup>[16]</sup>
- Modular architecture: like Rails and Django, Play comes with the concept of modules.

### 3.3 HBASE

HBase is an open source, non-relational, distributed database modeled after Google's BigTable and written in Java. It is developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of HDFS (Hadoop Distributed Filesystem), providing BigTable-like capabilities for Hadoop. That is, it provides a fault-tolerant way of storing large quantities of sparse data (small amounts of information caught within a large collection of empty or unimportant data, such as finding the 50 largest items in a group of 2 billion records, or finding the non-zero items representing less than 0.1% of a huge collection).

HBase features compression, in-memory operation, and Bloom filters on a per-column basis as outlined in the original BigTable paper. Tables in HBase can serve as the input and output for MapReduce jobs run in Hadoop, and may be accessed through the Java API but also through REST, Avro or Thrift gateway APIs.

### 3.4 JUNIT

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit.

### 3.5 Eclipse IDE:

Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications. By means of various plug-ins, Eclipse may also be used to develop applications in other programming languages: Ada, ABAP, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Perl, PHP, Python, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

### 3.6 HTML:

HTML5 is a mark-up language used for structuring and presenting content for the World Wide Web and a core technology of the Internet.

### 3.7 JSON:

**JavaScript Object Notation**, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML.

Although originally derived from the JavaScript scripting language, JSON is a language-independent data format, and code for parsing and generating JSON data is readily available in a large variety of programming languages.

The JSON format was originally specified by Douglas Crockford. It is currently described by two competing standards, RFC 7159 and ECMA-404. The ECMA standard is minimal describing only the allowed grammar syntax, whereas the RFC also provides some semantic and security considerations. The official Internet media type for JSON is application/json. The JSON file name extension is .json.

## CHAPTER 4 DESIGN

### 4.1 System Architecture

We have used Model,View and Controller architecture for this application. The Diagram follows below

#### 4.1.1 Architecture Diagram

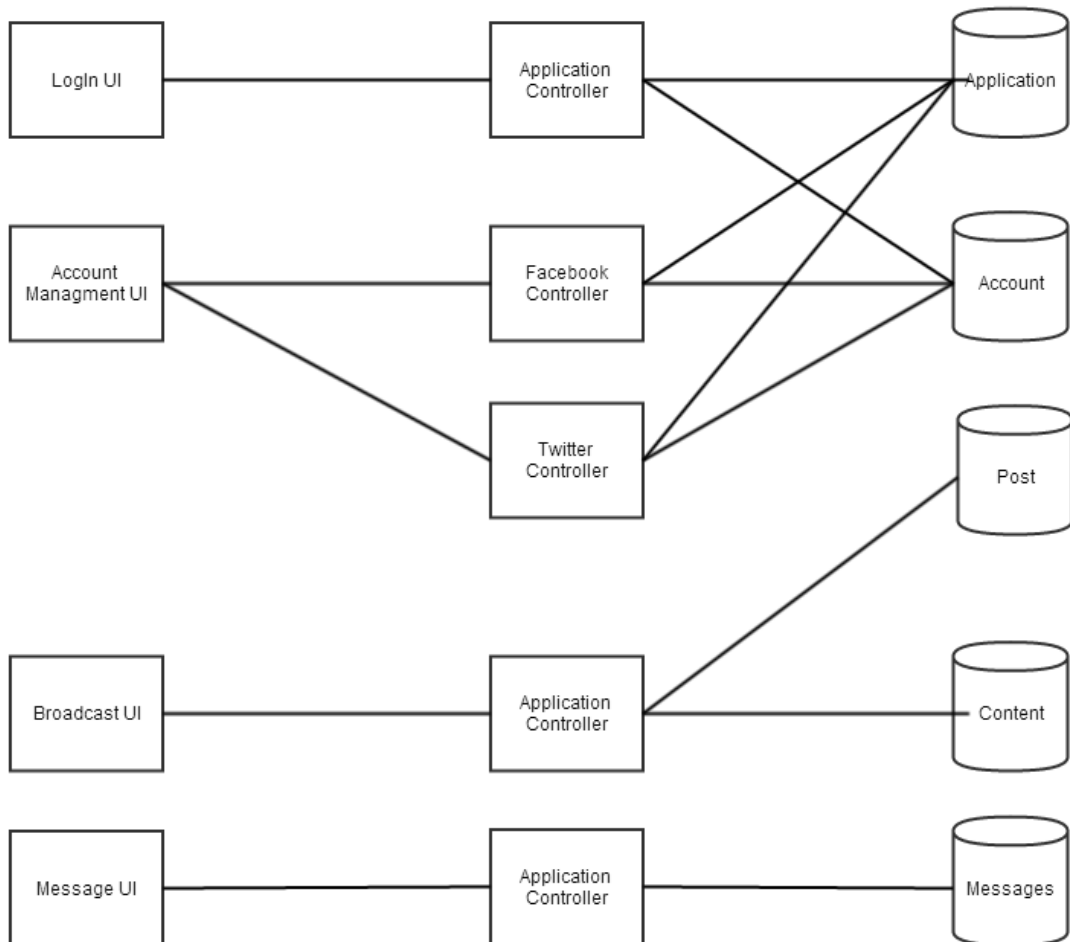


Fig. No. 4.1 Architecture Diagram

#### 4.1.2 Modules of the System

Different modules are provided in the application to ensure the friendly user interaction. Social Media Broadcast Client Application includes 7 modules. They are as follows

#### MODULE 1: FIRST SCREEN

The first screen is the very first screen that is displayed to the user on opening the application. This screen comprises of the sections mentioned below.

- Sign up with Facebook: On clicking this button, the user will be redirected to the Facebook login page if he is not already logged in. The access permissions are taken from the user for allowing access to the Facebook account.
- Sign up with Twitter: On clicking this button, the user will be redirected to the Twitter login page if he is not already logged in. The access permissions are taken from the user for allowing access to the Twitter account.

## **MODULE 2: REGISTRATION SCREEN**

The registration screen is the screen that is displayed to the user on logging in to the application for the very first time. This screen comprises of the elements mentioned below.

- AppID: This is the unique key through which the user can be authenticated the next time he/she uses the application.
- Register: On clicking this button, the user sends request to confirm his AppID.

## **MODULE 3: HOME SCREEN**

The Home screen is the first screen that is displayed to the user after registering with the application. This screen comprises of the sections mentioned below.

- Home: Home button redirects to the Home page which is available throughout the application on the navigation bar.
- Account: Account button links to another page where the management of the application accounts is done. It is available on the navigation bar.
- Messages: Message button is linked to the messages page where the incoming messages from different accounts and pages on Twitter and Facebook are displayed on single interface and respective actions are performed.
- Broadcast: This button on the menu enables the user to utilize the main feature of the application, i.e., to broadcast the same content to different networking sites with single click.
- Logout: Logout button enables the user to exit the application.
- Recent Posts: Recent posts indicate the users recent contents uploaded through the application containing a Content ID.
- Show Reach: Content reach on different sites is examined by the user by clicking on the 'show reach' button allotted for each posted content.

## **MODULE 4: BROADCAST SCREEN**

The broadcast screen is displayed to the user on clicking the broadcast button on the header in the home page. Also the broadcast module is fixed in the left division in various screens for easy access of broadcasting feature in the web application. This screen comprises of the fields mentioned below.

- Description: Description field is provided to the user to describe the content to be posted to different pages and accounts. This will also be used to identify the post by searching its name on respective websites.
- URL: URL field is provided for the link that the user needs to post to different accounts and

- pages.
- **Post To:** In this section the application will fetch the pages and accounts registered with the AppID and provides a list of all the accounts and pages on Facebook and Twitter respectively. It is provided with check boxes so that the ticked ones are selected and the content is posted on the respective addresses.

## **MODULE 5: REACH SCREEN**

The reach page is opened as an extension to the “Show Reach” button on the home page displaying recent posts. The details of the selected content are expanded enabling the user to compare the reach of posted content on Facebook and Twitter media respectively. This page comprises of the elements mentioned below.

- **Page Name:**The Facebook page name on which the content has been posted is displayed here.
- **Shares:** No. Of shares through the respective page is displayed in this field.
- **Likes:** The count of the likes on respective page is displayed here.
- **Comments:** No. Of comments on the respective page is displayed here.
- **Account Name:**The Twitter account name on which the content has been posted is displayed here.
- **Favorites:** On respective Twitter account, no of users that have marked the post as favorite is displayed here.
- **Retweets:** No. Of Retweets is displayed in this field for respective account on twitter.
- **Replies:**The count of the no. Of replies the post has received on the respective account is displayed in this field.

