SERVICIO

Project Report

Submitted To

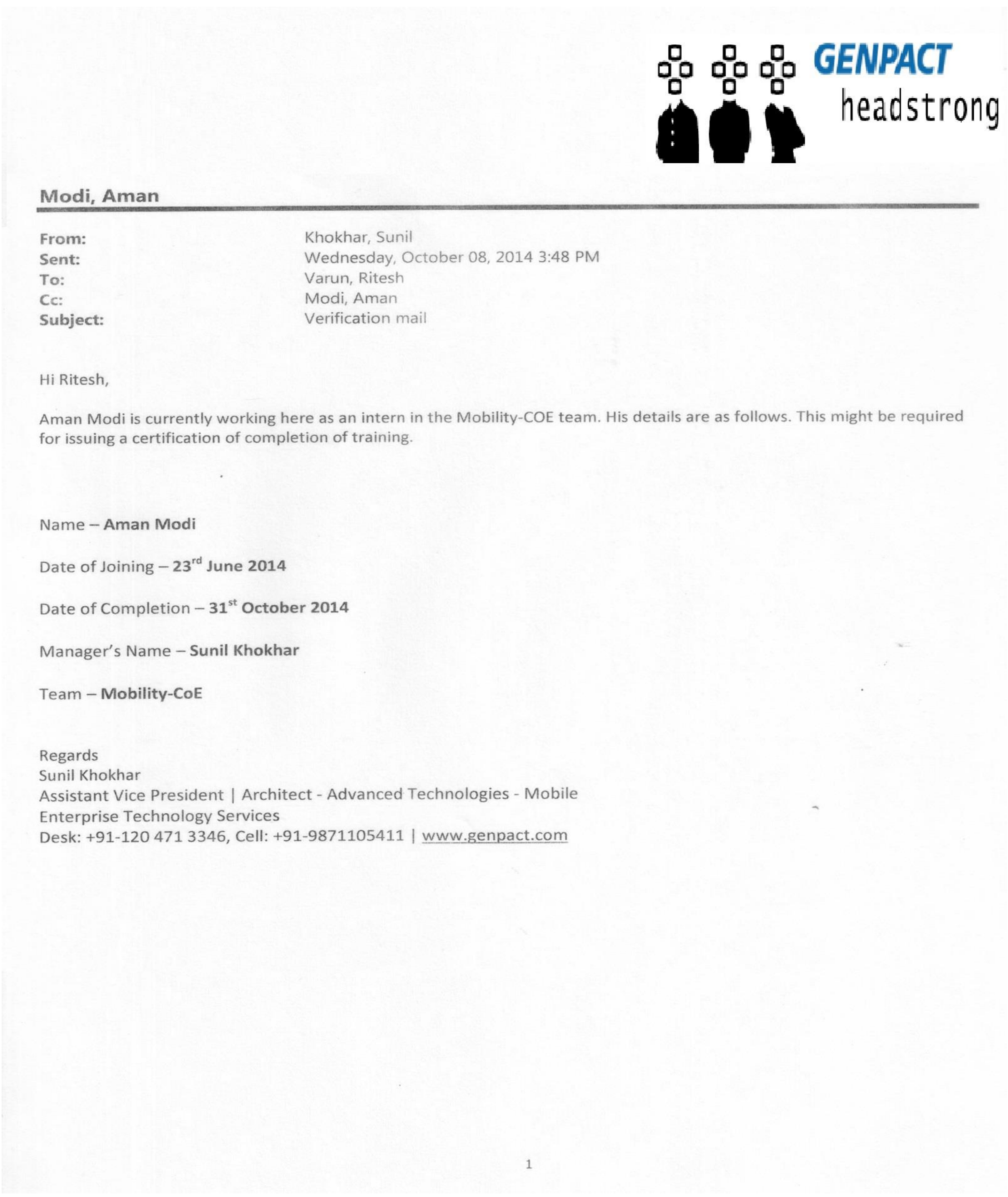
Department of Computer Science

Sharda University

Submitted By

Aman Modi

(110101034)



GENPACT headstrong

Modi, Aman

From:	Khokhar, Sunil
Sent:	Wednesday, October 08, 2014 3:48 PM
To:	Varun, Ritesh
Cc:	Modi, Aman
Subject:	Verification mail

Hi Ritesh,

Aman Modi is currently working here as an intern in the Mobility-COE team. His details are as follows. This might be required for issuing a certification of completion of training.

Name – **Aman Modi**

Date of Joining – **23rd June 2014**

Date of Completion – **31st October 2014**

Manager's Name – **Sunil Khokhar**

Team – **Mobility-CoE**

Regards
Sunil Khokhar
Assistant Vice President | Architect - Advanced Technologies - Mobile
Enterprise Technology Services
Desk: +91-120 471 3346, Cell: +91-9871105411 | www.genpact.com

1

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my guide Rahat Khanna for his exemplary guidance, monitoring and constant encouragement throughout the development of this project as well as during my internship at Genpact. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I also take this opportunity to express a deep sense of gratitude to Sunil Khokhar, Assistant Vice President, Genpact Headstrong Capital Markets, for his cordial support, valuable information and guidance, which helped me in completing this task through various stages.

I am obliged to staff members of Genpact Headstrong Capital Markets, for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Lastly, I thank almighty, my parents, brother and friends for their constant encouragement without which this assignment would not be possible.

(Aman Modi)

Technical Summary

Servicio is basically a web application designed especially for the field of Banking to help employees of the bank minimize their overhead and also to provide them with a proper channel of communication in their work. Servicio was designed as a web application because the ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers will be a key reason for its popularity, as is the inherent support for cross-platform compatibility.

Servicio is build using browser-supported programming languages such as **HTML5, CSS3, JavaScript, JQuery**. **Ajax**, a web development technique using a combination of various technologies, is an example of technology which creates a more interactive experience. At the backend, **MySQL** is used as the database. Node.js is used to create the web-server to process request via HTTP. **Node.js** is a cross-platform runtime environment for server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows and Linux with no changes. **WAMP** is an acronym for an archetypal model of web service solution stacks, originally consisting of largely interchangeable components: Windows, the Apache HTTP Server, the MySQL relational database management system, and the PHP programming language. As a solution stack, LAMP is suitable for building dynamic web sites and web applications has also been used in the development of Servicio.

APIs are also created and used in Servicio for accessing the database. Servicio allows managers to allocate certain visits and assign particular task with a due date to the employees. The employees can then find out the contact person for the particular visit using the Employee Dashboard of Servicio. Employee can also update the status of the task to keep the manager updated. Few other features are also provided such as weather tracker and a calendar on the dashboard itself.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

This document details the design implementation required for the Bank Customer Service Solution Application for web, mobiles and phablets. This document will cover each of the system's architecture, as well as offer a preliminary glimpse of the flow of the app functionality.

1.2 Scope

It covers the domain of Bank Customer Service Solution system. To facilitate the managers assign tasks to employees and for employees to keep a track of tasks assigned to them. It will also track the status of the tasks.

