ijecs open access www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 4 April 2017, Page No. 20976-20983
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i4.33

Hierarchical Messaging Application for Android Using Salesforce

Mr. Swapnil Manekar, Prof. Prashant Borkar, Mr. Vedprakash Hirwani
Department of Computer Science & Engineering
G. H. Raisoni College of Engineering
Email: manekar_swapnil.ghrcemtechcse@raisoni.net
Nagpur, India
Department of Computer Science & Engineering
G. H. Raisoni College of Engineering
Email: prashant.borkar@raisoni.net
Nagpur, India
Digital Business Unit (Salesforce)
Persistent Systems Limited
Email: vedprakash_hirwani@persistent.co.in
Nagpur, India

Abstract – Messaging applications are a dime a dozen nowadays and can only be differentiated on the basis of some unique features. These applications suffer from various limitations like free posting, joining, and lack of a hierarchy among other things when considered from an organizational or professional point of view. This paper presents the technology behind and structure of an application that limits the users in accordance with an organization specific hierarchy kept separate from the application to keep it universal so that this communication application would be professional and clutter-free. Users would be able to read relevant messages. Post if allowed, get update or news, and hale organizational events scheduled for them. These features would run on Android smartphones while other users would only be able to read the important messages via text SMS thus making the project available to all allowed, though in a limited capacity.

Keywords - Android; Salesforce; Communication; Cloud; Messaging; IIMP – Instant Important Messaging Platform.

I. INTRODUCTION

Computer-based messaging can be found as far back as early computing when users needed to communicate via archaic software. As the technology that allowed text messaging spread to the general public, people quickly became accustomed to communicating with peers or groups of people via text from a mobile or PC. Until the social networking boom, messaging was divided into two parts: you either chatted on your computer or you tested on your phone. The increased popularity of social networks and smartphones changed that, merging online behavior with constant mobile access. The new wave of mobile chat applications gained popularity as low- or no-cost alternatives to testing, but over time many changed into media portals, recognizing that media and content are basically how we now share and communicate in conversation [1]

A messaging application acts primarily as a one-to- one (or -few) communication mechanism, and can be temporary or long-lasting. Content is intended to be private or at least directed towards a specific group. A social network consists of "many to many" connections and is durable. When used to publish information, it acts primarily as a many-to- many broadcast mechanism.

Content is essentially public. However, both suffer from issues when considered from the point of view of an organization that wants to deploy such a system for their professional communication. The fundamental issue that crops up is that all the users of those platforms have equal rights and therefore such a system would not be suitable for a hierarchical system. In order to overcome this, among many other issues, an organization can either develop an internal solution, which would require time and effort for a very specific system or use an external solution which is what we aim to provide. We have created an application, IIMP, suitable for communication use by organizations that follow a hierarchy. The communication can be in the farm of messages or news/events. The flow of communication is controlled based on the direction of flow.

In the second section, we have discussed the related work done in our scope. The third section describes the background knowledge to understand our paper. In next section, we have given a description of our design. The fifth section lists out the functions and features given by our application. While the sixth section informs about usage scenarios and future scope. The last section gives a conclusion.

II. RELATED WORK

There are a lot of messaging applications fighting for space in a crowded market space but they cater to general needs and are not tailored specifically for an application like modeling the hierarchy of an organization. For example, WhatsApp caters to the general public and therefore cannot provide the structure and/or rules followed in any organization. The same goes for most applications out there. IIMP also offers the ability to create a group based on the certain condition that a user has to satisfy in order to be a member of that group. IIMP depends on Salesforce for cloud storage and functionality. Salesforce does offer an application called "Chatter" for intra-organization social-networking but it is built on their mobile API that misses out on Android-OS customizability. Also, IIMP has a broader future scope as stated in section 6.

