SALESFORCE INTEGRATION WITH TELEGRAM

MAJOR PROJECT

Submitted by

ARADHY RANADE
186320010

KEWAL SHARMA
186301042

Under the Guidance of

MR. SUMIT BANSAL

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

FACULTY OF ENGINEERING & TECHNOLOGY

GURUKULA KANGRI (DEEMED TO BE) UNIVERSITY

2022

GURUKULA KANGRI (DEEMED TO BE) UNIVERSITY

BONAFIDE CERTIFICATE

Certified that this project report **SALESFORCE INTEGRATION WITH TELEGRAM** is the bonafide work of **ARADHY RANADE** & **KEWAL SHARMA** who carried out the project work under my supervision.

SIGNATURE

DR. MAYANK AGARWAL

HEAD OF THE DEPARTMENT

Department Of Computer Science & Engineering

Faculty Of Engineering & Technology

Gurukula Kangri (Deemed To Be) University

SIGNATURE

MR. SUMIT BANSAL

SUPERVISOR

Department Of Computer Science & Engineering

Faculty Of Engineering & Technology

Gurukula Kangri (Deemed To Be) University

DISCLAIMER

This project, **SALESFORCE INTEGRATION WITH TELEGRAM** report/ dissertation and completely functional project has been prepared by the author under the Major Project of the Faculty of Engineering & Technology, Gurukula Kangri (deemed to be) University, for academic purposes only. The views expressed in the report are personal to the students and do not necessarily reflect the view of the FET, GKV or any of its staff or personnel and do not bind the Faculty of Engineering and Technology, Gurukula Kangri (deemed to be) University and the same or any part thereof may not be used in any manner whatsoever, without express permission of the Faculty of Engineering and Technology, Gurukula Kangri Vishwavidyalaya, in writing.

Yours Student

Aradhy Ranade

Kewal Sharma

ACKNOWLEDGEMENT

We would like to express our gratitude to **Mr Sumit Bansal,** Department of Computer Science Engineering for guidance and support throughout this project work. He has been a constant source of inspiration to us throughout the period of this work. We consider ourselves extremely fortunate for having the opportunity to learn and work under his supervision over the entire period.

Yours Student

Aradhy Ranade

Kewal Sharma

ABSTRACT

Salesforce provides functionality to generate leads with three standard methods:

1. Manual Lead generation through the User Interface
2. Lead generation through the web forms.
3. Lead generation through the email.

But the requirements are not always met with the standard functionality. So there is another method through which we can achieve all the requirements.

Salesforce supports OpenAPI standards through which it can be integrated with any of the platforms which also provides API support. Through API, leads can be generated via any of the platforms, like Telegram, Facebook, Twilio, Messenger, Twitter, and so on. In this project, Salesforce is being integrated with Telegram to generate leads and to include various business processes like sharing documents autonomously, sending various autonomous detail messages depending on the stage of the deal, etc.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	FULL FORM / MEANING
ORGS	Organisation
DML	Data Manipulation Language
CRM	Customer Management System
RegEx	Regular Expressions
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
UI	User Interface
HTML	Hypertext Markup Language
DOM	Document Object Model
CSS	Cascading Style Sheet
OpenAPI	Publicly Accessible Api
JSON	Javascript Object Notation
REST	Representational State
URI	Uniform Resource Identifier
UNIX	Uniplexed Information Computing System
ASCII	American Standard Code For Information Interchange
PDF	Portable Document Format
Fig	Figure

CHAPTER 0

Problem Description

Problem Statement

Currently in salesforce the generation and operation of lead is very slow. There are very limited options to generate a lead or create any object. Salesforce provides creation of lead either by mail or web form or by manual insertion. Also the ways to interact/communicate with the lead or person associated with the objects created via email or form are limited. The salesforce user can only do it by email, which is a very slow process and is not real time as the user needs to refresh the UI to get new mail by telegram user.

Solution Proposed

The solution proposed by us is rather simple. We are proposing that using the salesforce powerful APIs we can integrate it to other third party applications. These applications are real time, faster and responsive then salesforce. Using them with salesforce can speed up the work process, enhance the communication and provide automation to limited salesforce.

Here in this project we demonstrate the integration of salesforce with third party applications. We will be using telegram API for faster lead creation and faster communication by a real time chat system. We will also include automated messages that will be sended via salesforce to telegram.

CHAPTER 1

Introduction to Salesforce

What Is Salesforce?

Salesforce, Inc. is a famous American cloud-based software company that provides CRM (Customer Relationship Management) services. Salesforce is a popular CRM tool for support, sales, and marketing teams worldwide.

Salesforce services allow businesses to use cloud technology to better connect with partners, customers, and potential customers. Using the Salesforce CRM, companies can track customer activity, market to customers, and many more services.

A CRM platform helps you go deeper with all your metrics and data; you could also set up a dashboard that showcases your data visually. In addition to this, you can also have personalised outreach with automation. Another significant benefit is that a CRM platform can also improve customer service's ability to help customers or a sales team's outreach efforts.

Salesforce Architecture

Salesforce have three different layer in architecture

1. Multi-tenant: Salesforce stores data in a single database schema. There can be a single instance of a software server with multiple tenants. Speaking about a multi-tenant architecture, there is a single shared application service to several clients. This makes it cost-effective.
2. Metadata: Salesforce uses a metadata-driven development model. This allows developers to only focus on building the application. This metadata-driven platform makes customization and scaling up easy.

3. API: Salesforce provides a powerful source of APIs. This helps in developing and customising the Salesforce cloud application.

Standard Database Structure

In a standard database, we have 2 major entities: Database itself and tables that are part of the database which actually contain the data.

- Databases can contain multiple tables. Each table must have a unique identifier.
- Every table must contain at least 1 column.
- Each column has a specific data type.
- A row in a table represents a record.
- Each Row is a collection of columns representing the attributes of that record.