## **MODULE 6: MESSAGES SCREEN**

The messages screen is displayed to the user on clicking the messages button in the home page. This screen displays the incoming messages from the registered pages and accounts of Facebook and Twitter respectively.

- **Sender Name:** The name of the user on Facebook /Twitter is displayed in this field.
- **Account/Page Name:**The name of the account/page through which he/she has sent the message is displayed in this field.
- **Date and Time:** The Date and time at which the message was received is displayed in this field.
- **Reply:** The application user is provided an option to reply by using reply button and is provided a text area for typing the message.

## **MODULE 7: ACCOUNTS SCREEN**

The accounts screen is displayed to the user on clicking the accounts button on the home page header. It enables the users to manage their Facebook pages and Twitter accounts in this application.

- **AppID:** The unique ID through which the user is identified on the application is displayed here.
- **Page Name:**The list of Facebook page names that have been retrieved from Facebook account of the user is displayed in this field.

- Subscriber count: The count of the subscribers on the respective pages is displayed here.
- +Facebook: The application user is provided an option to add one Facebook account if he/she had no Facebook account registered for even once. On clicking this button the user is connected to the Facebook login page.
- Account Name: The list of Twitter account names that have been retrieved from Facebook account of the user is displayed in this field.
- Followers count: The count of the followers on the respective accounts is displayed here.
- +Twitter: The Button is provided to Add another Twitter account to the Application database.

## 4.2 UML Diagrams

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

So UML can be described as a general purpose visual modelling language to visualize, specify, construct and document software system. Although UML is generally used to model software systems but it is not limited within this boundary. It is also used to model non software systems as well like process flow in a manufacturing unit etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization UML is become an OMG (Object Management Group) standard.

### Object oriented concepts:

UML can be described as the successor of object oriented analysis and design. An object contains both data and methods that control the data. The data represents the state of the object. A class describes an object and they also form hierarchy to model real world system. The hierarchy is represented as inheritance and the classes can also be associated in different manners as per the requirement.

The objects are the real world entities that exist around us and the basic concepts like abstraction, encapsulation, inheritance, polymorphism all can be represented using UML.

So UML is powerful enough to represent all the concepts exists in object oriented analysis and design. UML diagrams are representation of object oriented concepts only.

### Role of UML in OO design:

UML is a modelling language used to model software and non-software systems. Although UML is used for non-software systems the emphasis is on modelling object oriented software applications. The input from the OO analysis and design is the input to the UML diagrams.

The building blocks of UML can be defined as:

- Things
- Relationships

- Diagrams

## Things:

Things are the most important building blocks of UML. Things can be:

- Structural
- Behavioral
- Grouping
- Annotational

### Structural things:

The Structural things define the static part of the model. They represent physical and conceptual elements. Following are the brief descriptions of the structural things.

#### 4.2.1 Class Diagram

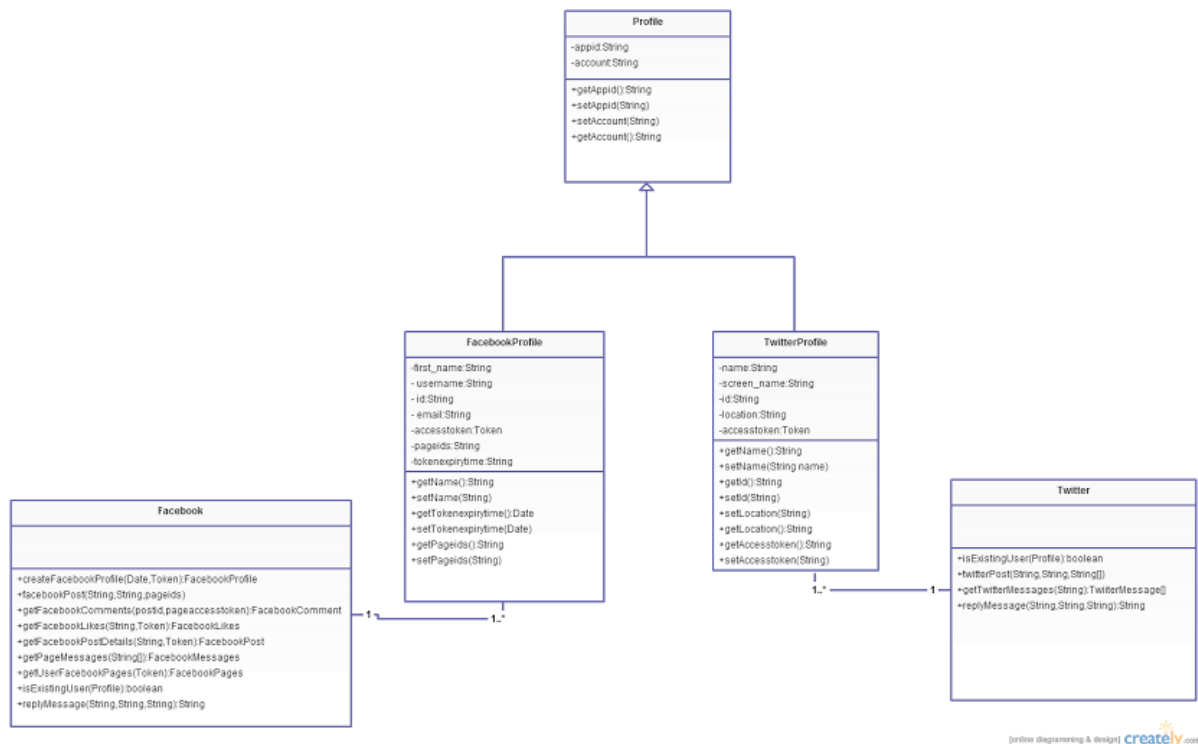
The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a *structural diagram*.

#### Purpose:

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.



**Fig. No. 4.2 The class diagram of the “Social Media Broadcast Client” web-application application.**

### Object diagrams:

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

Object diagrams are used to render a set of objects and their relationships as an instance.

### Purpose:

- Forward and reverse engineering.
- Object relationships of a system
- Static view of an interaction.
- Understand object behaviour and their relationship from practical perspective.

### Component diagrams:

Component diagrams are different in terms of nature and behaviour. Component diagrams are used to model physical aspects of a system.

Now the question is what are these physical aspects? Physical aspects are the elements like executable, libraries, files, documents etc. which resides in a node.

So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

**Purpose:**

- Visualize the components of a system.
- Construct executable by using forward and reverse engineering.
- Describe the organization and relationships of the components.

**Deployment diagrams:**

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

**Purpose:**

- Visualize hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe runtime processing nodes.

#### 4.2.2 Use case Diagram

To model a system the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, *dynamic behaviour* means the behaviour of the system when it is running /operating.

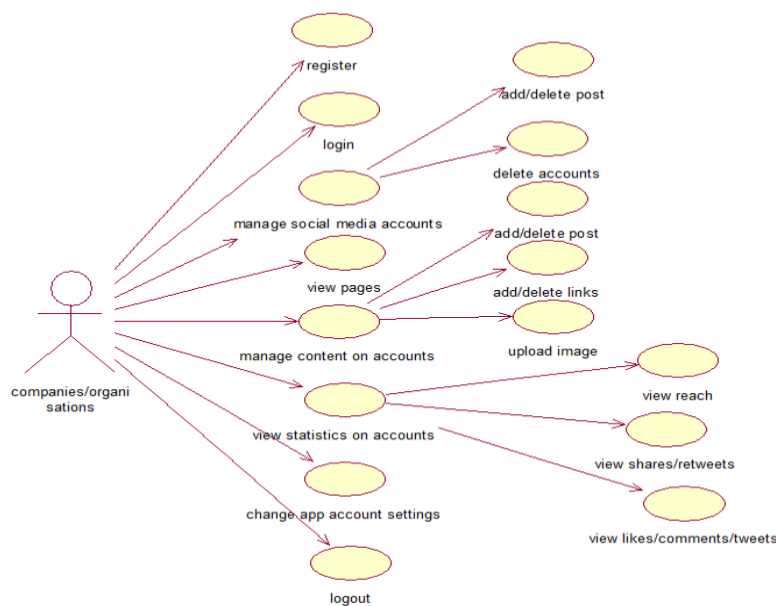
So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

So to model the entire system numbers of use case diagrams are used.

**Purpose:**

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.



**Fig. No. 4.3 Use Case Diagram**

### Interaction diagrams:

From the name *Interaction* it is clear that the diagram is used to describe some type of interactions among the different elements in the model. So this interaction is a part of dynamic behaviour of the system.

This interactive behaviour is represented in UML by two diagrams known as *Sequence diagram* and *Collaboration diagram*. The basic purposes of both the diagrams are similar.

Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

### Purpose:

- To capture dynamic behaviour of a system.
- To describe the message flow in the system.
- To describe structural organization of the objects.
- To describe interaction among objects.
- The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

## State chart diagram

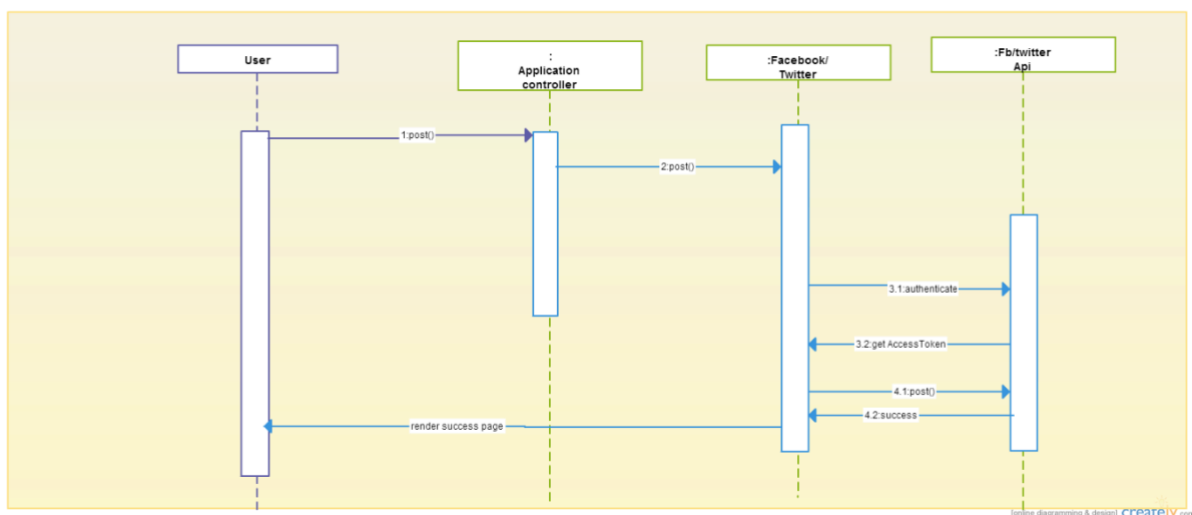
A Statechart diagram describes a state machine. Now to clarify it state machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Activity diagram explained in next chapter is a special kind of a Statechart diagram. As Statechart diagram defines states it is used to model lifetime of an object.

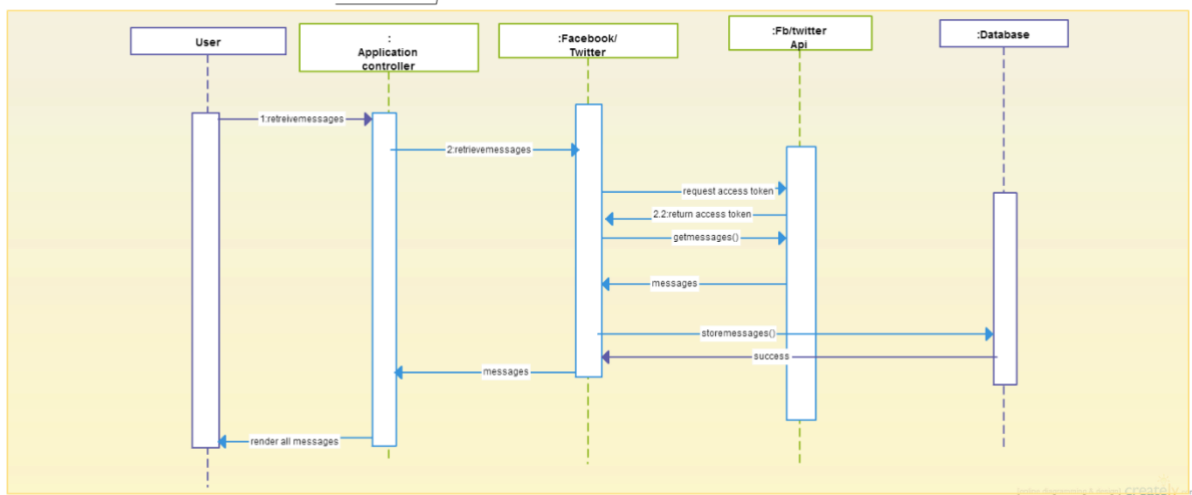
### Purpose:

- To model dynamic aspect of a system.
- To model life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model states of an object.

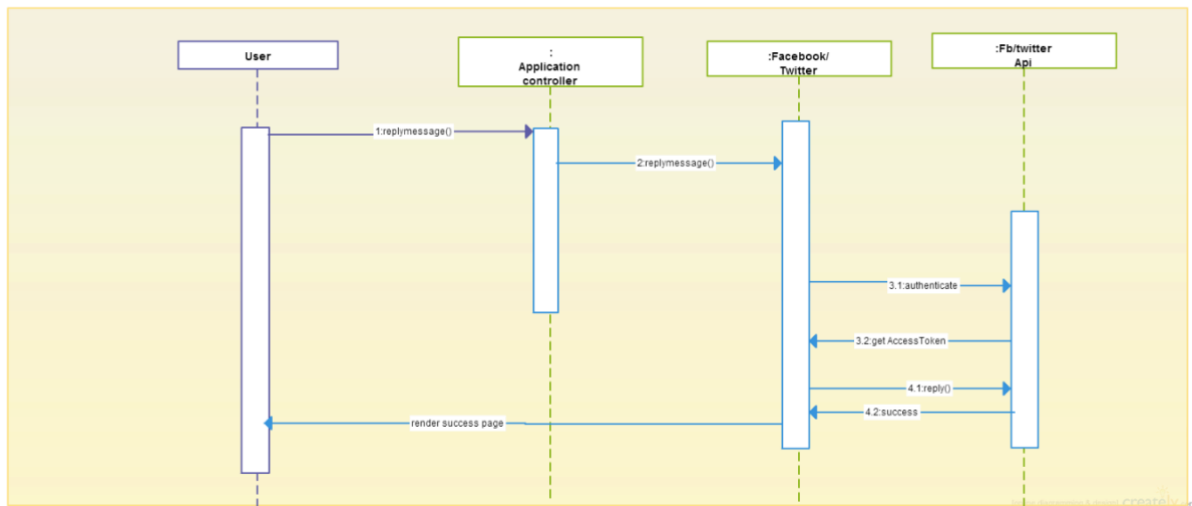
### 4.2.3 Sequence Diagram:



**Fig. No. 4.4 Sequence Diagram for posting**



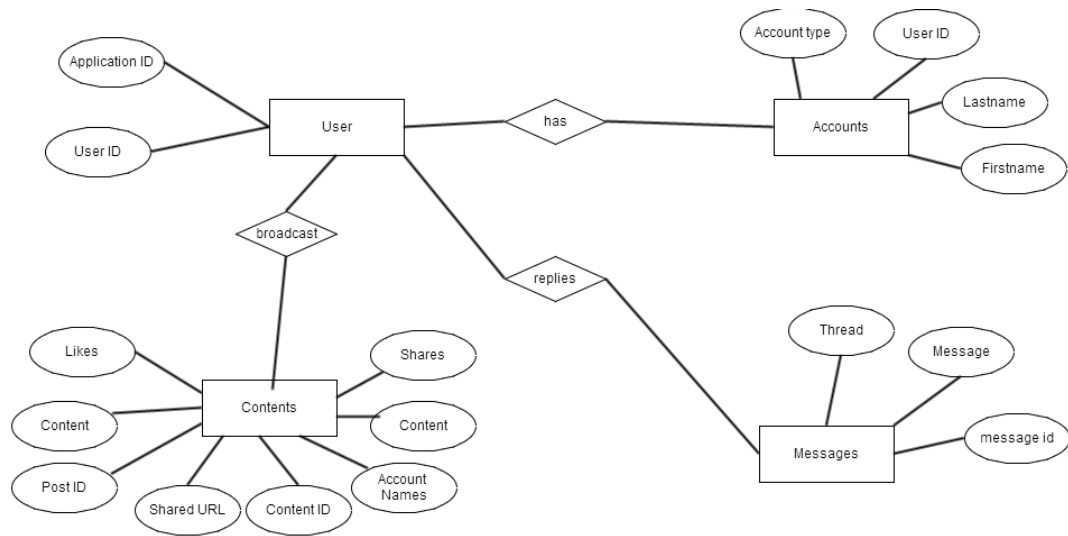
**Fig. No. 4.5 Sequence Diagram for Retrieving All Messages**



**Fig. No. 4.6 Sequence Diagram for Replying to a Message**

## 4.3 Database Design

### 4.3.1 ER Diagrams



**Fig. No. 4.7 ER Diagram**

# CHAPTER 5

## IMPLEMENTATION

### 5.1 Classes Used

#### **Facebook:**

This class implements all the functionality related to Facebook.