III. BACKGROUND KNOWLEDGE

A. *Salesforce*

Salesforce is a cloud offering marketed as a CRM platform. Customer relationship management (CRM) is a term that refers to practices, strategies, and technologies that companies use to manage and analyze customer interactions and data throughout the customer lifecycle, with the goal of improving business relationships with customers, assisting in customer retention and driving sales growth. CRM systems are designed to compile information on customers across different channels, or points of contact between the customer and the company, which could include the company's website, telephone, live chat, direct mail, marketing materials and social media. CRM systems can also give customer-facing staff detailed information on customers' personal information, purchase history, buying preferences and concerns [2]

Salesforce is object-oriented meaning that data is stored in the form of objects rather than relational tables. Custom Objects have to be created in order to model the specific application's needs. Some pre-defined objects are available that can be customized. Inside objects, we have fields that can be thought of as the attributes. They can be pre-existing or custom just like in the case of objects. Once these have been defined we can provide relationships among them in the form of lookup or pickup settings for fields. These work similarly to the foreign key concept. Next, we define the Apex classes. These Apex classes contain the code that would act as an interface between the application and the Salesforce objects. Since Salesforce works on HTTP/HTTPS we define methods for POST or GET requests as per requirement. We have chosen HTTPS for the extra security implied by using it. Next, *NW* assigns URLs that work as REST API from the point of

DOI: 10.18535/ijecs/v6i4.33

view of the application to the Apex classes we have created and we are good to go on the Salesforce side. Salesforce also provides certain features that enable us to create reports or a dashboard based on the data we have since it is a CRM. These, once created, can be viewed once we have logged into our Salesforce account.

B. *Android*

Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It was built to be truly open. For example, an application can call upon any of the phone's core functionality such as making calls, sending text messages, or using the camera, allowing developers to create richer and more cohesive experiences for users. Android is built on the open Linux Kernel. Furthermore, it utilizes a custom virtual machine that was designed to optimize memory and hardware resources in a mobile environment. Android is open source; it can be liberally extended to incorporate new cutting edge technologies as they emerge [3]. For Android, we have chosen Android Studio as our development environment because it is the industry norm and is recommended by the developers. For the layout, we create XML files that are kept separate and can be easily swapped in case of a change. For the Functionality, Java classes are created that can be classified as Activities, Fragments, Service classes, Broadcast Receivers, Services or Support classes for the purposes of our application.

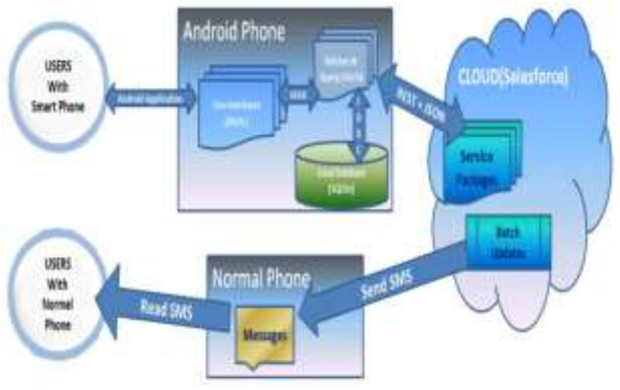

Fig. I. Overview Of Our Application.

IV. DESIGN

Fig. I, An Overview Of Our Application Is Elaborated.

A. *Salesforce*

Communication between the application and Salesforce is achieved through the use of REST API defined on the Salesforce cloud. HTTP GET and POST are used to request or send data respectively. An HTTP 200 response indicates a valid response. All the data that is transmitted is in the form of JSON. The REST API calling URL is called as a site in Salesforce with which Apex classes are associated so that a relationship materializes between resource and function. Apex classes can contain methods that handle the GET or POST requests appropriately. These methods have the ability to access the Custom Objects that have been created for the application. Custom Objects are like the table schemes we define in relational databases; custom or standard fields taking the place of columns or attributes. Linking between the objects is done through the type specification of fields in the object. If it is specified as lookup then the data in that field can only be something that is contained in the object that it looks up. New custom objects can be created or instances of them can be manipulated in the Apex class as per the need. The return value of the method is the response that we want to send to the application based on the request.