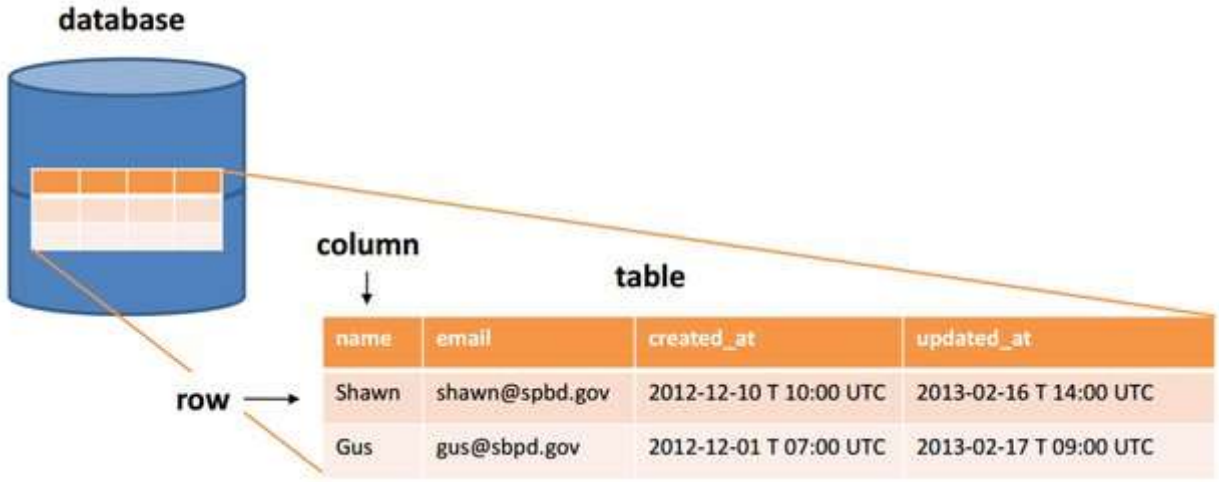

Fig 1.1: Standard Database Structure

Salesforce Database Structure

In salesforce also the data organisation is stored in a similar structure but with different terminologies mentioned below:

1. Fields : A field represents a column of a table in a standard database. It holds a value of a specific type.
2. Record : A record represents a row of a table in a standard database. It is a collection of fields.
3. Object: An object represents a table in a standard database. It is a collection of records of the same type.
4. Salesforce Orgs : This represents the database in the standard database. It is a collection of all the objects in the database.
5. Salesforce Applications: It is a group of specific objects which are linked logically to each other.

Fig 1.2: Salesforce Database Structure

Salesforce Sales Cloud

Sales cloud is a type of salesforce application that contains a group of standard objects that are provided by salesforce. These objects are generally sales oriented and used to store sales related data and to carry operations on it, like converting interested prospects into customers and storing the details of the deals that have been closed already or will be closed in future.

For all these operation salesforce provide us with some standard objects, some of them are as follows:

1. **Lead**: This the first stage through which any prospect will enter into the system or org. Any interested customer that is interested in our services or product is called lead in salesforce terminology.
   This object contains all the information about the intended customer like the source from which it is captured, product and services in which it is interested, probability of successful business etc. If the intended customer is ready to buy the services then it will be converted to **Account**, **Contact** and **Opportunity**.
2. **Account**: This object stores the information of the company or the organisation of the customer which is now our **Contact**. This **account** object will serve as a parent record for the newly created **contact** and **opportunity** records.
3. **Contact**: After conversion the intended customer which was our **lead** is now converted to **contact**. It inherits all the fields from the **lead** object.
4. **Opportunity**: This is the actual object which contains all the information about the deal with the account. This includes the deal amount, current stage of the deal, all the products, quotes, progress etc. The opportunity can only have two possible outcomes: Closed Won or Closed Lost. After closing the opportunity no more changes can be done to the record.

5. **Product**: This object contains the information about the products that are sold by our Organisation, this include base price and name of the product.

6. **Opportunity Line Item**: This object contains the information about the products that will be added to the **opportunity**. **Opportunity Line Item** record contains the information like quantity, price of product for the **opportunity**. The total amount of the opportunity will be calculated on the basis of opportunity line item only.

7. **Quote Line Items**: This object is similar to the **Opportunity Line Item**, other than that of the parent object. **Quote Line Item** can only be added to the **Quote**.

8. **Quote**: It is an object which is basically an invoice of the **opportunity**. To make an invoice we first make a **Quote** under that **opportunity**, then add the **Quote Line Items** to this quote. If this is the actual invoice and the products of this **quote** are the only products which are in the actual deal, then this **quote** can be synced to the **opportunity**. After that any product which will be added to this **synced quote** will also be added to the **Opportunity** as an **Opportunity Line Item**.

   Quote can also be rendered as a pdf file which contains all the information of this quote.