- Responsible for creating a Facebook profile.
- Retrieving all the pages of a particular user.
- Retrieving all the messages of all the pages of the user.
- Replying to a particular message.
- Posting a status on particular set of pages.
- Retrieving all the comments on a particular post.
- Retrieving all the likes on a particular post.

#### **Twitter:**

This class implements all the functionality related to twitter.

- Responsible of creating a Twitter profile.
- Posting on multiple twitter profiles.
- Checking whether the user is a existing user or not.
- Retrieving all the messages sent to a particular user.
- Replying to a particular message.

## CHAPTER 6

### Integration and Testing

#### 6.1. Integration:

##### What is integration testing?

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

##### Definition by ISTQB

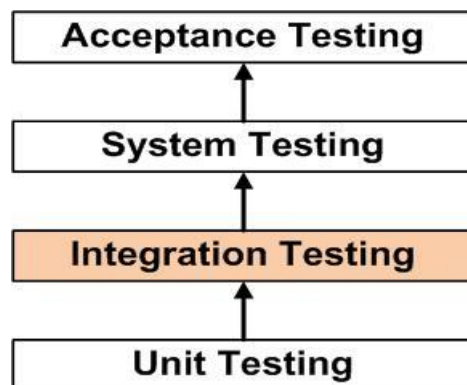
- **Integration testing:** Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.
- **Component integration testing:** Testing performed to expose defects in the interfaces and interaction between integrated components.
- **System integration testing:** Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

##### Why Integration testing is required?

Integration finds the bugs, that occur when two or more models are integrated .Main purpose of integration testing is to identify the functional, requirement and performance level bugs. When they are integrated, functional, requirement and performance related issues will occur due to the integration. Therefore the purpose of this level of testing is to expose faults in the interaction between integrated units.

##### When is Integration Testing performed?

Integration Testing is performed after Unit Testing and before System Testing.



### **Who performs Integration Testing?**

Either Developers themselves or independent Testers perform Integration Testing.

### **Approaches in Integration testing:**

There are four approaches used in Integration testing.

- Big Bang Testing
- Top-Down Testing
- Bottom-Up Testing
- Sandwich Testing

#### **Big Bang Testing:**

In this approach, all or most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. The Big Bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

#### **Top-Down Testing:**

In this approach, the highest-level modules are tested first and progressively lower-level modules are tested after that.

#### **Bottom-Up Testing:**

Bottom-up integration testing begins with unit testing followed by higher level combinations of units called modules. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

#### **Sandwich Testing:**

This is an approach to combine top down testing with bottom up testing.

From above mentioned approaches only Top-Down & Bottom-Up are used mostly. So the main advantage of bottom-up approach is that bugs are more easily found as compared with top-down approach, it is easier to find the missing branch link and show the whole software process to user early.

### **Steps to perform integration testing:**

**Step 1:** Create a Test Plan

**Step 2:** Create Test Cases and Test Data

**Step 3:** Once the components have been integrated execute the test cases

**Step 4:** Fix the bugs if any and re test the code

**Step 5:** Repeat the test cycle until the components have been successfully integrated.

### **Essentials of effective integration testing:**

There are various factors that affect Software Integration and hence Integration Testing:

### **Software Configuration Management:**

Since Integration Testing focuses on Integration of components and components can be built by different developers and even different development teams, it is important the right versions of components are tested. This may sound very basic, but the biggest problem faced in n-tier development is integrating the right version of components. Integration testing may run through several iterations and to fix bugs components may undergo changes. Hence it is important that a good Software Configuration Management (SCM) policy is in place. We should be able to track the components and their versions. So each time we integrate the application components we know exactly what versions go into the build process.

### **Automate Build Process where necessary:**

A Lot of errors occurs because the wrong versions of components were sent for the build or there are missing components. If possible write a script to integrate and deploy the components this helps reduce manual errors.

### **Document:**

Document the Integration process/build process to help eliminate the errors of omission or oversight. It is possible that the person responsible for integrating the components forgets to run a required script and the Integration Testing will not yield correct results.

### **Defect Tracking:**

Integration Testing will lose its edge if the defects are not tracked correctly. Each defect should be documented and tracked. Information should be captured as to how the defect was fixed. This is valuable information. It can help in future integration and deployment processes.

### **Use of stubs and drivers in integration testing:**

While doing an Integration testing, if we don't have all the modules ready and need to test a particular module which is ready then we use Stubs and Drivers.

Stubs and drivers are used for a Top down Integration testing and Bottom up Integration Testing.

### **For e.g.:**

If we have Modules X, Y and Z. X module is ready and we need to test it, but it calls functions from y and z (which are not yet ready). To test at such a module, we write a small dummy piece

a code which Simulates Y and Z and which will return values for X. This piece of dummy code is called Stub in a Top down Integration Testing.

So Stubs are called Functions in Top down Integration.

Similar to the above: If we have Y and Z modules ready and x module is not ready, and we need to test Y and Z modules which accepts values from X, So to get the values from X We write a small piece of dummy code for X which returns values for Y and Z. So this piece of code is called Driver in Bottom up Integration testing.

### **Difference between system integration testing and integration testing:**

System integration testing takes multiple integrated systems that have passed system testing as input and tests their required interactions. Following this process, the deliverable systems are passed on to acceptance testing.

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

### **Individual Steps and Test Description:**

For each step of the integration process identified above, describe the type of tests that will be used to verify that the elements integrated in this step perform as expected. Describe in general the expected results of the test set. At the lower levels, these tests will focus on direct testing of interfaces between software functions and modules, between software functions or modules and specific electronics, etc. As more of the system is put together, tests will continue to be concerned with verifying interfaces, but will include more of a “functional test” Flavor. At each step, specify where re-running of tests from previous steps may be required to verify that integrating this new element has not caused previously tested functions to fail.

- Software Integration Test Description
- Hardware Integration Test Description
- Hardware/Software Integration Test Description
- Subsystem Integration Test Description
- Final Functional Tests

Final Functional Tests describe the set of functional tests to be run at the end of integration to verify the basic functionality of the system.

This test is intended to confirm that the system has been successfully integrated and is now ready for full System Test by Engineering.

### **Tools and Test Equipment Required:**

Identify all tools and test equipment needed to accomplish the integration. Examples are computer workstations, oscilloscopes, meters, host operating systems, etc. Specify revision levels or version numbers where necessary. The integration protocols will specify in detail all configuration information, set-up procedures, parameters, environmental conditions (lighting, humidity, etc.), etc. necessary for the integration testing. In this plan summarize any significant configuration efforts, special environmental requirements, etc.

### **Program Stubs and Test Data Required:**

Based on the testing strategy and test design, identify any program stubs or special test data required for each integration step.

### **Responsibilities and Schedule:**

Identify all personnel skill types and quantities, define the responsibilities assigned to each and develop a schedule which shows the sequence of major test set execution and the amount of time estimated for that test set. Identify known risks and any assumptions. NOTE: If any of this information is included in the project plan instead, reference the section in that document.

- Roles and Responsibilities
- Key Dependencies
- Risks and Assumptions
- Schedule

### **Problem Recording and Resolution:**

Define the mechanism to be used for problem recording and resolution, including any necessary rework of documents, software or hardware elements, test plan or procedures. Include (or reference) any forms. Reference any standard problem tracking procedures that will be used.

### **Rework, Review, and Retest Procedures:**

Define the process for rework, review, and retest of any element needing modification. This process must include sufficient retest to verify that any modifications have not impacted other functions already tested. This process must also define the project guidelines for how related design and requirements documents will be updated as changes are made.

### **Suspension, Restart, and Exit Criteria:**

Define criteria for the following

- Suspending testing before completion (for instance, the discovery of major problems with a certain feature,
- Criteria for restarting testing following such a suspension.
- The criteria for determining that integration test is complete and system test can begin.

### **Conclusion:**

Integration testing is a phase of testing. Once Modules are developed and tested, all components are grouped together to check the functionality of System as a whole. All modules are coupled together to check the functionality i.e. know whether system breakdown at some point, whether data is moving properly from one module to dependent module, etc. In Organizations there is a separated Integration testing team who test the system as whole. If done properly it can reduce the risk of defect leaking, also it can reduce the risk of defects to be caught by client during UAT phase. Thus, integration plays an important role in testing cycle.

## **6.2 Testing:**

A primary purpose for testing is to detect software failures so that defects may be uncovered and corrected. This is a non-trivial pursuit.

Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions.

The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do.

In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

### **Defects and failures:**

Not all software defects are caused by coding errors. One common source of expensive defects is caused by requirements gaps, e.g., unrecognized requirements that result in errors of omission by the program designer. A common source of requirements gaps is non-functional requirements such as testability, scalability, maintainability, usability, performance, and security.

Software faults occur through the following process. A programmer makes an error (mistake), which results in a defect (fault, bug) in the software source code. If this defect is executed, in certain situations the system will produce wrong results, causing a failure. Not all defects will necessarily result in failures. For example, defects in dead code will never result in failures. A defect can turn into a failure when the environment is changed.

Examples of these changes in environment include the software being run on a new hardware platform, alterations in source data or interacting with different software. A single defect may result in a wide range of failure symptoms.

### **Compatibility:**

A frequent cause of software failure is compatibility with another application or new operating system (or, increasingly web browser version). In the case of lack of backward compatibility this can occur because the programmers have only considered coding the programs for, or

testing the software, on the latest operating system they have access to or else, in isolation (no other conflicting applications running at the same time) or under 'ideal' conditions ('unlimited' memory; 'superfast' processor; latest operating system incorporating all updates, etc).

In effect, everything is running "as intended" but only when executing at the same time on the same machine with the particular combination of software and/or hardware. These are some of the hardest failures to predict, detect and test for and many are therefore discovered only after release into the larger world with its largely unknown mix of applications, software and hardware. It is likely that an experienced programmer will have had exposure to these factors through "co-evolution" with several older systems and be much more aware of potential future compatibility problems and therefore tend to use tried and tested functions or instructions rather than always the latest available which may not be fully compatible with earlier mixtures of software/hardware. This could be considered a prevention oriented strategy that fits well with the latest testing phase suggested by Dave Gelperin and William C. Hetzel cited below.

### **Input combinations and preconditions:**

A problem with software testing is that testing under all combinations of inputs and preconditions (initial state) is not feasible, even with a simple product. This means that the number of defects in a software product can be very large and defects that occur infrequently are difficult to find in testing.

More significantly, non-functional dimensions of quality (how it is supposed to *be* versus what it is supposed to *do*) -- for example, usability, scalability, performance, compatibility, reliability -- can be highly subjective; something that constitutes sufficient value to one person may be intolerable to another.

### **Static vs. dynamic testing:**

There are many approaches to software testing. Reviews, walkthroughs or inspections are considered as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. The former can be, and unfortunately in practice often is, omitted, whereas the latter takes place when programs begin to be used for the first time - which is normally considered the beginning of the testing stage. This may actually begin before the program is 100% complete in order to test particular sections of code (modules or discrete functions). For example, Spreadsheet programs are, by their very nature, tested to a large extent "on the fly" during the build process as the result of some calculation or text manipulation is shown interactively immediately after each formula is entered.

### **Software verification and validation:**

Software testing is used in association with verification and validation:

- Verification: Have we built the software right (i.e., does it match the specification?)? It is process based.
- Validation: Have we built the right software (i.e., is this what the customer wants?)? It is product based.

### **The software testing team:**

Software testing can be done by software testers. Until the 1950s the term "software tester" was used generally, but later it was also seen as a separate profession. Regarding the periods and the different goals in software testing there have been established different roles: test lead/manager, test designer, tester, test automation developer, and test administrator.

### **Software Quality Assurance (SQA):**

Though controversial, software testing may be viewed as an important part of the software quality assurance (SQA) process. In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in defect rate.

What constitutes an acceptable defect rate depends on the nature of the software. An arcade video game designed to simulate flying an airplane would presumably have a much higher tolerance for defects than mission critical software such as that used to control the functions of an airliner. Although there are close links with SQA, testing departments often exist independently, and there may be no SQA function in some companies.

Software Testing is a task intended to detect defects in software by contrasting a computer program's expected results with its actual results for a given set of inputs. By contrast, QA is the implementation of policies and procedures intended to prevent defects from occurring in the first .

### **Testing methods:**

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

#### **Black box testing:**

Black box testing treats the software as a black box without any knowledge of internal implementation. Black box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

#### **Specification-based testing:**

Specification-based testing aims to test the functionality according to the requirements. Thus, the tester inputs data and only sees the output from the test object. This level of testing usually requires thorough test cases to be provided to the tester who then can simply verify that for a given input, the output value (or behaviour), is the same as the expected value specified in the test case.

Specification-based testing is necessary but insufficient to guard against certain risks.

#### **Advantages and disadvantages:**

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code MUST have bugs. Using the principle, "Ask and you shall receive," black box testers find

bugs where programmers don't. BUT, on the other hand, black box testing is like a walk in a dark labyrinth without a flashlight, because the tester doesn't know how the back end was actually constructed. That's why there are situations when 1. A black box tester writes many test cases to check something that can be tested by only one test case and/or 2. Some parts of the back end are not tested at all

Therefore, black box testing has the advantage of an unaffiliated opinion on the one hand and the disadvantage of blind exploring on the other.

### **White box testing:**

White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these)

The following types of white box testing exist:

- API testing - Testing of the application using Public and Private APIs.
- Code coverage - creating tests to satisfy some criteria of code coverage. For example, the test designer can create tests to cause all statements in the program to be executed at least once.
- Fault injection methods.
- Mutation testing methods.
- Static testing - White box testing includes all static testing.

### **Code completeness evaluation:**

White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.

Two common forms of code coverage are: *Function coverage*, which reports on functions executed and *statement coverage*, which reports on the number of lines executed to complete the test. They both return coverage metric, measured as a percentage.

### **Grey Box Testing:**

In recent years the term Grey box testing has come into common usage. This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level.

Manipulating input data and formatting output do not qualify as Grey-box because the input and output are clearly outside of the black-box we are calling the software under test. This is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.

Grey box testing may also include reverse engineering to determine, for instance, boundary values or error messages.

### **Acceptance testing:**

Acceptance testing can mean one of two things:

- A smoke test is used as an acceptance test prior to introducing a build to the main testing process.
- Acceptance testing performed by the customer is known as user acceptance testing (UAT).

### **Regression Testing:**

Regression testing is any type of software testing that seeks to uncover software regressions. Such regressions occur whenever software functionality that was previously working correctly stops working as intended.

Typically regressions occur as an unintended consequence of program changes. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged.

### **Non Functional Software Testing:**

Special methods exist to test non-functional aspects of software.

- Performance testing checks to see if the software can handle large quantities of data or users. This is generally referred to as software scalability. This activity of Non Functional Software Testing is often times referred to as Load Testing.
- Usability testing is needed to check if the user interface is easy to use and understand.
- Security testing is essential for software which processes confidential data and to prevent system intrusion by hackers.
- Internationalization and localization is needed to test these aspects of software, for which a pseudo localization method can be used.

In contrast to functional testing, which establishes the correct operation of the software (correct in that it matches the expected behavior defined in the design requirements), non-functional testing verifies that the software functions properly even when it receives invalid or unexpected inputs. Software fault injection, in the form of fuzzing is an example of non-functional testing.

Non-functional testing, especially for software, is designed to establish whether the device under test can tolerate invalid or unexpected inputs, thereby establishing the robustness of input validation routines as well as error-handling routines.

Various commercial non-functional testing tools are linked from the Software fault injection page; there are also numerous open-source and free software tools available that perform non-functional testing.

## Testing process:

A common practice of software testing is performed by an independent group of testers after the functionality is developed before it is shipped to the customer. This practice often results in the testing phase being used as project buffer to compensate for project delays, thereby compromising the time devoted to testing. Another practice is to start software testing at the same moment the project starts and it is a continuous process until the project finishes.

In counterpoint, some emerging software disciplines such as extreme programming and the agile software development movement, adhere to a "test-driven software development" model. In this process unit tests are written first, by the software engineers (often with pair programming in the extreme programming methodology). Of course these tests fail initially; as they are expected to. Then as code is written it passes incrementally larger portions of the test suites. The test suites are continuously updated as new failure conditions and corner cases are discovered, and they are integrated with any regression tests that are developed. Unit tests are maintained along with the rest of the software source code and generally integrated into the build process (with inherently interactive tests being relegated to a partially manual build acceptance process).

Testing can be done on the following levels:

- Unit testing tests the minimal software component, or module. Each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.
- Integration testing exposes defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.
- System testing tests a completely integrated system to verify that it meets its requirements.
- System integration testing verifies that a system is integrated to any external or third party systems defined in the system requirements.

## Before shipping the final version of software, alpha and beta testing are often done additionally:

- Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.
- Beta testing comes after alpha testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

Finally, acceptance testing can be conducted by the end-user, customer, or client to validate whether or not to accept the product. Acceptance testing may be performed as part of the hand-off process between any two phases of development.