B. *UI*

The UI design is intentionally kept simplistic so that it is naturally intuitive and reflects the professional nature of the application. Since the application is built around data, it is kept front and center, the other features requiring further interaction and navigation for usage. The UI is further divided into 3 parts-Login, Home, Features.

1. Login:

Fig. 2, once a user installs the application, he is presented with a screen that prompts the user to make a choice between being an existing user or a new one. If he is an existing user, meaning that he had at least once used the application before, then he can log in by simply providing his previous mobile number after which an OTP is generated which needs to be verified. After this sequence of steps, the user reaches the home page which is described in the next sub-section. However, if the user is *new* and specifies as such, he has to fill the required details which are then sent to the Salesforce cloud after which an OTP is sent. After its verification, a request to verify the details is generated. Now, the user is blocked from accessing the application further until the handler he has specified in the registration verifies his details and updates his status on the cloud. The handler is a previously registered user that the new user specifies as having the right to verify or accept/reject his requests. The handler needs to be higher than the new user in the hierarchy or on the same level. Once the status is updated, the user can now use the application. Note that this process is required the first time the application is used on the device.

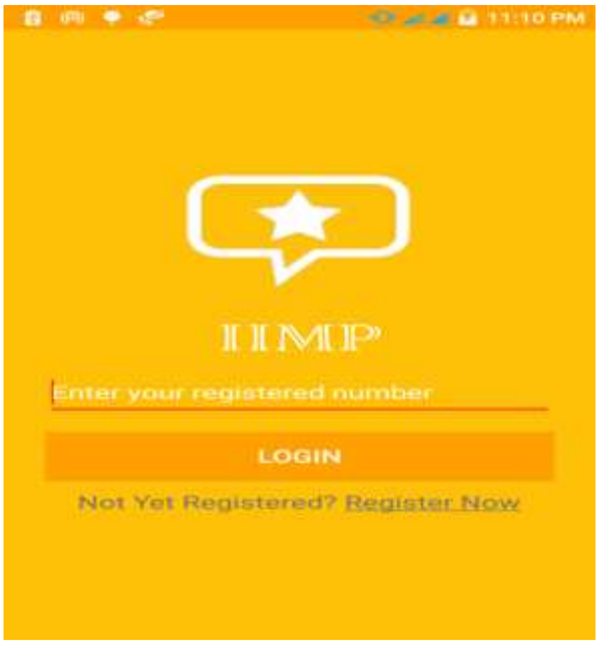
11:10 PM
IIMP
Enter your registered number
LOGIN
Not Yet Registered? Register Now

Figure 2. Login Screen

2. Home:

Fig. 3, once the user is allowed to access the actual application, he is greeted with a home screen that contains three major parts- group messaging tabs, personal message tab, and the navigation menu that directs the user to the other features. The group messaging tab displays the groups that the user is subscribed to. On clicking an individual group in the list, the user is redirected to an activity that displays the messages in that group. The message can be simple text, event, poll or an image with each being handled differently. The user can play one of three pre-defined roles in each group. A user may play a different role in a different group. He can be a normal user, a management type or an admin type user with each having increasing rights in the group respectively. A normal user can only view the messages in the group. A management type user can post messages as well. An admin can do everything a management type can and more. Admin has rights to perform anything possible in the group. In the activity, the user is identified to be one of the three specified above and accordingly the UI changes.

Since this type of messaging promotes downward messaging in a hierarchy, personal messaging was introduced to allow limited and controlled upward messaging as well. The personal messaging tab contains all the personal messages received and a floating button that allows the user to create a message to send. The navigation menu contains the link to different parts of the application offering the other features as well as certain activities that allow the user to create a group or edit his profile for example. The groups are classified as conditional or customized depending on the way they were created. A customized group has members that were individually selected. A conditional group has members that satisfy certain conditions that the creator of the group has set. If the profile has been edited, it has to be checked by the selected handler before the changes come into effect.