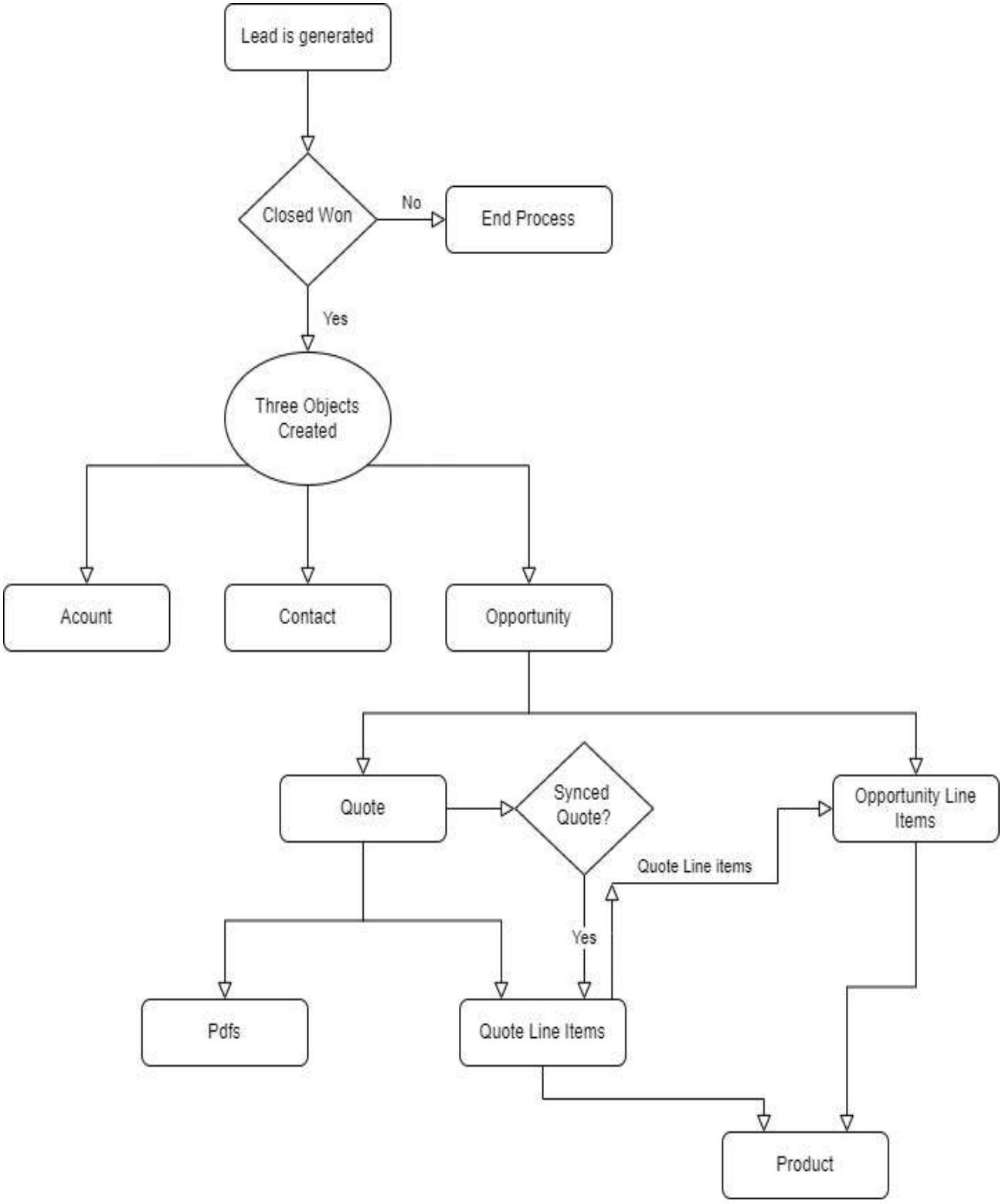

Fig 1.3: Salesforce Sales Process

CHAPTER 2

Introduction to Apex

What Is Apex?

Apex is a proprietary language developed by Salesforce.com. As per the official definition, Apex is a strongly typed, object-oriented programming language that allows developers to execute the flow and transaction control statements on the Force.com platform server in conjunction with calls to the Force.com API.

It has a Java-like syntax and acts like database stored procedures. It enables the developers to add business logic to most system events, including button clicks, related record updates, and Visualforce pages. Apex code can be initiated by Web service requests and from triggers on objects. Apex is included in Performance Edition, Unlimited Edition, Enterprise Edition, and Developer Edition.

Type of Apex

There are two type of Apex code that a salesforce developer can work on that are:

1. Asynchronous Apex

   Asynchronous Apex is **used to run processes in a separate thread at a later time**. It is a process or function that executes a task "in the background" without the user having to wait for the task to finish.

   These code are of 4 types:

   i. Batch Apex: In this the data on which a code has to be executed will be divided into batches. These batches will run separately. This code is used to process a large amount of data.

   ii. Future Method: In this the method is defined as the future method. These methods will be run by salesforce when the resources are available.

   iii. Queueable Apex: In this a certain block of code can be run after the execution of a code that is placed in queue.

iv. Schedulable Apex: In this we can schedule a block of code so that it will run on the defined time. We can schedule a class by using CRON string or by SETUP in org.

2. Synchronous Apex:

   Synchronous Apex means the entire Apex code is executed in one single thread. For example, when we call a method of any class in our code, the statements preceding this method call have to wait for the time this method finishes its execution. After the method finishes its execution, the preceding statements will then be executed.

   That is used when we want to run a logic on button click or certain action. For this salesforce provide **Triggers.**

Triggers

A **Trigger** is an **Apex script** that executes before or after data manipulation language (**DML**) events occur. Apex triggers enable you to perform custom actions before or after events to record in Salesforce, such as insertions, updates, or deletions. Just like database systems support triggers, Apex provides trigger support for managing records.

Trigger events in salesforce

A trigger is a set of statements which can be executed on the following events. In above trigger events one or more of below events can be used with comma-separated.

Here is a list of trigger events in salesforce

- before insert
- before update
- before delete
- after insert
- after update
- after delete
- after undelete

Types of Triggers

There are two types of triggers:

- **Before triggers** are used to perform a task before a record is inserted or updated or deleted. These are used to update or validate record values before they are saved to the database.
- **After triggers** are used if we want to use the information set by the Salesforce system and to make changes in the other records. are used to access field values that are set by the system (such as a record's Id or LastModifiedDate field), and to affect changes in other records. The records that fire the *after trigger* are read-only.

Implementing the Triggers

Consider the following before implementing the triggers.

- Upsert trigger fires on 4 different events :- before(insert, update), after (insert, update)
- Merge trigger are fired on both events on delete
- Field history is updated after the trigger has successfully finished processing data.
- Any callout should be asynchronous so that trigger does not have to wait for the response.
- A trigger cannot have a static keyword in its code.
- If a trigger completes successfully the changes are committed to the database and if it fails the transaction is rolled back.

Lightning Component

Lightning includes the Lightning Component Framework and some exciting tools for developers. Lightning makes it easier to build responsive applications for any device. Lightning includes these technologies:

- Lightning components accelerate development and app performance. Develop custom components that other developers and admins can use as reusable building blocks to customise Lightning Experience and the Salesforce mobile app.
- Lightning App Builder empowers admins to build Lightning pages visually, without code, using off-the-shelf and custom-built Lightning components. Make your Lightning components available in the Lightning App Builder so administrators can build custom user interfaces without code.
- Experience Builder empowers admins to build communities visually, without code, using Lightning templates and components. Make your Lightning components available in Experience Builder so administrators can build community pages without code.

Using these technologies, you can seamlessly customise and easily deploy new apps to mobile devices running Salesforce. In fact, the Salesforce mobile app and Salesforce Lightning Experience are built with Lightning components.

Aura Component

Aura components are the self-contained and reusable units of an app. They represent a reusable section of the UI, and can range in granularity from a single line of text to an entire app.

The framework includes a set of prebuilt components. For example, components that come with the Lightning Design System styling are available in the lightning namespace. These components are also known as the base Lightning components. You can assemble and configure components to form new components in an app. Components are rendered to produce HTML DOM elements within the browser.

A component can contain other components, as well as HTML, CSS, JavaScript, or any other Web-enabled code. This enables you to build apps with sophisticated UIs.

The details of a component's implementation are encapsulated. This allows the consumer of a component to focus on building their app, while the component author can innovate and make changes without breaking consumers. You configure components by setting the named attributes that they expose in their definition. Components interact with their environment by listening to or publishing events.

Chapter 3

Introduction to Telegram API

Telegram is an open source, multi-platform, online chatting platform. It sends a message from a sender to a recipient using the internet. It also supports the bot accounts which are a small piece of code which can send and receive messages programmatically.