1.3 Assumption

Assumptions are made for the following:

1. Google APIs will be used for locating address.

1.4 Risk Issues

The risk factors involved with the application are:

1. The API doesn't work
2. Error at the backend in the data base.
3. Error in server
4. Any change in requirement.
5. Network Connectivity issues.
6. Any change in database externally may risk security issues.

2 Architecture Overview

2.1 Detailed Architecture

The Architecture of the Bank Customer Service Solution app consists of the server side and the client side which is briefly elaborated as:

We are building a website where in the manager will be able to allocate a company and the client representative of that company and certain tasks for a particular visit to be done to the Service Employee. The Employee will receive notifications whenever a visit-task will be allocated to him and he has to then send a feedback to the manager of the task performed by him. The first page will be a login page. Based on the login the next webpage will be displayed. We will have separate functionality for Manager, Service Employee & Admin.

No. of Users: 3

- Admin
- Manager
- Employee

2.2 Functionality by Role

2.2.1 *Admin*

- Add a new User | Assign Role to him (i.e. Manager, Employee or Client).
- Update details of User.
- Add a new Company.
- Update details of the company.
- Soft delete the details of User or Company if required.

2.2.2 *Manager*

- Select a Service Employee and allocate a company and the client representative of that company.
- Allocate certain Task to the Service Employee for a particular visit.
- Send notification/mail to Service Employee when a Visit-Task is allotted.

2.2.3 *Service Employee*

- Lists to update on completion of a Task assigned in that particular visit.
- Generation of a feedback report.
- Send feedback report to manager.

2.3 Extra Functionality

- A calendar can be provided to keep record of the Visit Task.

- The website can be transferred to *PhoneGap* to create mobile applications as well.

3 Specific Consideration

3.1 Authentication

The username and password will be authenticated from the server backend in order to have a successful login. While using the Bank Customer Services related to Username will be authenticated from the backend server.

3.2 Authorization

Authorization will be done by the admin when he will be storing in the details.

4 Database

Table 1: Schema for User Table

Fields	Data type	Index
U_ID	Int(6)	Primary Key
UserName	Varchar(20)	
Name	Varchar(30)	
Sex	Varchar(6)	
DOB	Date	
Age	Int(3)	
Address	Varchar(100)	
City	Varchar(30)	
State	Varchar(30)	
Mobile	Bigint(10)	
Email	Varchar(50)	
Role	Varchar(15)	
Password	Varchar(15)	
IsArchive	Boolean	
DateCreated	Date Time	

Table 2: Schema for Company Table

Fields	Data type	Index
C_ID	Int(6)	Primary Key
CompanyName	Varchar(40)	
City	Varchar(30)	
U_ID	Int(6)	Foreign Key(User Table)
IsArchive	Boolean	
DateCreated	Date Time	

Table 3: Schema for Relation Table

Fields	Data type	Index
R_ID	Int(6)	Primary Key
E_ID	Int(6)	Foreign Key(User Table)
M_ID	Int(6)	Foreign Key(User Table)
IsArchive	Boolean	
DateCreated	Date Time	

Table 4: Schema for Visit Table

Fields	Data type	Index
V_ID	Int(6)	Primary Key
E_ID	Int(6)	Foreign Key(User Table)
M_ID	Int(6)	Foreign Key(User Table)
C_ID	Int(6)	Foreign Key(Company Table)
DOA	Date Time	
DOC	Date	
IsArchive	Boolean	
DateCreated	Date Time	

Table 5: Schema for Task Table

Fields	Data type	Index
T_ID	Number(10)	Primary Key
V_ID	Varchar(100)	Foreign Key(Visit Table)
Task	Varchar(200)	
Status	Varchar(20)	
IsArchive	Boolean	
DateCreated	Date Time	

Table 6: Schema for Message Table

Fields	Data type	Index
MSG_ID	Number(10)	Primary Key
MsgTo	Number(10)	Foreign Key(User Table)
MsgFrom	Number(10)	Foreign Key(User Table)
Subject	Varchar(200)	
Message	Varchar(1000)	
IsArchive	Boolean	
DateCreated	Date Time	

Table 7: Schema for Feedback Table

Fields	Data type	Index
F_ID	Number(10)	Primary Key
E_ID	Number(10)	Foreign Key(User Table)
M_ID	Number(10)	Foreign Key(User Table)
Subject	Varchar(200)	
Message	Varchar(1000)	
IsArchive	Boolean	
DateCreated	Date Time	

5 APIs

Table 8: API for User Table

POST	http://localhost:7000/api/user	Add a new user
GET	http://localhost:7000/api/user	Select all users
GET	http://localhost:7000/api/userbyrole/:Role	Select Users with particular role
GET	http://localhost:7000/api/userbyusername/:UserName	Select a user by his UserName
PUT	http://localhost:7000/api/userupdate/:UserName	Update details of user by UserName
PUT	http://localhost:7000/api/userdelete/:UserName	Delete a User via UserName
PUT	http://localhost:7000/api/userpasswordupdate/:UserName	Update Password

Table 9: API for Company Table

POST	http://localhost:7000/api/company/	Add a new company
GET	http://localhost:7000/api/company/	Select all company
GET	http://localhost:7000/api/company/:C_ID	Select a company using company name
PUT	http://localhost:7000/api/companydelete/:C_ID	Delete a company
PUT	http://localhost:7000/api/companyupdate/:C_ID	Update Details of a company
GET	http://localhost:7000/api/companyname/	Select all Company's Name & ID

Table 10: API for Relation Table

POST	http://localhost:7000/api/relation/	Add a New Relation
GET	http://localhost:7000/api/relation/	Select All relations
GET	http://localhost:7000/api/relation/:E_ID	Select a relation based on E_ID
PUT	http://localhost:7000/api/relationdelete/:E_ID	Delete a Relation

Table 11: API for Visit Table

POST	http://localhost:7000/api/visit/	Add a new Visit
GET	http://localhost:7000/api/visit/	Select all Visits
GET	http://localhost:7000/api/visit/:V_ID	Select a Visit based on V_ID
PUT	http://localhost:7000/api/visitdelete/:V_ID	Delete a Visit

Table 12: API for Task Table

POST	http://localhost:7000/api/task/	Add a New Task
GET	http://localhost:7000/api/task	Select all Tasks
GET	http://localhost:7000/api/taskbyvisit/:V_ID	Select all Tasks for a particular Visit
GET	http://localhost:7000/api/taskbytask/:T_ID	Select a Task based on T_ID
PUT	http://localhost:7000/api/taskupdate/:T_ID	Update the status of a Task
PUT	http://localhost:7000/api/taskdelete/:T_ID	Delete a Task

Table 13: API for Message Table

POST	http://localhost:7000 /api/message	Add a new Message
GET	http://localhost:7000 /api/inbox/:MsgTo	Select Messages
GET	http://localhost:7000 /api/viewmessage/:MsgID	View Detailed Message