3. Features

Once a group has been selected, the UI shows a list of messages, the message layout based on its type. Fig. 4, the type can be - text, event, poll, or image. Other types can be supported as per the requirements of the organization. These types were specifically chosen because they can be called general types that any organization would *need*. Fig. 5, if a message is identified as an event then a notification is scheduled on the user's device based on the information contained in the message. A poll type of message is one wherein the user can give an answer to a question from a predetermined group of responses. Once answered, the poll results can be viewed in the form of a pie chart. Other types are self-explanatory. The organization-wide news is also supported in the application. A user has to get permission from his handler before sending news.

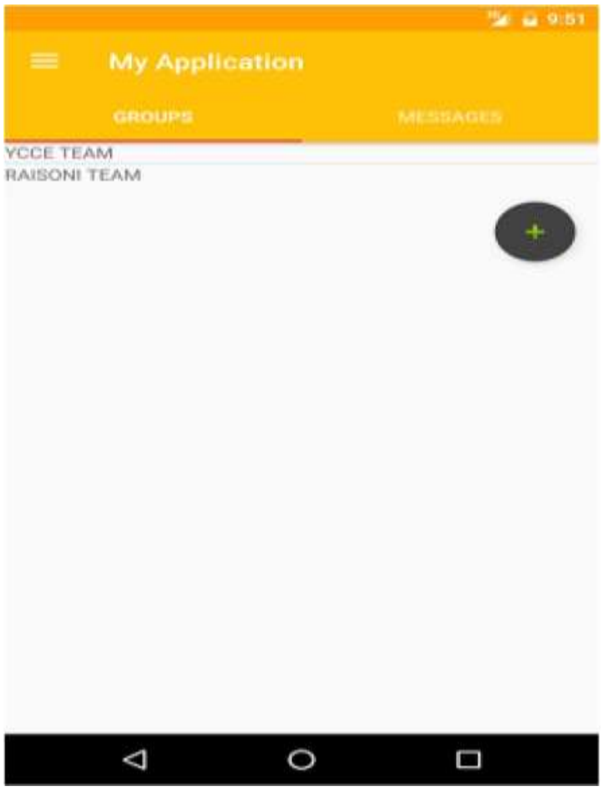

Figure 3. Home Page

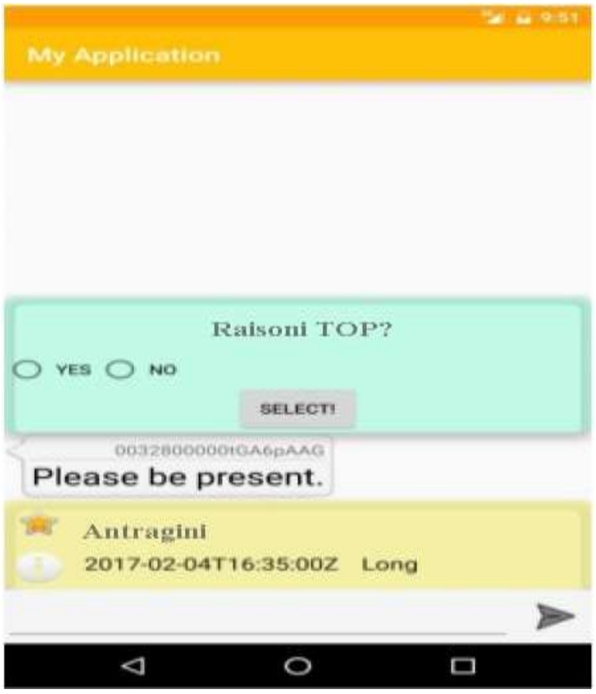

Figure 4. Different Feature Present In Group Messaging Like Polling & Events

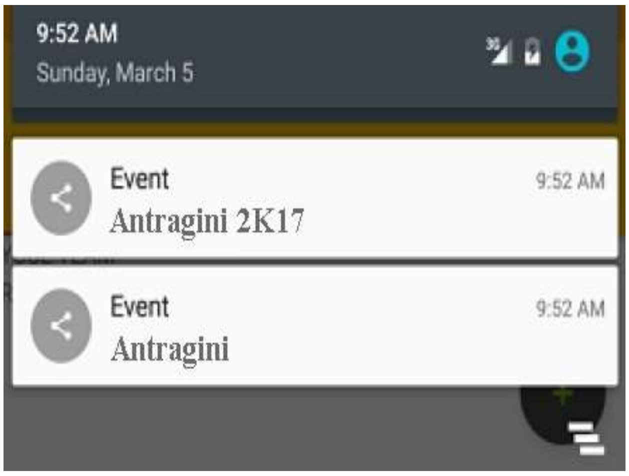

Figure 5. Feature Of Event To Set Notification At Defined Time

C. *Local Storage*

SQLite databases have been selected to store local data that needs to be retained even if the application is not running. It allows for the application to run correctly even if the user device is currently offline. Messages to be uploaded to Salesforce are first stored in SQLite database. They are then uploaded when the user connects to the internet. Images or files are stored in the user device's internal storage. A specific directory for our application is created where the files are stored so as to avoid clutter.

V. FUNCTIONS AND FEATURES

A. *Functions*

The following functions are performed by our application:

1. Data is synced between Salesforce and SQLite local storage.
2. Messages can be exchanged one-to-one.
3. Group messages can be read by all but only sent by some based on the requirements and structure of the organization it is deployed in.
4. Organization-wide news can be read by everyone.

5. Events are scheduled for the users.
6. Polls can be created within groups.
7. User registration and authentication are handled by the application itself.
8. Group can be created in one of two ways - conditional or custom. Custom is the general creation method where each individual group member is added. In a conditional group, a condition is set as the requirement for joining and if the user meets that condition then he/she is a member of that group.

B. *Features*