## **Regression testing:**

After modifying software, either for a change in functionality or to fix defects, a regression test re-runs previously passing tests on the modified software to ensure that the modifications haven't unintentionally caused a regression of previous functionality. Regression testing can be performed at any or all of the above test levels. These regression tests are often automated.

More specific forms of regression testing are known as sanity testing, when quickly checking for bizarre behaviour, and smoke testing when testing for basic functionality.

Benchmarks may be employed during regression testing to ensure that the performance of the newly modified software will be at least as acceptable as the earlier version or, in the case of code optimization, that some real improvement has been achieved

## **Testing Tools:**

Program testing and fault detection can be aided significantly by testing tools and debuggers. Types of testing/debug tools include features such as:

- Formatted dump or Symbolic debugging, tools allowing inspection of program variables on error or at chosen points.
- Benchmarks, allowing run-time performance comparisons to be made.
- Performance analysis, or profiling tools that can help to highlight hot spots and resource usage

## **Measuring software testing:**

Usually, quality is constrained to such topics as correctness, completeness, security, but can also include more technical requirements as described under the ISO standard ISO 9126, such as capability, reliability, efficiency, portability, maintainability, compatibility, and usability.

There are a number of common software measures, often called "metrics", which are used to measure the state of the software or the adequacy of the testing.

## **Testing artifacts:**

Software testing process can produce several artifacts.

## **Test case:**

A test case in software engineering normally consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output, expected result, and actual result.

Clinically defined, a test case is an input and an expected result. This can be as pragmatic as 'for condition x your derived result is y', whereas other test cases described in more detail the input scenario and what results might be expected. It can occasionally be a series of steps (but often steps are contained in a separate test procedure that can be exercised against multiple test cases, as a matter of economy) but with one expected result or expected outcome. The optional fields are a test case ID, test step or order of execution number, related requirement(s), depth,

test category, author, and check boxes for whether the test is automatable and has been automated. Larger test cases may also contain prerequisite states or steps, and descriptions.

A test case should also contain a place for the actual result. These steps can be stored in a word processor document, spreadsheet, database, or other common repository. In a database system, you may also be able to see past test results and who generated the results and the system configuration used to generate those results. These past results would usually be stored in a separate table.

### **Test script:**

The test script is the combination of a test case, test procedure, and test data. Initially the term was derived from the product of work created by automated regression test tools. Today, test scripts can be manual, automated, or a combination of both.

### **Test data:**

The most common test manually or in automation is retesting and regression testing. In most cases, multiple sets of values or data are used to test the same functionality of a particular feature. All the test values and changeable environmental components are collected in separate files and stored as test data. It is also useful to provide this data to the client and with the product or a project.

### **Test suite:**

The most common term for a collection of test cases is a test suite. The test suite often also contains more detailed instructions or goals for each collection of test cases. It definitely contains a section where the tester identifies the system configuration used during testing. A group of test cases may also contain prerequisite states or steps, and descriptions of the following tests.

### **Test plan:**

A test specification is called a test plan. The developers are well aware what test plans will be executed and this information is made available to the developers. This makes the developers more cautious when developing their code. This ensures that the developers code is not passed through any surprise test case or test plans.

### **Test harness:**

The software, tools, samples of data input and output, and configurations are all referred to collectively as a test harness.

### **Software testing certification types:**

Certifications can be grouped into exam-based and education-based.

- Exam-based certifications: For these there is the need to pass an exam, which can also be learned by self-study: e.g. for ISTQB or QAI.

- Education-based certifications: Education based software testing certifications are instructor-led sessions, where each course has to be passed, e.g. IIST (International Institute for Software Testing).

### **Testing certifications:**

- CATE offered by the International Institute for Software Testing
- CBTS offered by the Brazilian Certification of Software Testing (ALATS)
- Certified Software Tester (CSTE) offered by the Quality Assurance Institute (QAI)
- Certified Software Test Professional (CSTP) offered by the International Institute

### **Software Testing:**

- CSTP (TM) (Australian Version) offered by *K. J. Ross & Associates*
- ISEB offered by the Information Systems Examinations Board
- CBTS offered by the *Brazilian Certification of Software Testing* (ALATS)
- TMPF Next Foundation offered by the *Examination Institute for Information Science*.
- ISTQB Certified Tester, Foundation Level (CTFL) offered by the International Software Testing Qualification Board

### **Quality assurance certifications:**

- CSQE offered by the American Society for Quality
- CSQA offered by the *Quality Assurance Institute*
- CQIA offered by the American Society for Quality
- CMSQ offered by the *Quality Assurance Institute*

### **Controversy:**

Some of the major software testing controversies include:

### **What constitutes responsible software testing?**

Members of the "context-driven" school of testing believe that there are no "best practices" of testing, but rather that testing is a set of skills that allow the tester to select or invent testing practices to suit each unique situation.

### **Agile vs. traditional:**

Should testers learn to work under conditions of uncertainty and constant change or should they aim at process "maturity"? The agile testing movement has received growing popularity since 2006 mainly in commercial circles, whereas government and military software providers are slow to embrace this methodology, and mostly still hold to CMMI.

#### **Exploratory test vs. scripted:**

Should tests be designed at the same time as they are executed or should they be designed beforehand?

#### **Manual testing vs. automated:**

Some writers believe that test automation is so expensive relative to its value that it should be used sparingly. Others, such as advocates of agile development, recommend automating 100% of all tests. More in particular, test-driven development states that developers should write unit-tests of the x-unit type before coding the functionality. The tests then can be considered as a way to capture and implement the requirements.

#### **Software design vs. software implementation:**

Should testing be carried out only at the end or throughout the whole process?

#### **Who watches the watchmen?**

The idea is that any form of observation is also an interaction that the act of testing can also affect that which is being tested

### **6.2.1 Test cases:**

S.No	Test Cases	Expected Output	Actual Output	Results
1.	On load of home screen	Sign In With Twitter and Facebook	Show Sign In With Twitter and Facebook	Success

2.	On Signing In with Facebook	Allow access permission of Facebook	Open access permission page of Facebook.	Success
3.	On Signing In with Twitter	Allow access permission of Twitter.	Open access permission page of Twitter.	Success
4.	On clicking Broadcast button	Open Broadcast page	Broadcast screen displayed.	Success
5.	Broadcasting the content on selected pages/accounts clicking the Broadcast button.	Post content on the different pages and accounts.	Content posted on the selected pages/accounts.	Success
6.	On clicking account button.	Open the accounts page	Accounts page opened.	Success
7.	On clicking on the Register button.	Registration successful and AppID registered.	AppID registered.	Success
8.	On tapping “Reach”	Expand the details of the content..	Display the Facebook and Twitter reach	Success
9.	On clicking “messages”	Display messages page with all messages.	All messages displayed	Success
10.	On clicking “reply”	Message is sent to the sender of the respective account.	Message sent	Success