Telegram Bot API

Telegram Bot API is the OpenAPI for the bot platform of the telegram. There are two methods through which we can retrieve messages from the telegram bot:

a. **Long polling method**

   Whenever any message is received by the bot, this message will be added to the *updates* object of the bot. This message will have to be retrieved by the user manually by hitting the bot api using the `getUpdates` method (mentioned below). As we don't know when the message can come, we have to continuously check for the new messages using long polling[1]. Telegram will give a response with the list of the new messages received since the last request.

b. **Webhook method**

   Whenever any message is received by the bot, then it will send a POST request to the subscribed webhook[2]. This webhook needs to be available for global access, and the bot needs to be subscribed manually to this webhook. This webook should be able to handle the POST request sent by the bot and the JSON body of this request.

[1] long polling: Hit the api continuously to get updates. These requests are separated by a small time interval in between.
[2] webhook: A public piece of code which contains the different methods which can be called using REST api.

URI of the bot api has following format:

`https://api.telegram.org/bot<bot_api_token>/<method>`

Where,

- *`<bot_api_token>`* is the api token of the telegram bot. It is a unique token given by the bot father when the bot was created. It is used to control the bot through the API.
- *`<method>`* is the one of the method available to use

1. `sendMessage`

   This method is used to send messages to the user via bot. It is a POST method which requires a json body. Body requires at least two parameters: **chat_id**, **text**.

2. `sendDocument`

   This is also a POST method used to send any file document. It's body must contain following fields:

   a. `chat_id -` unique identifier of the telegram chat

   b. `document -` file to be sent to the user. File should be either in multipart/form-data format or a public http url of the file.

3. `setWebhook?url={webhook_url}`

   This is a GET method used to subscribe the bot to a webhook. `{webhook_url}` is the public url of the webhook to which the bot is being subscribed.

4. `getWebhookInfo`

   This is a GET method used to get the info of the updates pending and the status of the webhook which is subscribed to the bot.

CHAPTER 4

Salesforce Admin Implementation

Custom Object and Fields

Custom objects and fields are the fields that are created by the developer to fulfil the custom requirement. For this project we will create some custom objects and fields that we will use to store some information.

Custom object created for saving the history of chat with our lead. We named this object **TelegramChat**

In this object we will create some files for storing chat related data. Those fields are:

Chat: This will store the chat with the lead.

ChatSeprator: This contain a special separator that separator messages from one another

TelegramChat Name: This will store the name of the user or lead.

Telegram User Id: This will store the unique telegram chat id created when we start a chat.

In this for a particular lead we will store the whole chat in a single field, where every message will be separated by a special separator stored in Chat Separator. Each message stored in JSON format will have some parameters that will have information about the message like time, message content, document link etc.

Whenever a message comes:

Let us assume the chat field has a chat history as ‘A’.

A new message is coming, let it be ‘n’.

We will convert that message in json. So ‘n’ => ‘N’.

Now we will add this message to our chat history as ‘N’ + separator + ’A’.

We also created some custom fields on standard object Lead and Opportunity names as **Telegram User Id**. This field will store the unique chat id that we will use to map it with our lead and opportunity records and identify it on telegram.
Telegram Template is also a custom field that we make on lead and opportunity that contains the message template of the automated messages that has to be sent to the customer when they are created.

Public site

Slaesforce provides us with the feature of a public site that allows us to take input from guest users, making our Org secure. This public site also allows us to host webhooks that can be used by other APIs to send data to salesforce.
For this project we created a public site with a label **telegram.** This public site is used to host a webhook that is used by telegram bot api to send messages to salesforce. We need to give public access of the webhook to the public site so that it can give access to the bot publicly.

Custom Labels

Custom labels enable developers to create multilingual applications by presenting information (for example, help text or error messages) to users in their native language. A custom label is a custom text value that can be accessed from Apex classes, Visualforce pages, and Lightning components.
We created a custom label called **Telegram bot api key** that will store the Bot API key, so that it can be easily accessed in code later.

Platform Event

Use platform events to connect business processes in Salesforce and external apps through the exchange of real-time event data. Platform events are secure and scalable messages that contain data. Publishers publish event messages that subscribers receive in real time. To customise the data published, define platform event fields.

Salesforce has a special class to publish the platform events **EventBus** which has methods publish method. Once the event is published you can consume the events from the channel.

Here in this project we are creating a platform event for notifying our system about the incoming message from telegram. This platform event is called **ChatUpdated**. This event is published by the webhook after our chat history gets updated. The event is subscribed to our telegram aura component that will react accordingly when the event is generated.

CHAPTER 5

Salesforce Apex Implementation

Telegram Webhook

Telegram Webhook is a custom REST api which is hosted on the public site on the salesforce platform itself. This webhook will be subscribed by the telegram bot.

So whenever the bot receives any telegram message, it will send a POST request to this webhook.

This request contains following JSON in its body:

```
{
    "update_id": 16378737,
    "message": {
        "message_id": 90,
        "from": {
            "id": 729858641,
            "is_bot": false,
            "first_name": "Kewal",
            "username": "sayKewal",
            "language_code": "en"
        },
        "chat": {
            "id": 729858641,
            "first_name": "Kewal",
            "username": "sayKewal",
            "type": "private"
        },
        "date": 1653756122,
        "text": "hii"
    }
}
```

- `message`: message object which contains the actual message details mentioned below.
    - `text`: message text.
    - `from`: the details of the user from which this message was received.
    - `chat`: the details of the chat from which this message was received. This object contains an id field which uniquely identifies the chat from which this message was received.
    - `date`: the date of the message in UNIX datetime format.
    - `document`: the details of the document object if the message contains any document.
    - `photo`: list of the photo objects which contain different sizes of the image in the message.
    - `video`: the details of the video object if the message contains video.

This webhook has to parse this request and have to perform these three operations:

1. Create a lead if the message is received from a new user. While creating this lead, the Telegram User Id field will be populated with the `message.chat.id` from the above JSON. After the creation of the lead, the acknowledgement message will also be sent to the user.
2. Save a copy of this message to the salesforce database.
3. Publish a platform event so that the chat U.I. can know that a new message is received by the system.

Trigger

Lead creation/conversion: Whenever a lead is created by Telegram Webhook a trigger will be run on Lead. This trigger will ensure that the new lead/ intended customer will get an acknowledgment about the lead creation. The lead that is generated by this will be assigned to the integrated user that will handle the new lead.

The trigger will also run on the lead conversion. This trigger will ensure that the lead now the contact should get confirmation that it is converted to a contact and an opportunity is created with his account.

The message that is sent to the customer can contain dynamic elements like lead name or lead source, these fields are generally called merge fields in salesforce terminology. Before sending the message these merge fields are replaced by the actual data that we need to send.