Table 14: API for Feedback Table

POST	http://localhost:7000 /api/feedback	Add a new Feedback
GET	http://localhost:7000 /api/feedback/:M_ID	Select Feedbacks
GET	http://localhost:7000 /api/viewfeedback/:F_ID	View Detailed Feedback

Table 15: API for Miscellaneous Use

GET	http://localhost:7000/api/messagenotif/:U_ID	Count of Messages
GET	http://localhost:7000/api/pendingtaskforemployee/:E_ID	Count of Pending Task
GET	/api/pendingvisitadmin	Select a Task based on T_ID

6 E-R Diagram

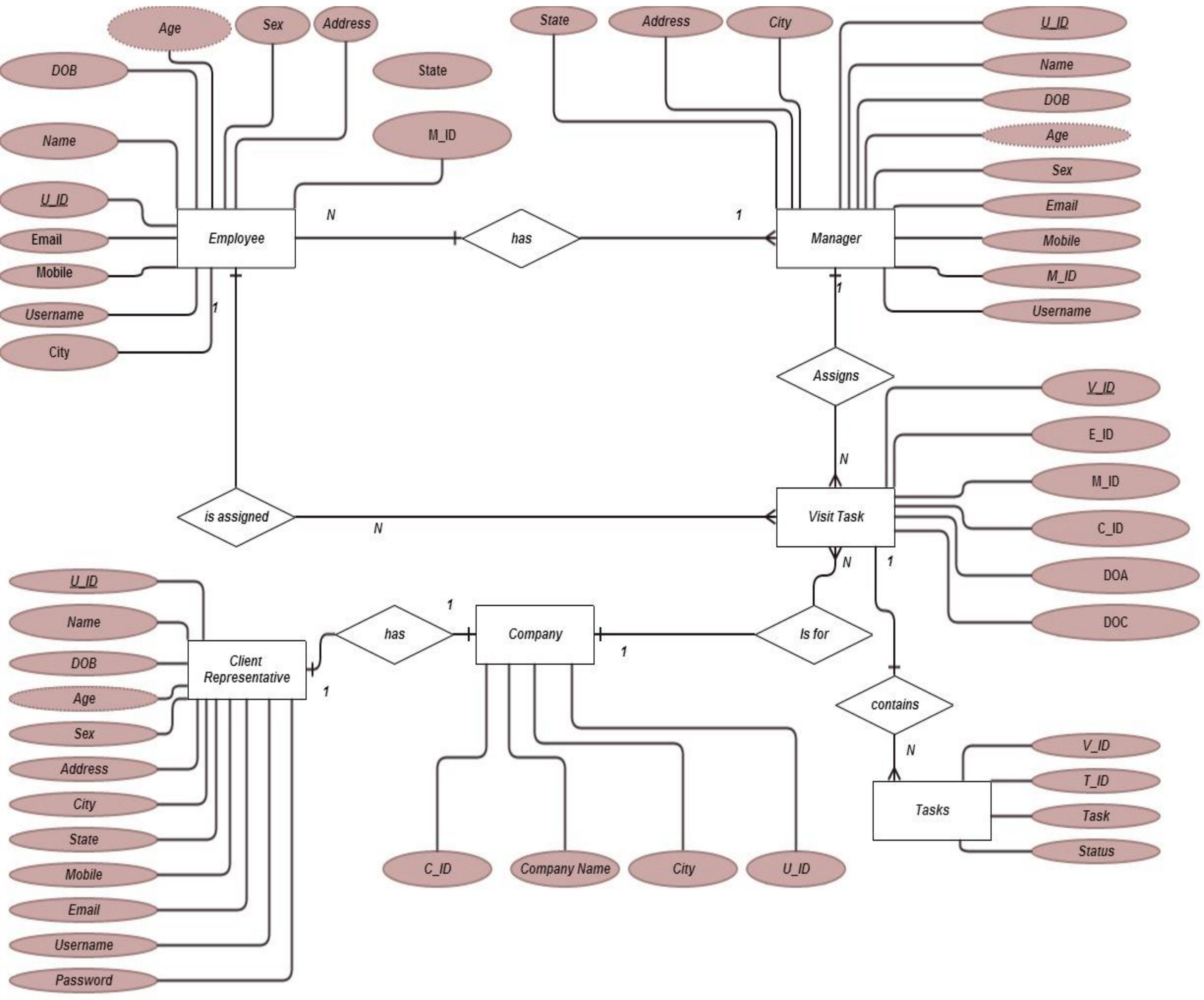


Figure 1: E-R Diagram for Servicio

7 Snapshots

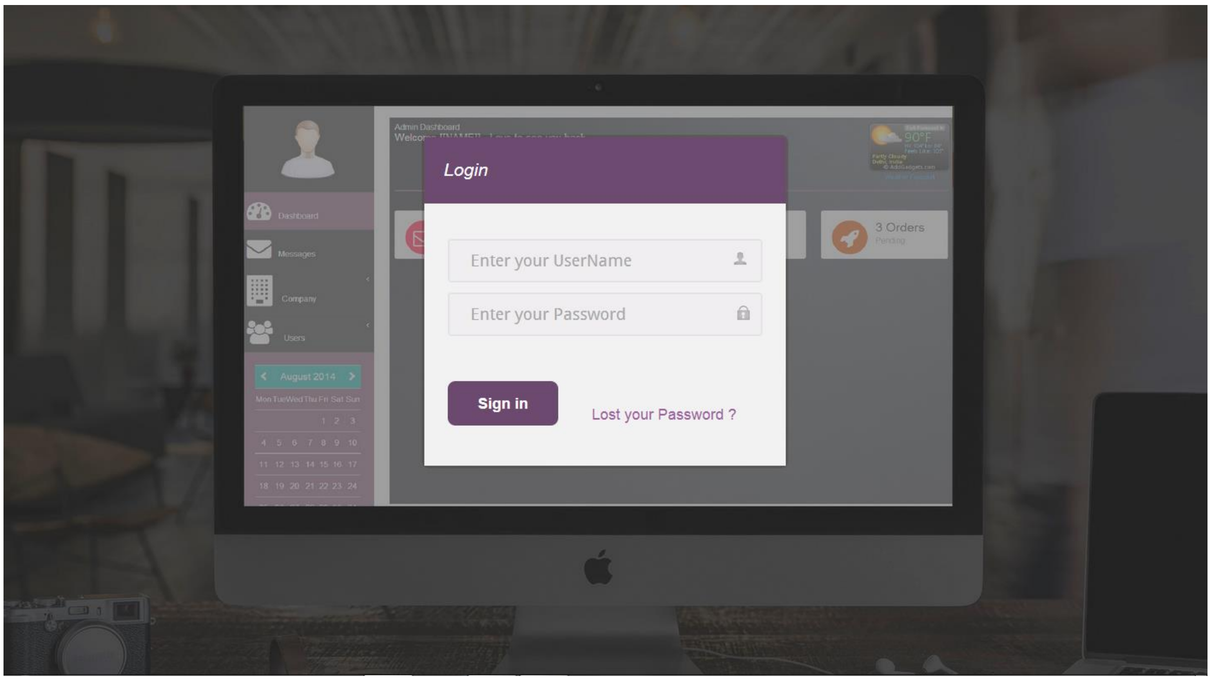
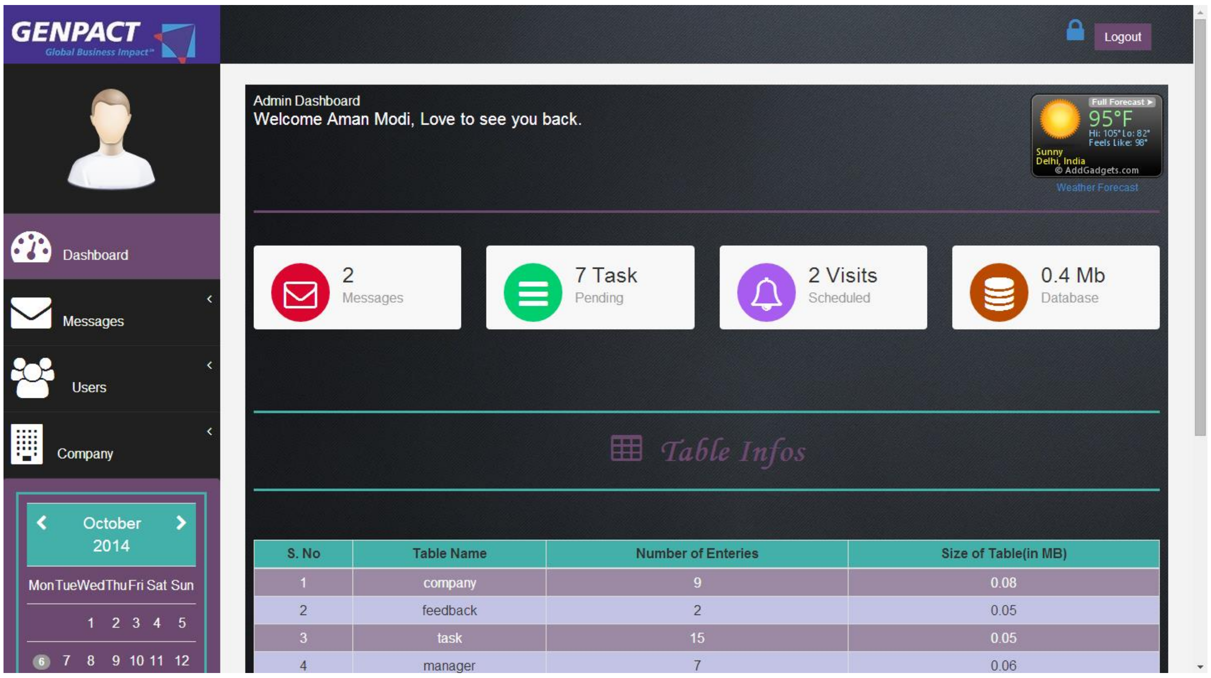


Figure 2: Login Screen



Admin Dashboard
Welcome Aman Modi, Love to see you back.

S. No	Table Name	Number of Enteries	Size of Table(in MB)
1	company	9	0.08
2	feedback	2	0.05
3	task	15	0.05
4	manager	7	0.06