## CHAPTER 7

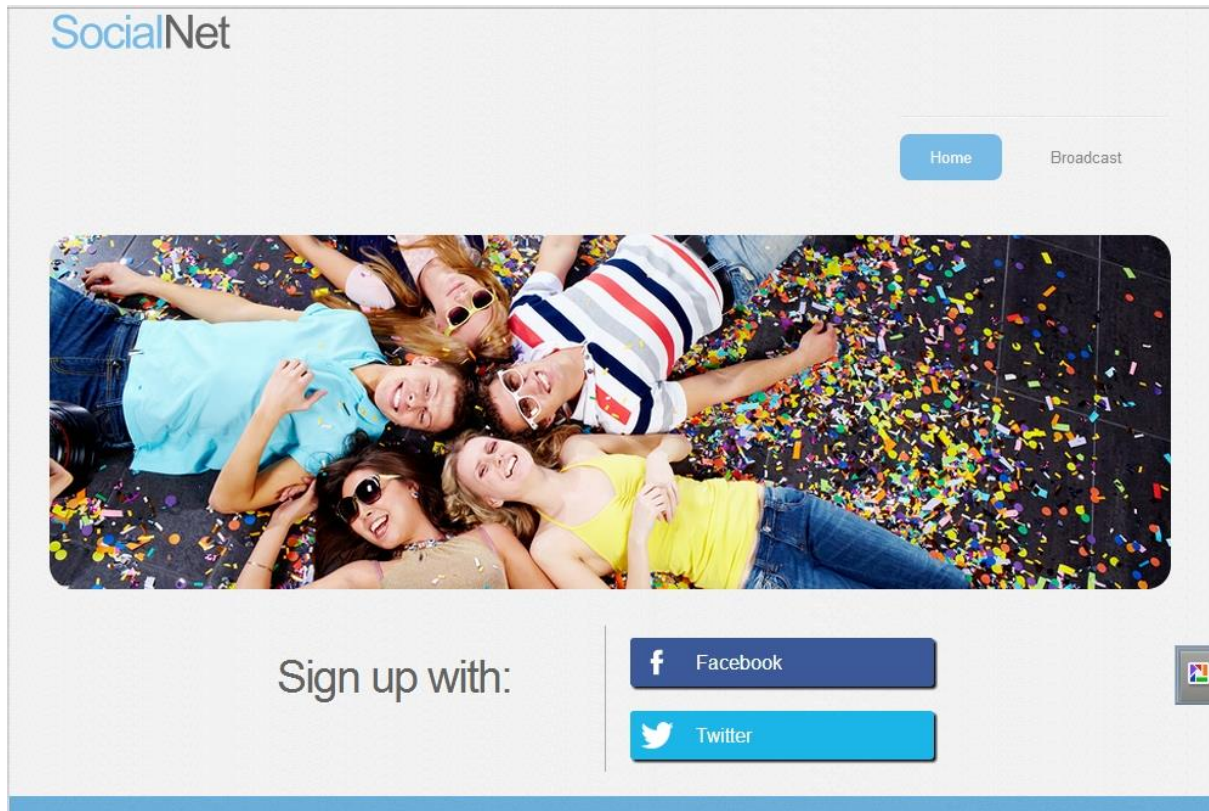
### 7.1. Output Screens

The user interface is kept simple and understandable. The user need not take any additional

effort to understand the functionality and navigation in the application. The colors are chosen in such a way that user can easily understand where the input has to be given.

The following are the main screens and features in this application.

### First Screen:



**Fig. No. 7.1 First Screen**

### Registration Screen:

SocialMedia

You are Logged in through :


Home

Account

Messages

Broadcast

logout



Register with a Unique Application ID!

AppID :

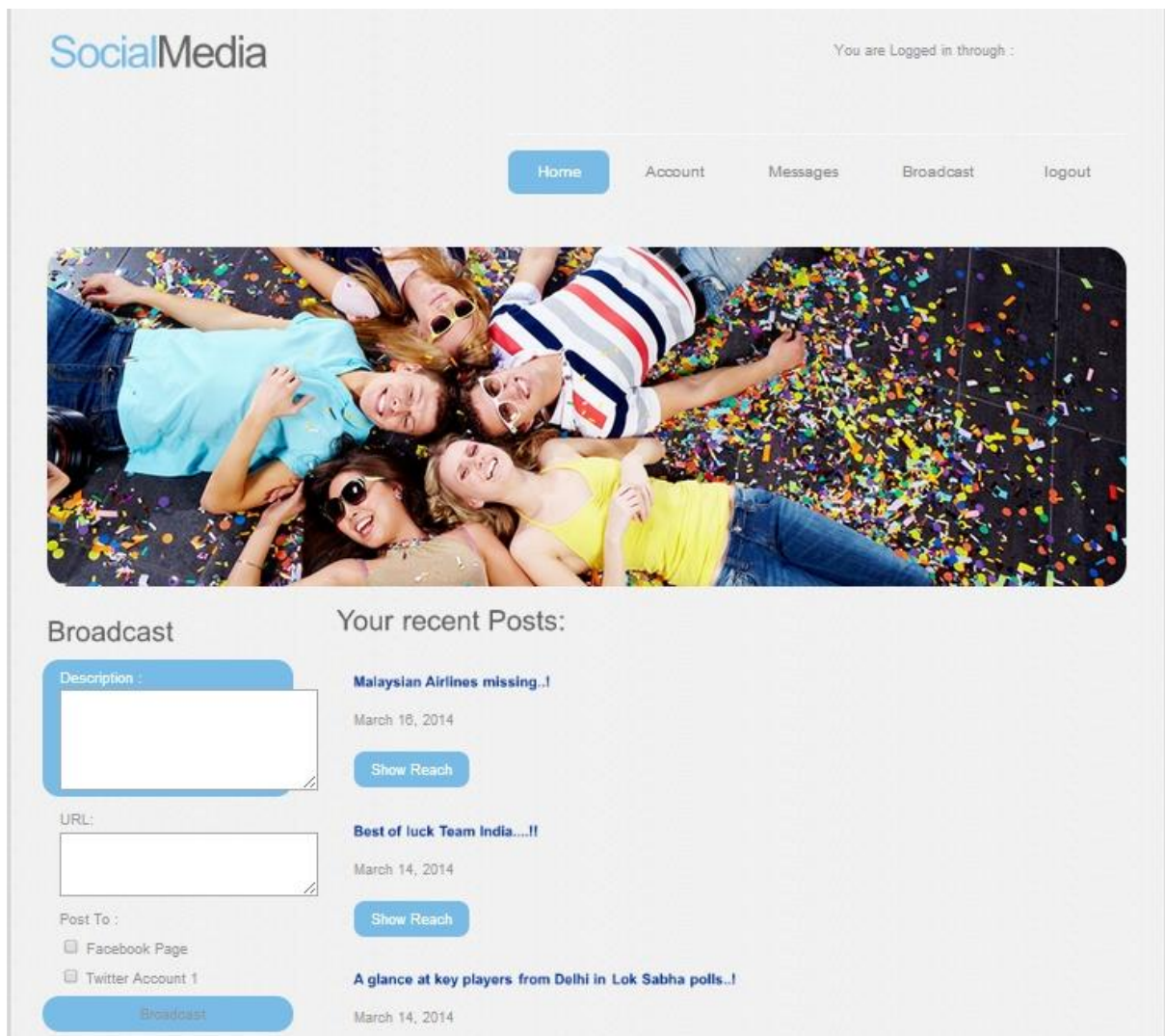
UserName already exists!

UserName:

Register

**Fig. No. 7.3 Registration Screen**

**Index Screen:**



**Fig. No. 7.4 Index Screen**

**Broadcast Screen:**

SocialMedia

You are Logged in through :


Home

Account

Messages

Broadcast

logout



Broadcast

Description :

URL:

Post To :