1. The hierarchical structure of the organization is maintained. The professional needs of the organization are met. Users can have limited interaction based on the roles that they serve within the organization. This makes the communication channel clutter-free. Important messages are preserved and are not lost in the barrage of pointless messages. Rights are decided on a group-by-group basis because a user can be limited in a particular context but have more power in the other. For this feature to work smoothly, the entire organization hierarchy in terms of the divisions needs to be mapped out in the objects in Salesforce.
2. Self-managing application. Since this is an infra-organization communication application, the users that join have to be authenticated. This would normally require some manpower on the cloud side that would handle this. However, we have implemented a novel solution that circumvents this requirement. A user that wants to join can create a profile but does not get access to the main application itself till the handler (another user that has already been authenticated) he/she has selected accepts the request to join. Once accepted, the user gets access to the application.

3. Content monitoring. In order to keep the communication professional, only certain users are given rights to post messages, news or events. This cuts down the clutter and risk while making particular users responsible. Also, groups that are created aren't operational until they have been judged to be acceptable by the creator's handler. Hence, the group creation is actually a request that needs to be approved.

VI. TYPICAL USAGE SCENARIOS AND FUTURE SCOPE

The application design has separated the hierarchy aspect from the core messaging application and can be used by any organization after an initial low-time, a low-cost setup based on the requirements. Traditionally, big companies have come up with such software internally for their own use dedicating precious resources.

Now, we live in an age where we are witnessing an increase in the number of start-ups that would be looking for such an application. They cannot afford to put in the time required and would, therefore, prefer an external solution. Educational institutions could also use this application to enhance the communication while maintaining the decorum expected in that institute. In brief, any organization/institute that follows a hierarchy can use this application if they desire.

Since this application has been built on the Android platform, it would be able to support all the future updates/improvements that Android developers release, if needed. The functions that the application servers have been created so that any general organization would find them useful but if the need arises then the application can include organization-specific functions because the design is nonrestrictive. Further ideas for improvement include but are not limited to - file sharing, local area network messaging, file repository on the cloud accessed by the application.