Opportunity Closed Won: A trigger will run on the update of opportunity with condition that it is closed and won at this point the contact will get a acknowledgement that the deal is closed and if there is a synced quote with the particular opportunity that is closed won then that will be attached with the message and send to the customer.

Telegram Utility

Telegram utility is an apex class that is used to send different kinds of messages to lead or contacts or opportunity. This message includes the normal text messages or the file attachments. This utility is used by backend autonomous calls from the trigger to send the template messages when a lead is created or converted and when an opportunity is closed won.

The telegram utility class is consist of two methods:

1. **Sending Text Message**: The telegram utility class sends text messages with the help of `sendMessage.` This method required two parameters: `chatId` and `message` string to send the data. This method uses a http request to send data to telegram by which the customer gets the message by salesforce.
2. **Sending Attachment**: The telegram utility class uses `sendAttachment` method to send any attachment to the customer via telegram. This function takes four parameters: chatId, filename, mimptype, fileContent. This uses multipart/form-data to send the file via HTTP request to telegram.

Multipart/form data generally contains all the data and information about the types, size, name of the file; this is equal to uploading a file via http request. The encoding process is performed before data is sent to the server as spaces are converted to (+) symbol and non-alphanumeric characters or special characters are converted to hexadecimal (**0-9, A-F**) values as the ASCII character set is the format for sending data on the Internet. So, the real purpose of encoding is to make the data in a standard format so that it can be sent on the Internet.

Merge Field

Merger field is another apex class that focuses on inserting data in the template that are stored in lead and opportunity. This is done with the use of RegEx. The main motive of the class is to recognize the field name from the object and replace that from the value stored in that field. The process is simple: we pass record id and the template to the class. The class first identifies the fields that are in the template. After identifying the field it will create a dynamic query to fetch the data. Once the data is fetched the class will replace the data with the field name and will return that template.

Http Form Builder

HTTP Form builder is an Apex class that is used by the telegram utility class. This class is used to create the multipart/form-data to to upload the files such as images PDFs and the files of other types to create the form data. The class requires all the information about the file that is uploaded via the UI or all of the data required is retrieved from the content version of the salesforce.

CHAPTER 6

Salesforce Aura Implementation

Telegram Chat UI

Fig 6.1 shows the Telegram chat UI in salesforce. This chat UI consist of three major part:

1. Chat message Input:

   The chat message input takes the input from the salesforce user and delivers it to the customer telegram via a HTTP request to telegram.

2. Upload File Component:

   The File Upload component allows the salesforce user to select file(s) and send them to the telegram user. **Clear** option will be visible once any file is selected along with the number of files selected in the label (Fig

3. Chat window:

   The Chat window shows the history and all the messages received and sent by the salesforce user.

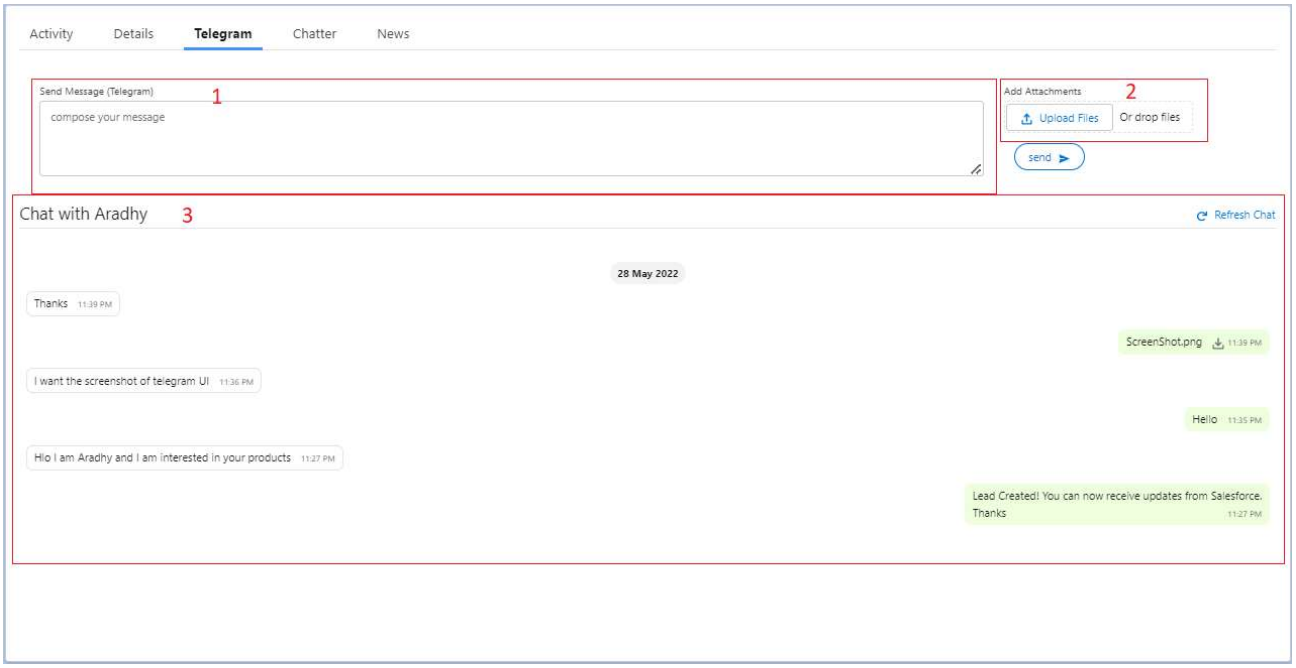

Fig 6.1: Telegram UI in salesforce

Send Message

Send message part of the UI is responsible for sending messages and attachment to the telegram user via telegram bot. The message is sent to the telegram user by clicking on the send button.

We can send messages by clicking on the send button. It can also be sent by pressing ENTER on the keyboard.

This input field supports multiple lines. To move to the next line press SHIFT+ENTER. The message API is called via JavaScript and the API is the same which is used by the Telegram Utility class.

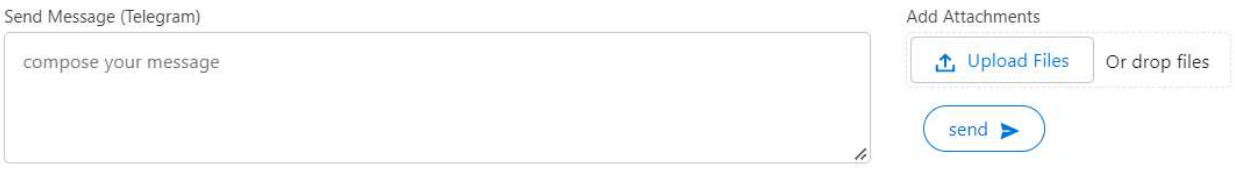

Fig 6.2: Send Message UI

Fig 6.2 shows the send message part of the UI as it includes the input box, send button, and upload attachment component.