Figure 3: Admin Dashboard

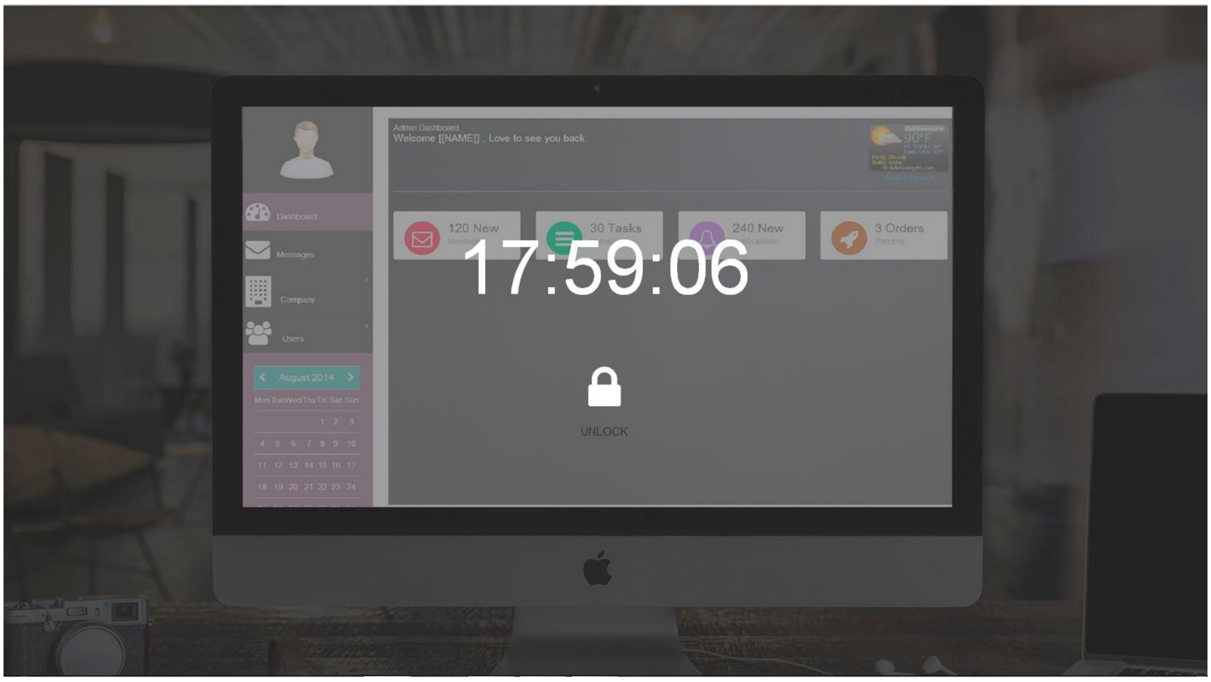


Figure 4: Break Lock

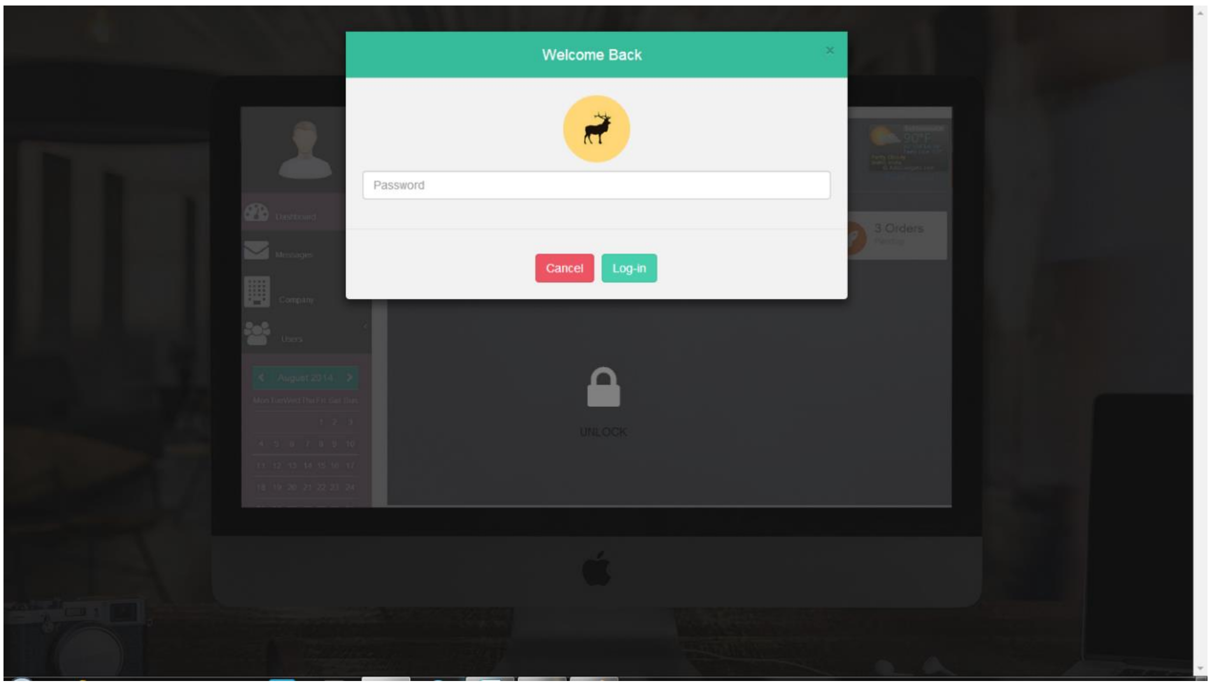
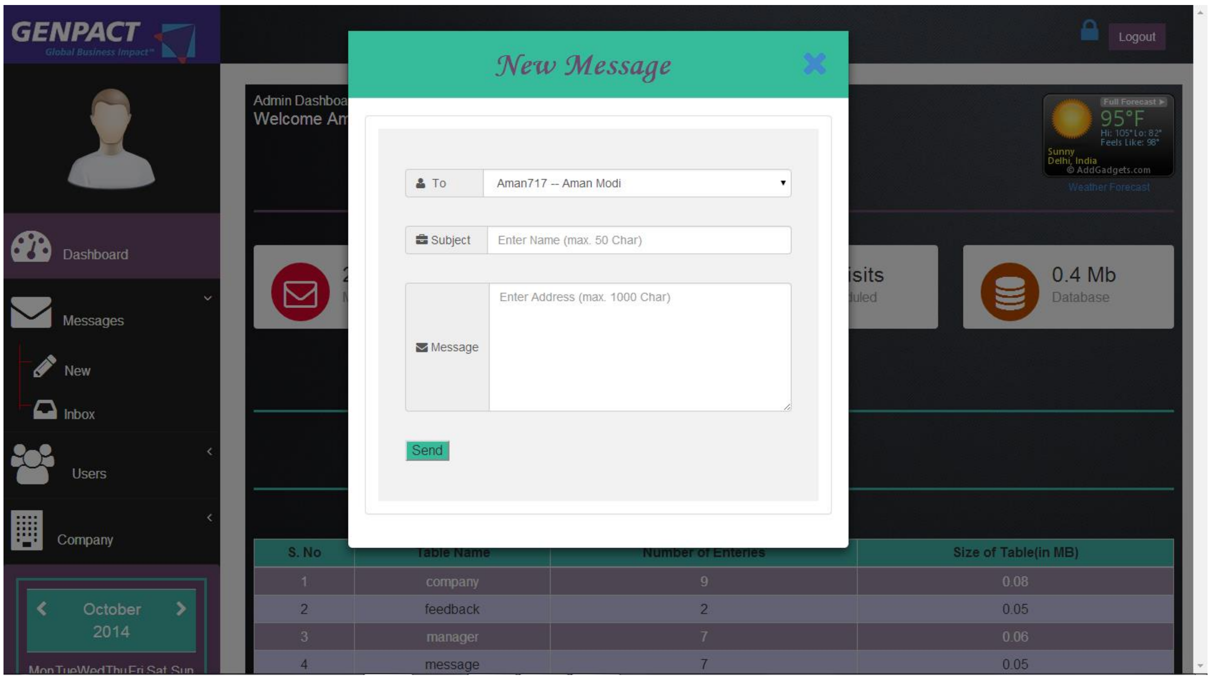