VII. CONCLUSION

In this paper, we have presented a messaging application that serves a specific purpose of catering to organizations rather than being general and geared towards the general public. Any organization with any kind of hierarchy can use this application with only one setup required on Salesforce that sets up some static data regarding the organization's structure. While the application has been termed as a messaging application, it offers much more than just that. Since it has been developed on Android, there is a lot of scope for future development improving the general features or adding some more based on the needs of the organization.

ACKNOWLEDGMENT

I would like to thank Prof. Prashant Borkar and Mr. Vedprakash Hirwani for constant support and guidance during application development.

VIII. REFERENCES

[1] Sungchul Lee1, Ju-Yeon Jo2, Yoohwan Kim3, “Method for Secure RESTful Web Service”, Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on Date of Conference:28 June-1 July 2015 INSPEC Accession Number: 15311886 DOI: 10.1109/ICIS.2015.71 66573.

[2] Raheleh B. Dilmaghani and Ramesh R. Rao, “A Systematic Approach to Improve Communication for Emergency Response”, Proceedings of the 42nd Hawaii International Conference on System Sciences – 978-0-7695-3450-3/09 $25.00 © 2009 IEEE

[3] D.D. Kouvatsos and I.M. Mkwawa, “Multicast communication in grid computing networks with background traffic”, IEE Proceedings online no. 20030810 doi: 10.1049/ip-sen:20030810 The authors are with the Department of Computing, University of Bradford Paper received 25th April 2003 IEE Proc.-Softw., Vol. 150, No. 4, August 2003

[4] E. T. Leal, O. Chiotti, and P. D. Villarreal, “Software Agents for Management Dynamic Inter- Organizational Collaborations,” IEEE LATIN AMERICA TRANSACTIONS, VOL. 12, NO. 2, MARCH 2014.

[5] Sikandar Ali, Siffat Ullah Khan, “Critical Success Factors for Software Outsourcing Partnership (SOP): A Systematic Literature Review”, 2014 IEEE 9th International Conference on Global Software Engineering 978-1-4799-4360-9/14 $31.00 © 2014 IEEE DOI 10.1109/ICGSE.2014.12

[6] Zhang Zhiying, Kang Kai, Wang Yunfeng, “The Investigation of Inter-organizational Information Systems Capabilities in HEBEI

Province, China”, 2009 World Congress on Computer Science and Information Engineering 978-0-7695-3507-4/08 $25.00 © 2008 IEEE DOI 10.1109/CSIE.2009.1013.

[7] Claudia Mathis, Bernhard Rinner, Martin Schmidt, “ new approach to model communication for mapping and scheduling DSP-applications”, Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference 0-7803-6293-4/001$10.0002000 IEEE.

[8] YUAN Hong, GU Ping-ping, “Application of Windows Interprocess Communication in Software System Integration”, 2010 International Conference on Intelligent System Design and Engineering Application 978-0-7695-4212-6/10 $26.00 © 2010 IEEE DOI 10.1109/ISDEA.2010.270

[9] Enter WANG, Xiang LI., The communication research of mobile phones and database server based on Socket. Technology and Development of Computer, vo17. 2,pp.82-84,2007 (In Chinese).

[10] Doc88.com, Cloud Computing of New Techniques of Computer Network.pp1-4, September,2010

[11] Jiahui HUANG, Java Network Program Design. Beijing: Tsinghua University Press,pp.30-49, 2002

[12] Sok H (2013). User Mobility in IEEE 802.11Network Environments. Thesis for B.E. in computing, Imperial College of London, 1-115

[13] Islam M. R, Islam M.R &Mazumder T.A (2010). Mobile application and its global impact. IJET, 10, 72–78.

[14] Mahajan A, Dahiya M.S &Sanghvi H.P (2013). Forensic Analysis of Instant Messenger Applications on Android Devices. IJCA, 68, 39–44.

[15] Shanmugapriya M &Tamilarasi A (2013). Design and development of mobile-assisted language learning (MALL) application for English language using Android PushNotification Services. IJRCCT, 2, 329–338.

[16] Capelli S, Ghetti A, Mora D &Mutti S (2008). An Eclipse framework to ease notification among MyUniBG app. Paper 2008.