Fig 6.3: Enter Text for the message

Fig 6.3 shows the input section of the UI.

Send Attachment

The Send attachment part of the UI is used to upload files and send it to the telegram user. We used Lightning Input. This helps us to select files from our system. Fig 6.4 shows us the window that pops up when we click on the Upload File button.

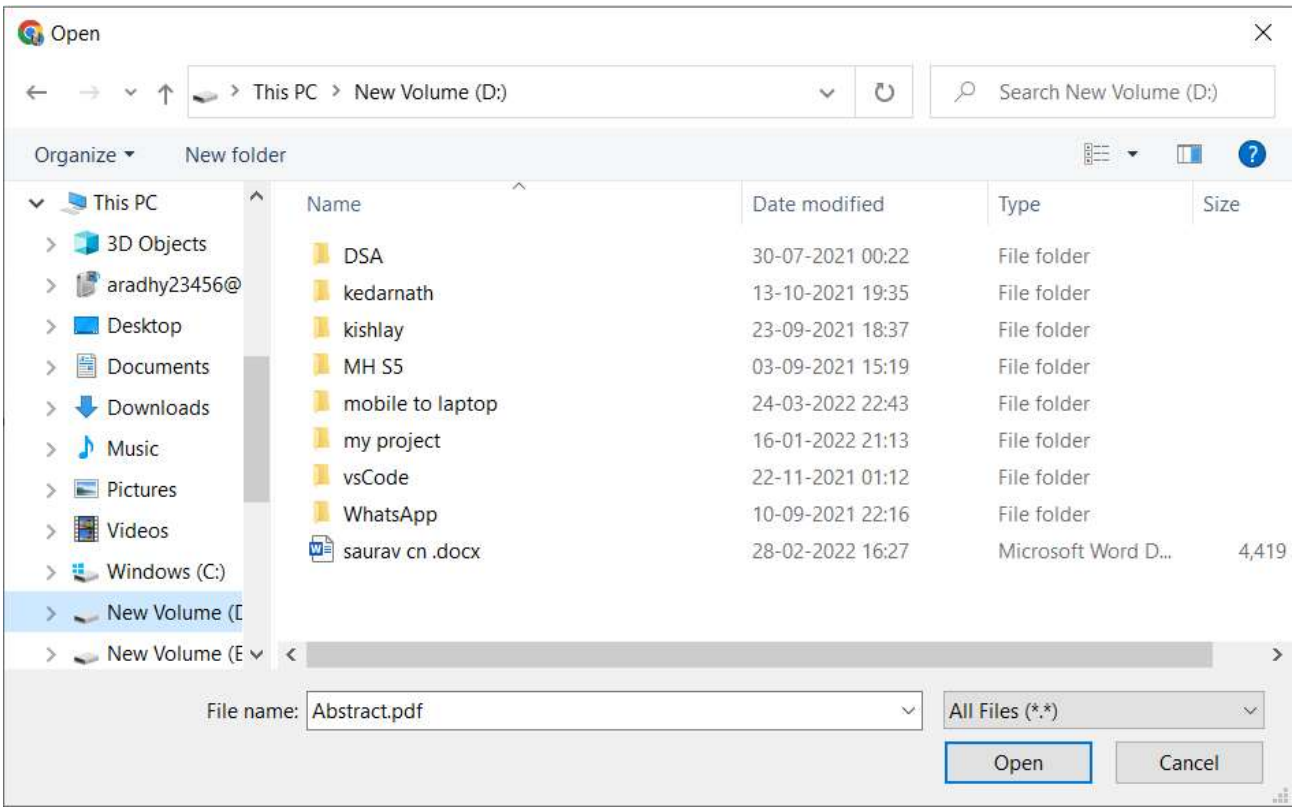

Fig 6.4: Select Attachment UI

Once the file is selected, it reaches the javascript and gets stored in an array in javascript. Also after selecting the file we will get an option for clearing all the files that are uploaded. We can send the file to the telegram user by clicking on the send button or by pressing enter in the input box.

Fig 6.5: Attachment Selector UI after selecting attachment

Chat Window

Chat window is the part of the UI that shows the real time chat to salesforce users. This window shows new messages at the top. The UI we created is the same as the telegram UI, received messages are at left and sent messages are at right. The chat window like telegram divides the chat by date. It shows a chat of a respected date under the label of that date. We can access the rest of the chat by scrolling down.

Fig 6.6 shows the UI of the chat window which is shown to salesforce users for real time chat.

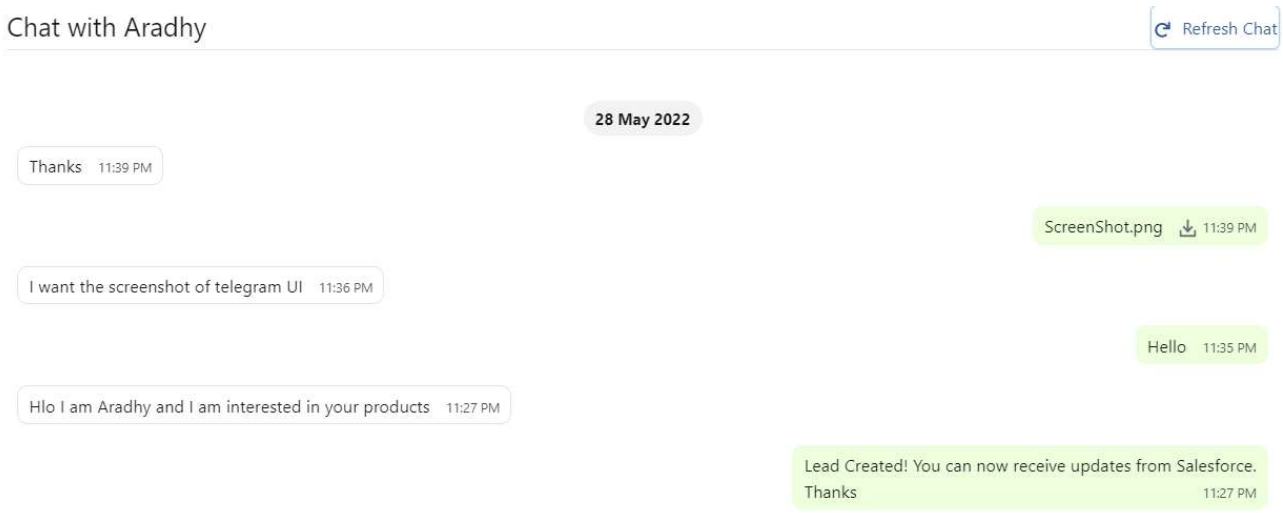

Fig 6.6: Telegram Chat UI from the salesforce user Perspective

In Fig 6.6 we can also see a sample chat where we have sent some messages and images to our Lead.