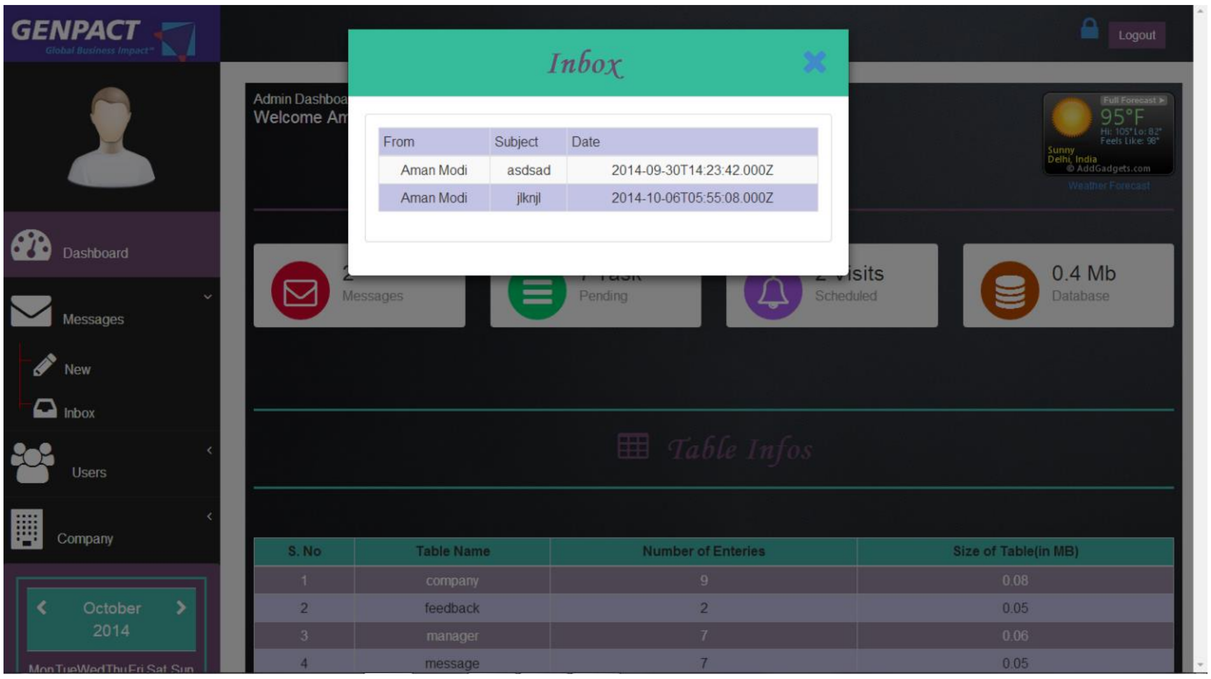
Figure 5: Break Lock Password



To	Aman717 -- Aman Modi
Subject	Enter Name (max. 50 Char)
Message	Enter Address (max. 1000 Char)