☐ Sample Page1

☐ Sample Page2

☐ First Page

☐ Sample Account1

☐ Sample Account2

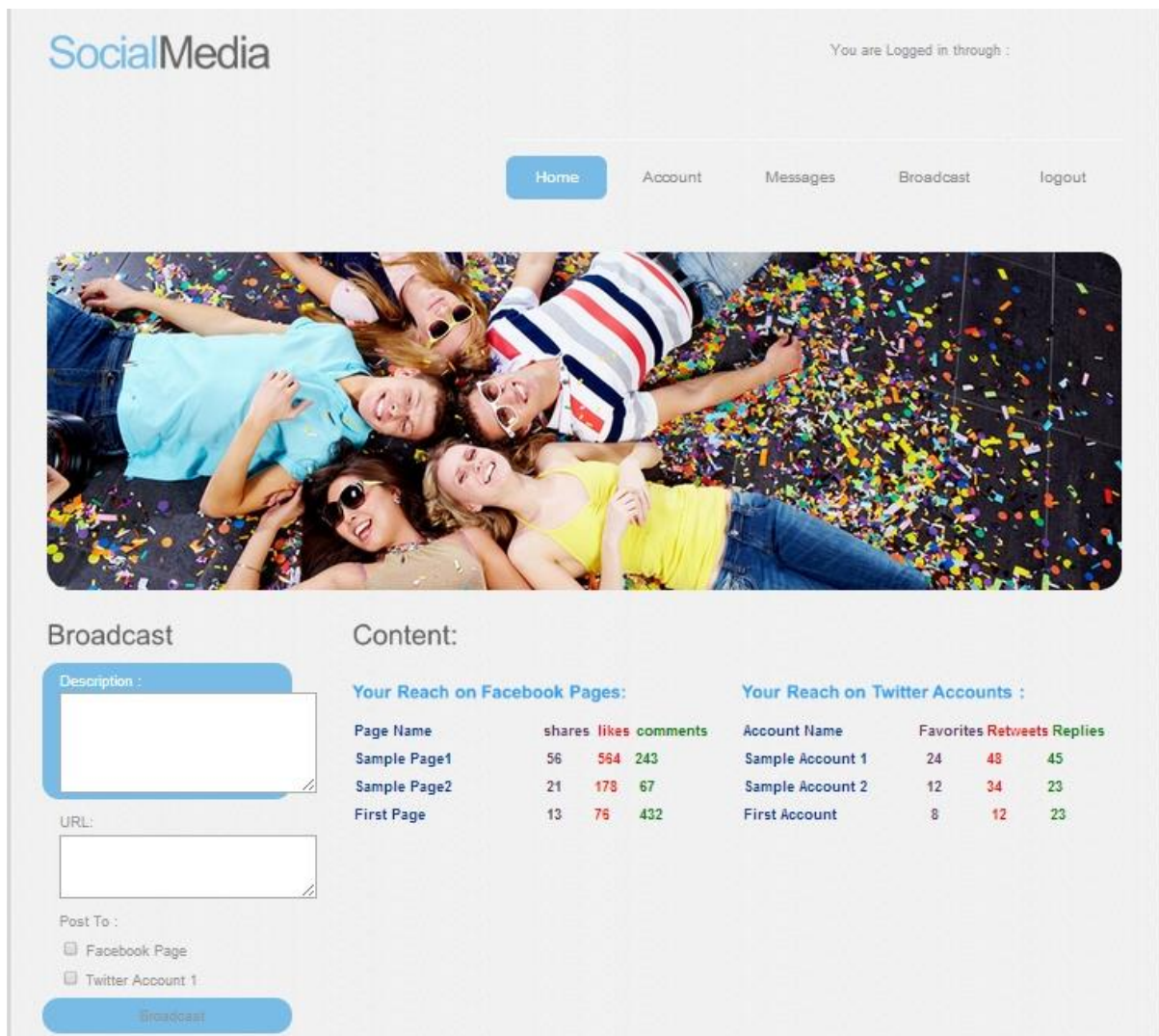
☐ First Account

Broadcast

851/Broadcast.html#

**Fig. No. 7.5 Broadcast Screen**

**Reach Screen:**



**Fig. No. 7.6 Reach Screen**

SocialMedia

You are Logged in through :

Home

Account

Messages

Broadcast

Logout

Broadcast

Description :

URL :

Post To :

☐ Facebook Page
 ☐ Twitter Account 1

Broadcast

AppID:

Your Facebook Pages:

Sample Page1	221 Subscribers
Sample Page2	187 Subscribers
First Page	201 Subscribers

+ Facebook

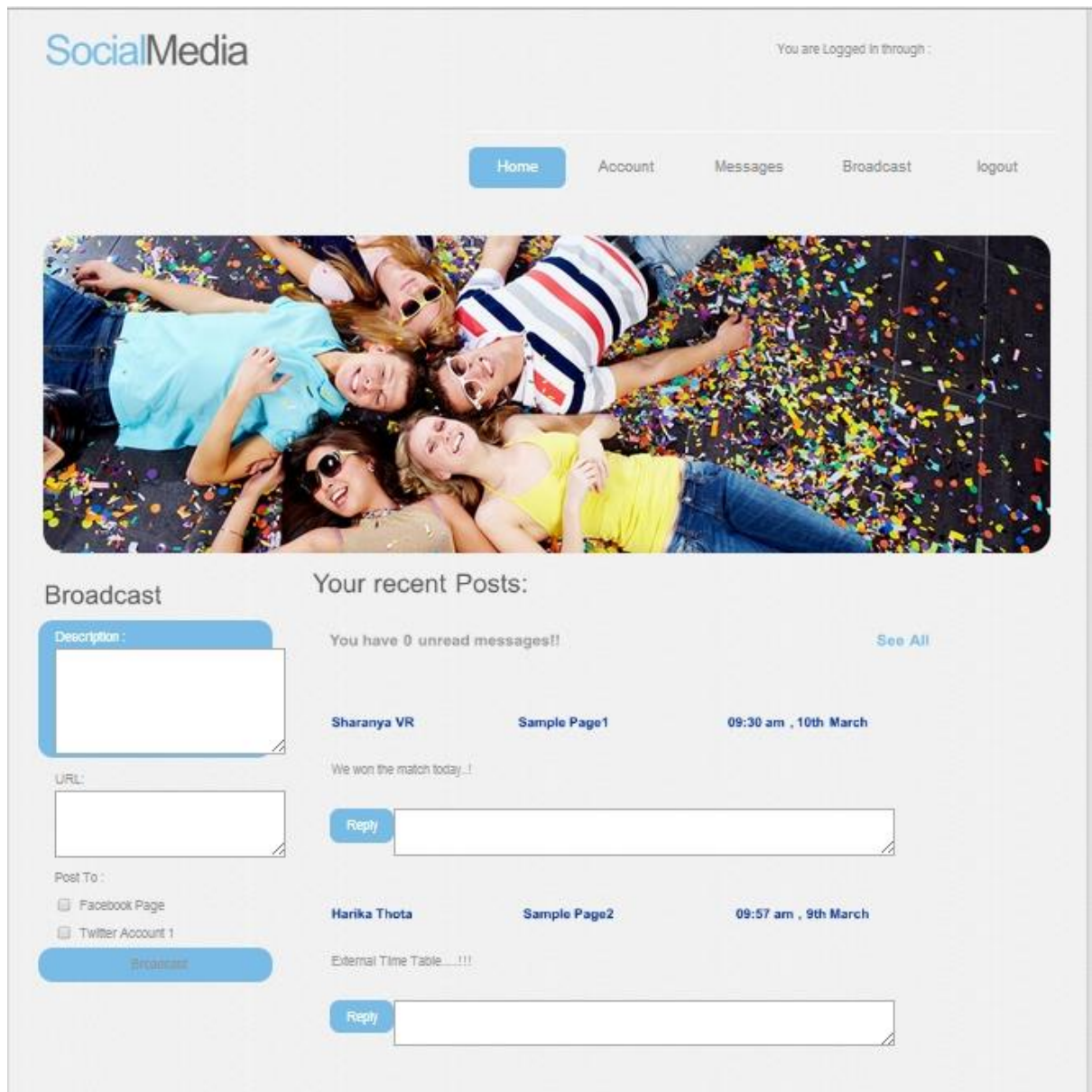
Your Twitter Accounts :

Sample Account1	330 Followers
Sample Account2	190 Followers
First Account	157 Followers

+ Twitter

**Fig. No. 7.7 Accounts Screen**

**Messages Screen:**



**Fig. No. 7.8 Messages Screen**

## **7.2 Future Enhancements**

**Auto Scheduling Posts:** The application can be further enhanced to auto-schedule content to post them at peak times and reach audience while not at desk using PC. This feature gives the leisure of posting things whenever the user wants without having to manually sit and upload at the required time.

**Analytics:** Graph analytics and demographic insights can be developed to know which messages drive the most traffic. It enables users to know the people that engage with their content by age, gender, location or more. This can be used as a great Marketing tool and can highly help in promotion of brands.

**Search Streams:** Search Streams can be designed such that target precise phrases and keywords can be searched and the users can see the conversations relevant to audience as they unfold.

**Increased Security levels:** Increase permission levels and give people as much or as little access as necessary thereby increasing security in the app. This feature will provide the ease of using the app not at the cost of security. Thus, each user can perform actions according to their roles.

**Targeted Messaging:** A feature to create messages and share them according to location, language, and demographic to have precision targeted posts. This feature can prove to be very useful to target regions or groups and to spread the Brand's reach to every corner of the masses.

### 7.3. Conclusion

The main aim of this application is to focus on the Corporate Companies which struggle to keep their users engaged through posts on social media sites. It also aims to provide analytics on data across sites through one interface.

This application eases the task of posting the same content across different pages and also enables the companies to analyze the statistics of posts across all the pages from one place. It also provides an interface to get feedback from users across all the pages.

This project has been a great learning platform for us to understand the fundamentals of Web-Application development. It gave us an insight into a very flexible and efficient framework (Play), the details of the API's of Facebook and Twitter and the database which can hold enormous amount of data (HBase).

This application has been tested using demo testers and developer accounts on Facebook and Twitter.

### 7.4. References:

- Lars George; HBase - The Definitive Guide., 2nd ed., O' Reilly, pp.499
- Herbert Schildt; The Complete Reference - Java., 7th ed., TBS, pp.1152
- Alexander Reelsen; Play Framework Cookbook., Packt Publishing, p.292
- PlayFramework - <http://www.playframework.com/documentation/2.2.x/JavaHome>
- HBase 0.99.0-Snapshot API - <https://hbase.apache.org/apidocs/>
- Twitter - REST API v1.1 Resources - <https://dev.twitter.com/docs/api/1.1>
- Facebook - Graph API Reference - <https://developers.facebook.com/docs/graph-api/reference/>
- LinkedIn Engineering - <http://engineering.linkedin.com/>
- Scribe-Java - <https://github.com/fernandezpablo85/scribe-java>
- Maven - <http://maven.apache.org/guides/index.html>
- Git - <http://git-scm.com/docs/gittutorial>