Fig 6.7: Telegram Messages in the Android Telegram Client from customer Perspective

Fig 6.6 shows the example of real time chat with our Telegram Bot. You can see the sent image in Fig 6.5 here.

CHAPTER 7

Working and Design

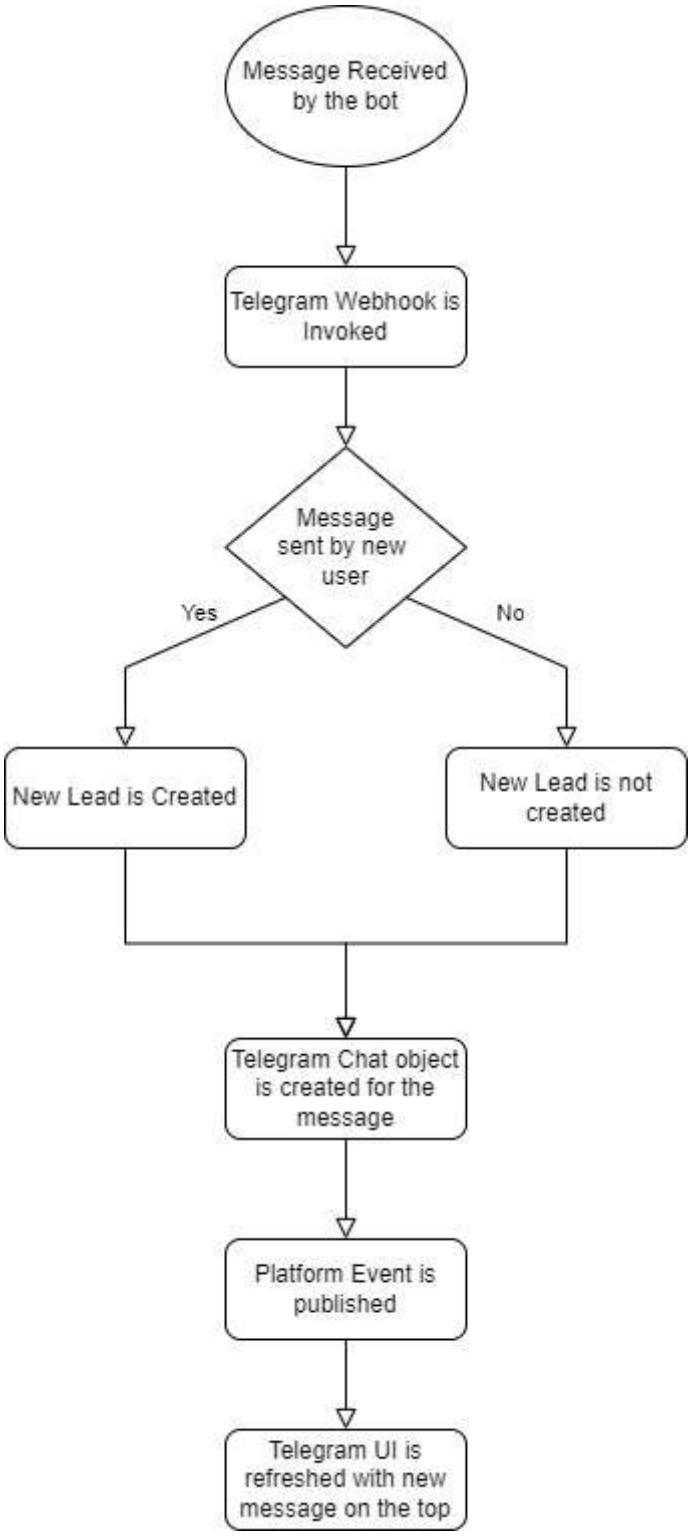

Fig 7.1: Structure of project when message is received by the salesforce user

In this project there are 2 major scenarios, to understand the working we will understand both of the scenarios.

Message is Received

In this scenario we have to handle the incoming message that is sent by telegram. This is handled by the following process:

- First when the client sends a message to our bot the bot initiates a sequence and sends the message to salesforce by hitting the public webhook. In that the telegram sends the message in JSON format with message text along with the information like time, chatId etc.
- Once the message is received by the webhook in salesforce it will check the chatId in the message JSON and search it in the telegram User id fields of lead and opportunity. If it is found in any lead, then it will simply update the chat history of the lead and if not then it will create a new lead using the information sent by telegram and creates a new Telegram Chat record
- Once the lead is created or history is updated the webhook creates a platform event that is subscribed by our UI. This event is published with the help of Event Publish Bus class provided by salesforce.
- Once the UI gets the notification via the platform event it will go to the chat history of the lead and update the chat window accordingly in the chat window section.
- For every record there is a different instance of UI, so every time a platform event is published all the UI will update their chat window simultaneously.

Fig 7.1 shows the above mentioned steps in a flow for better understanding.