S. No	Table Name	Number of Enteries	Size of Table(in MB)
1	company	9	0.08
2	feedback	2	0.05
3	manager	7	0.06
4	message	7	0.05

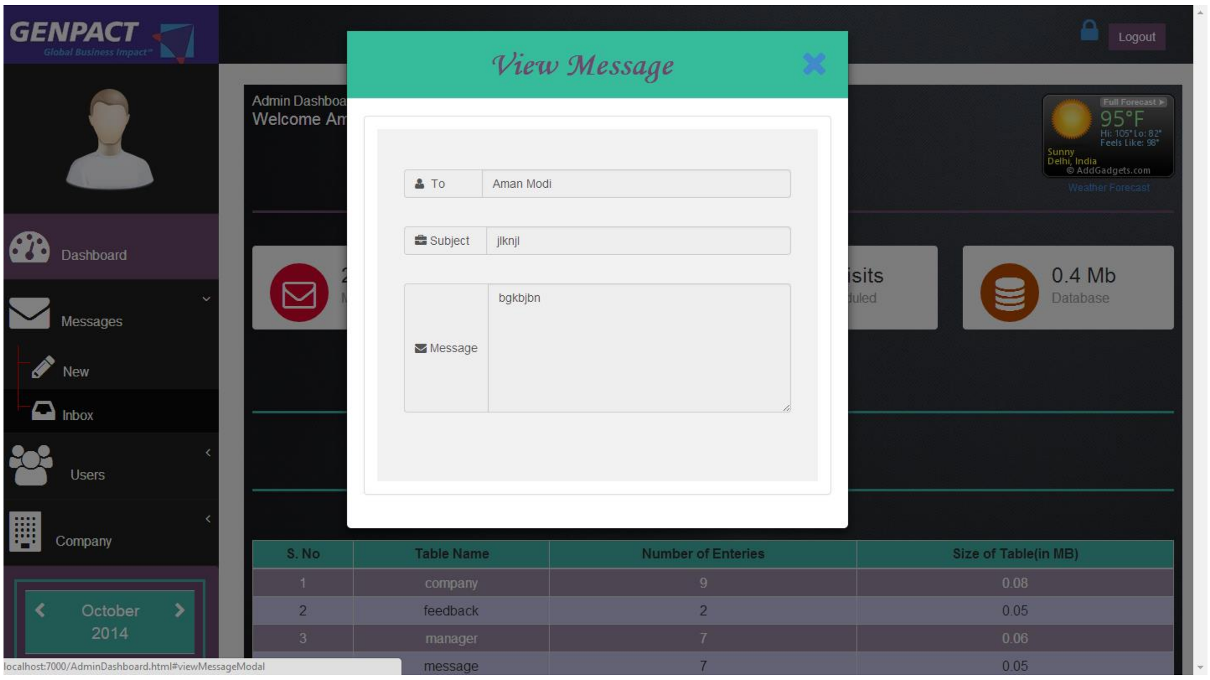
Figure 6: New Message



From	Subject	Date
Aman Modi	asdsad	2014-09-30T14:23:42.000Z
Aman Modi	jlknjl	2014-10-06T05:55:08.000Z

S. No	Table Name	Number of Enteries	Size of Table(in MB)
1	company	9	0.08
2	feedback	2	0.05
3	manager	7	0.06
4	message	7	0.05

Figure 7: Inbox

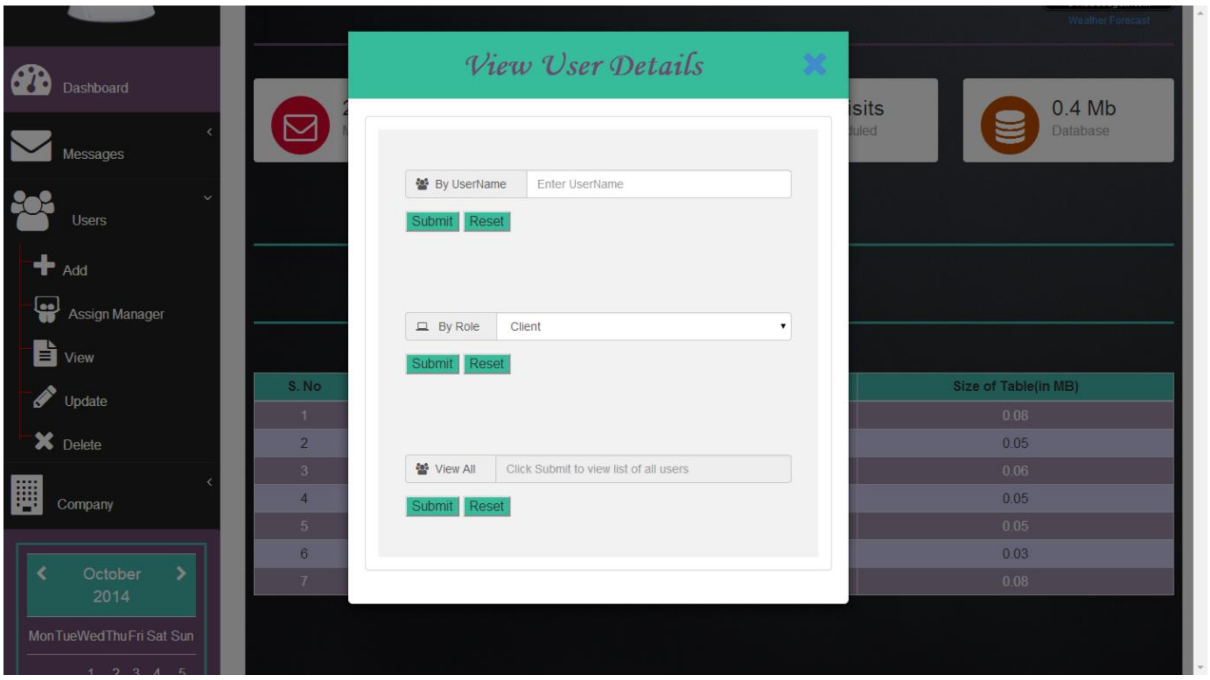


To	Aman Modi
Subject	jlknjl
Message	bgkbjbn

S. No	Table Name	Number of Enteries	Size of Table(in MB)
1	company	9	0.08
2	feedback	2	0.05
3	manager	7	0.06
	message	7	0.05

localhost:7000/AdminDashboard.html#viewMessageModal

Figure 8: View Message



By UserName	Enter UserName
By Role	Client
View All	Click Submit to view list of all users