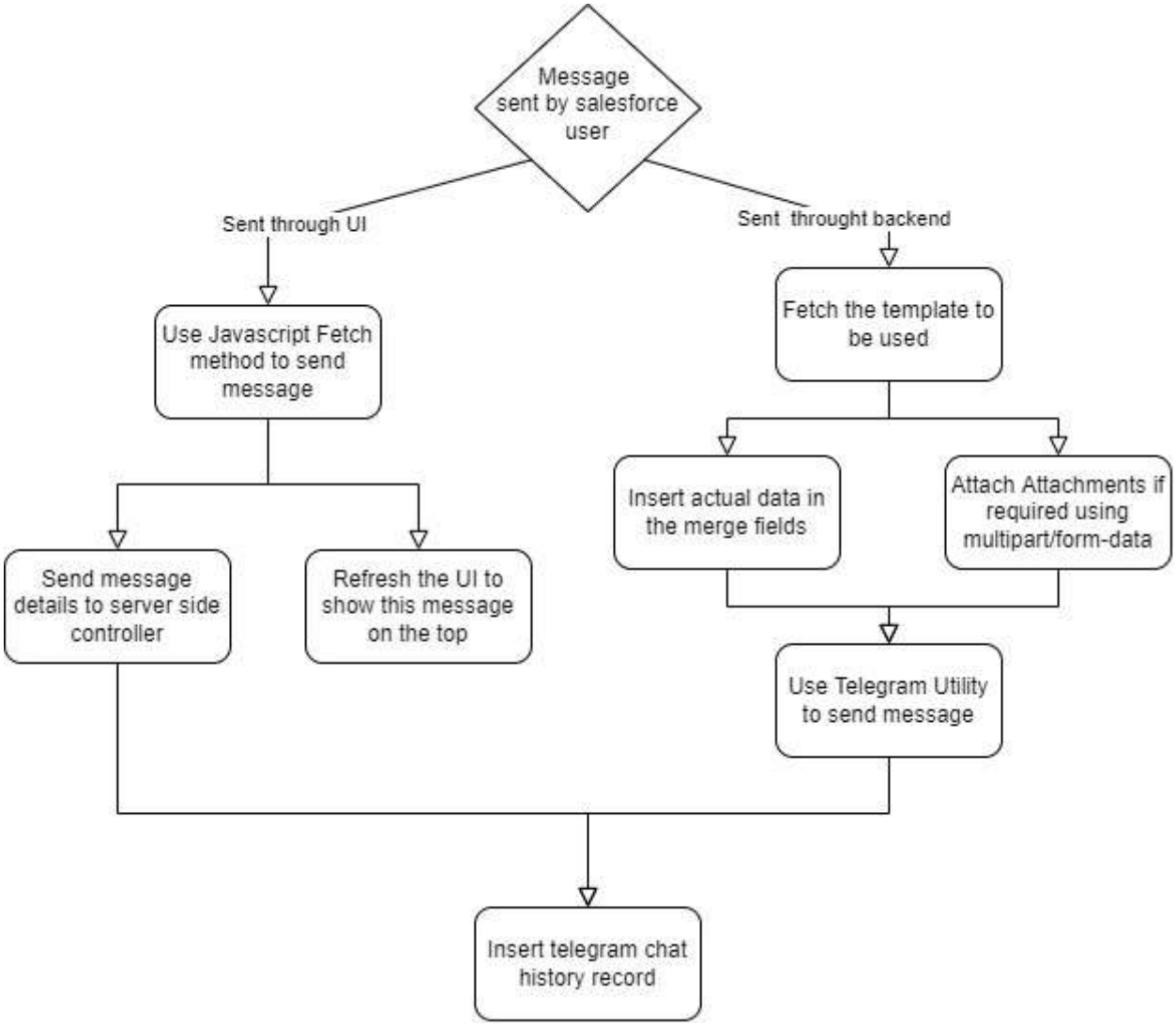

Fig 7.2: Structure of project when the message is sent to the telegram user

Message is Sent

In this scenario we handle the message and attachment sent to telegram via UI or automation. This is handled by the following process:

- If the message is send via UI then:
    - First the message is captured by the UI, with the help of the Input field.
    - Once it's done the message is sent to telegram by using the fetch method.
    - The message is sent using the Telegram User Id that is fetched from the record by the UI at the time of initialization.
    - Once the message is sent successfully then the JS sends the information about the message to the server side control.

    - The information reached to the server side control and using the Telegram user Id or the chat Id the telegram history gets updated.
    - Once the history update is completed then the UI chat window gets updated with a new message at the top. Assuring salesforce users of successful delivery of messages.
- If the message is generated via automation:
    - First this scenario occurs when a lead is created via webhook, or lead is converted to opportunity or the opportunity is updated to closed won.
    - When this scenario occurs then the trigger on these objects gets invoked.
    - The trigger fetches the telegram template stored in the record.
    - Once the template is fetched then the merge fields are updated via merge class.
    - When we get the actual data we need to send to telegram then we do that using telegram utility.
    - If a file/PDF is needed to be sent then this is identified and done by creating a multipart/form data and using telegram utility class the file is sended to the telegram.
    - Once the message/file is sended to the telegram the chat history related to the lead or opportunity gets updated with the help of unique chat Id stored in them.

Fig 7.2 shows the working of this scenario in simple flow.

These two scenarios working in sync creates a required real time chat system with telegram that to some extent can send the messages by its own through triggers and is able to receive and update the database without refreshing the page.

CHAPTER 8

Conclusion & Limitations

The concussion of this project report is, we developed a real time telegram Chat with salesforce. We solved the problem of lead generation by limited and standard methods. We also see that a faster way to communicate with the lead or customer is possible by integrating salesforce with third party applications like telegram.

Limitations:

- Animated sticker

  Users can not send or receive animated stickers from telegram to salesforce. As in Aura the support for Sticker is not present.

- Polls

  We can not have, see or participate in polls in telegram which is one the major features.

  Aura is not able to show and interact with the poll in telegrams.

- Limited chat history

  The chat history that we are saving is limited by the size of the Org. We can not have unlimited chat history, after some time we need to buy storage to store chats.

- Privacy oriented

  In this the application is not privacy oriented means that if a user deletes the chat from his side the chat saved in salesforce will not be deleted, and there is no indication by telegram to notify salesforce of chat deletion.

- Responsive

  The chat UI or the salesforce can not be reactive as a mobile application of telegram. The aura component uses a heavy Aura framework to render that causes lag sometimes.

- Attachment

  Apex does not have support for all types of file as for now we have to send the file in document form.

CHAPTER 9

Future Scope

Due to time limitations and resource limits this project is done to the point where it can be presented to the client and it is acceptable but there certain improvement and functionality that can be added implemented in same project, some of them are mentioned below:

- UI

  UI can be further developed to be more reactive to user input. It can be lag free.

- File selection and removal

  The file selection right now is limited, in future we can show the name of each file that has been uploaded to the telegram and we can select a file to remove rather than removing all files.

- Integration with more platform

  Like telegram salesforce can be integrated with other platforms like Messenger , WhatsApp etc. as new ways to generate leads or with other business processes.

- Integration with more business processes.

  The telegram-like lead can be used to generate cases and other objects to follow other business processes.

- Can add prompts to get more detail

  In telegram automation we can add prompts so that the user can select the business process in which he/she is interested.

- Chat window improvement

  The chat window can be made more like the actual telegram or chats.

- Host on own server

  Rather then using telegram to store files and message we can use our own server to store files and message that will remove the limitation imposed by telegram

References

- Salesforce Documentation
  https://developer.salesforce.com/docs
- Telegram API Documentation
  https://core.telegram.org/bots/api
- Lightning Component Library
  https://developer.salesforce.com/docs/component-library/overview/components
- Lightning Design System
  https://www.lightningdesignsystem.com/
- Apex Developer Guide
  https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_dev_guide.htm
- TrailHead
  https://trailhead.salesforce.com/en