S. No	Size of Table(in MB)
1	0.08
2	0.05
3	0.06
4	0.05
5	0.05
6	0.03
7	0.08

Figure 9: View User Details

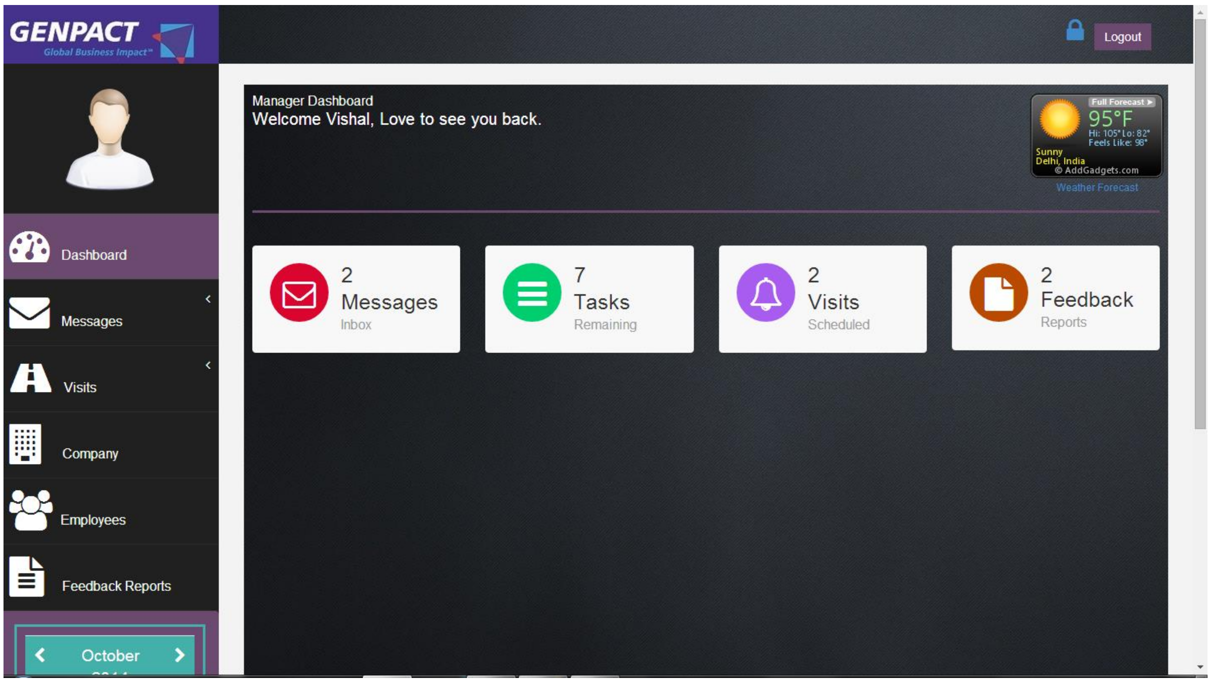
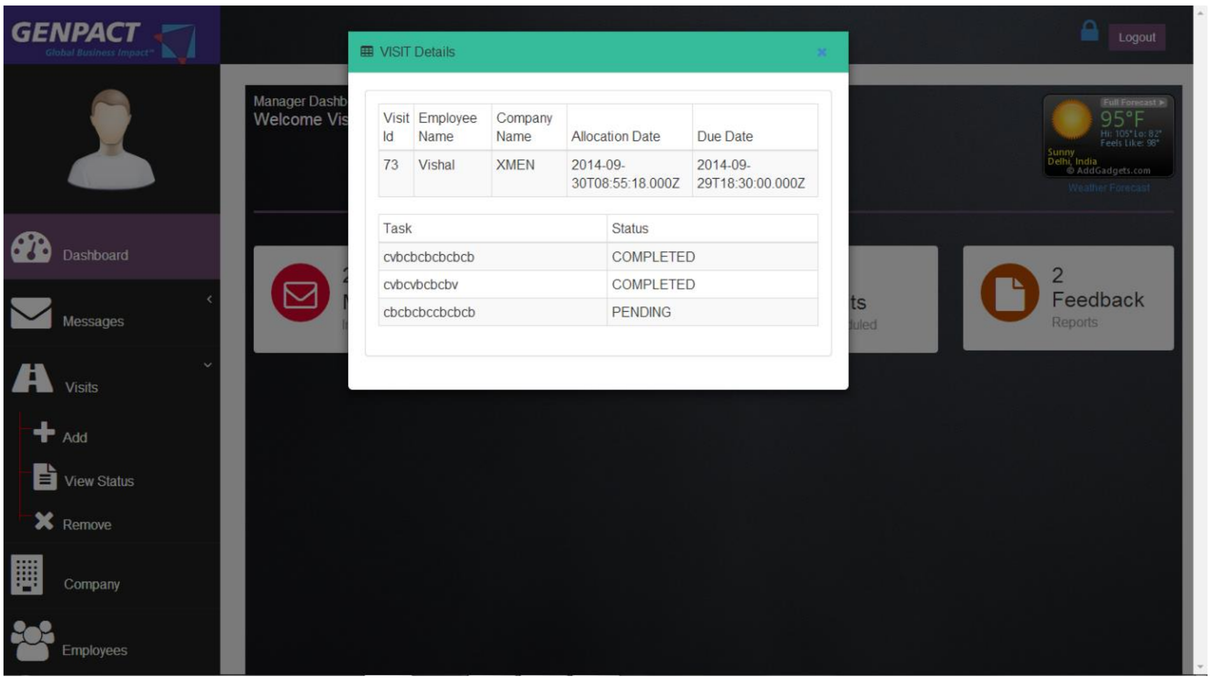


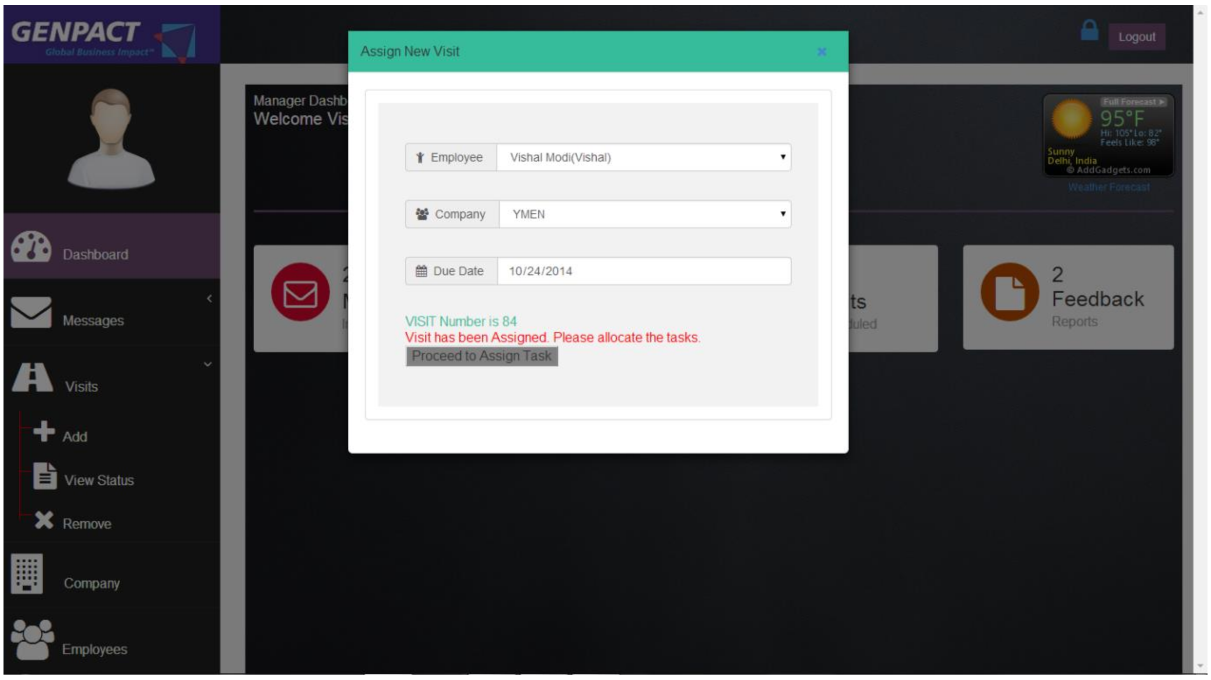
Figure 10: Manager Dashboard



Visit Id	Employee Name	Company Name	Allocation Date	Due Date
73	Vishal	XMEN	2014-09-30T08:55:18.000Z	2014-09-29T18:30:00.000Z

Task	Status
cvbcbcbcbcbcb	COMPLETED
cvbcvbcbcbv	COMPLETED
cbcbcbccbcbcb	PENDING

Figure 11: Visit Status



Assign New Visit

Employee: Vishal Modi(Vishal)

Company: YMEN

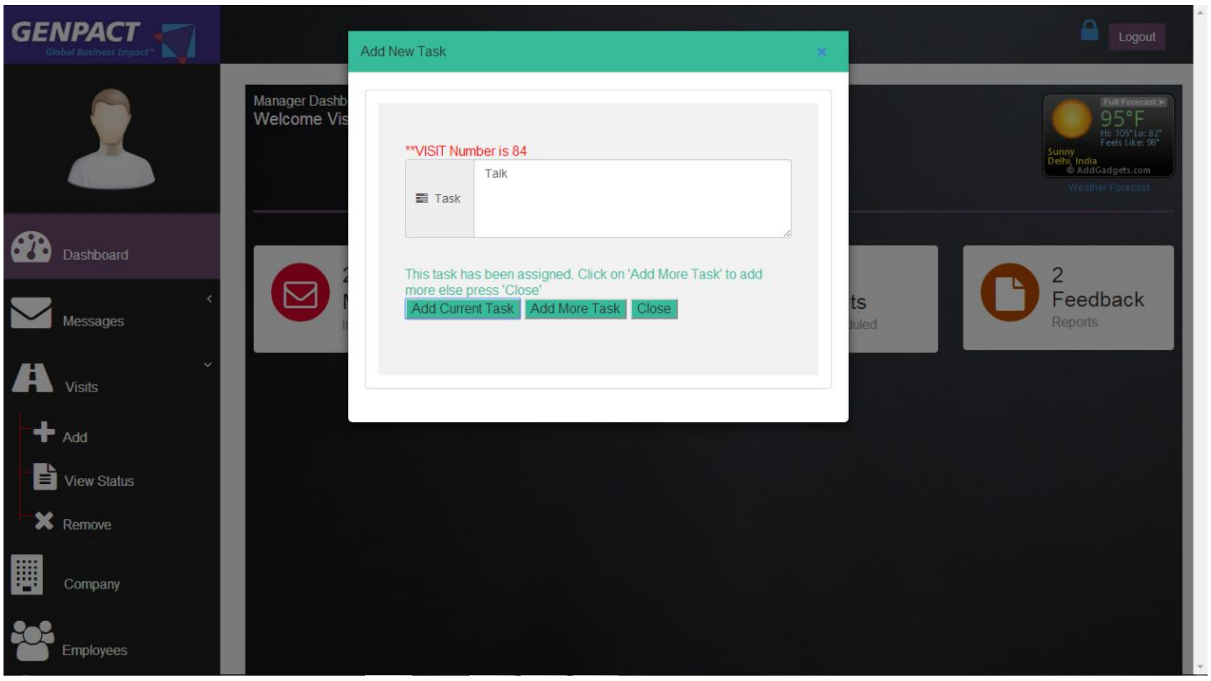
Due Date: 10/24/2014

VISIT Number is 84

Visit has been Assigned. Please allocate the tasks.

Proceed to Assign Task

Figure 12: Assign New Visit



Add New Task

**VISIT Number is 84

Task: Talk

This task has been assigned. Click on 'Add More Task' to add more else press 'Close'

Add Current Task | Add More Task | Close

Figure 13: Assign Task for New Visit

8 Conclusion

Project Servicio made the work of Manager and Employee look very simple. With the help of Servicio, now the managers can easily keep a track of their Service employees as well as the visits that are being allotted. Servicio also provided the ability to store the task which again made it very easy for the managers to confirm the status of the tasks assigned just by using the web application directly. With the help of Servicio now the managers will be able to be more productive, easily maintain the records of the visits and the tasks. They can track the task's status or can also look back at a visit assigned 4 months back in case needed with quite a bit of ease.

When I began with this project, Node.Js was like a foreign entity for me but now after the completion of this project I believe it is one of the most amazing platforms for building fast, scalable networks. Its non-blocking I/O model makes it very light weight and perfect for data-intensive real time applications. Learning Node.js was very exciting and it always amazed me with its performance as compared to other platforms.

Working on this project was really a very good experience for me. It made me open to a lot of new technologies which will be very prominent in near future. With the hybrid apps in development for Servicio, it will make it very efficient once they are completed as it will provide users with an opportunity to be updated to work and also work if they want while they are on the go.

9 References

1. *Node Js | Wikipedia*
   http://en.wikipedia.org/wiki/Node.js

2. *HTML, CSS, JavaScript | W3 School*
   http://www.w3schools.com/

3. *Node.Js | Youtube*
   https://www.youtube.com/watch?v=jo_B4LTHi3I

4. Manuel Keissling, "The Node Beginner Book"

10 Appendices

10.1 Abbreviations:

C_ID	-> Company Id
M_ID	-> Manager Id
R_ID	-> Relation Id
T_ID	-> Task Id
U_ID	-> User Id
V_ID	-> Visit Id
DOA	-> Date of Task Allocation
DOB	-> Date of Birth
DOC	-> Date of Task Completion
MOB	-> Mobile Number
Msg_ID	-> Message Id
F_ID	-> Feedback Id