MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

Project Report

on

EMAIL-NOTIFICATION SYSTEM USING MULESOFT

Submitted By:

PALAK GUPTA

0901CS191072

AMAN DAYANI

0901CS191014

Faculty Mentor:

Mr. Mir Shahnawaz Ahmad

Asst. Prof.

CSE, MITS

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE

GWALIOR - 474005 (MP) est. 1957

MAY-JUNE 2022

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

Project Report

on

EMAIL-NOTIFICATION SYSTEM USING MULESOFT

A project report submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

PALAK GUPTA

0901CS191072

AMAN DAYANI

0901cs191014

Faculty Mentor:

Mr. Mir Shahnawaz Ahmad

Asst. Prof.

CSE, MITS

Submitted to:

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE

GWALIOR - 474005 (MP) est. 1957

MAY-JUNE 2022

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

CERTIFICATE

This is certified that **PALAK GUPTA** (0901CS191072) has submitted the project report titled **E-MAIL NOTIFICATION SYSTEM USING MULESOFT** under the mentorship of ~~**Asst.**~~ **Mr. Mir Shahnawaz Ahmad**, in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering from Madhav Institute of Technology and Science, Gwalior.

09/05/22

Mr. Mir Shahnawaz Ahmad
Faculty Mentor
Assistant Professor
Computer Science and Engineering

09/05/2022

Dr. Manish Dixit
Professor and Head
Computer Science and Engineering

Dr. Manish Dixit
Professor & HOD
Department of CSE
M.I.T.S. Gwalior

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

CERTIFICATE

This is certified that **AMAN DAYANI** (0901CS191014) has submitted the project report titled **E-MAIL NOTIFICATION SYSTEM USING MULESOFT** under the mentorship of ~~**Asst.**~~ **Mr. Mir Shahnawaz Ahmad**, in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering from Madhav Institute of Technology and Science, Gwalior.

09/05/22

Mr. Mir Shahnawaz Ahmad

Faculty Mentor

Assistant Professor

Computer Science and Engineering

09/05/2022

Dr. Manish Dixit

Professor and Head

Computer Science and Engineering

Dr. Manish Dixit
Professor & HOD
Department of CSE
M.I.T.S. Gwalior

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

DECLARATION

I hereby declare that the work being presented in this project report, for the partial fulfilment of requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering at Madhav Institute of Technology & Science, Gwalior is an authenticated and original record of my work under the mentorship of **Mr. Mir Shahnawaz Ahmad**, **Assistant Professor, CSE, MITS.**

I declare that I have not submitted the matter embodied in this report for the award of any degree or diploma anywhere else.

PALAK GUPTA

0901CS191072

III Year,

Computer Science and Engineering

AMAN DAYANI

0901CS191014

III Year,

Computer Science and Engineering

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

ACKNOWLEDGEMENT

The full semester project has proved to be pivotal to my career. I am thankful to my institute, **Madhav Institute of Technology and Science** to allow me to continue my disciplinary/ interdisciplinary project as a curriculum requirement, under the provisions of the Flexible Curriculum Scheme (based on the AICTE Model Curriculum 2018), approved by the Academic Council of the institute. I extend my gratitude to the Director of the institute, **Dr. R. K. Pandit** and Dean Academics, **Dr. Manjaree Pandit** for this.

I would sincerely like to thank my department, **Department of Computer Science and Engineering, for allowing** me to explore this project. I humbly thank **Dr. Manish Dixit**, Professor and Head, Department of Computer Science and Engineering, for his continued support during the course of this engagement, which eased the process and formalities involved.

I am sincerely thankful to my faculty mentors. I am grateful to the guidance of **Asst. Prof. Mir Shahnawaz Ahmad ,CSE, MITS**, for his continued support and guidance throughout the project. I am also very thankful to the faculty and staff of the department.

PALAK GUPTA

0901CS191072

III Year,

Computer Science and Engineering

AMAN DAYANI

0901CS191014

III Year,

Computer Science and Engineering

Abstract

Underlying all IT architectures are core systems of records that are often not readily available due to its complexity and connectivity concerns. System APIs provide a means of hiding that complexity from the user while exposing data and providing downstream insulation from any interface changes or rationalization of those systems. This API provides an implementation best practice for a notification service that integrates with multiple systems like Gmail and Twilio and exposes a RESTful interface that triggers a notification.

सार:

अंतर्निहित सभी आईटी आर्किटेक्चर रिकॉर्ड की कोर सिस्टम हैं जो अक्सर इसकी जटिलता और कनेक्टिविटी चिंताओं के कारण आसानी से उपलब्ध नहीं होते हैं। सिस्टम एपीआई डेटा को उजागर करते समय उपयोगकर्ता से उस जटिलता को छिपाने का एक साधन प्रदान करते हैं और किसी भी इंटरफ़ेस परिवर्तन या उन प्रणालियों के युक्तिकरण से डाउनस्ट्रीम इन्सुलेशन प्रदान करते हैं। यह एपीआई एक अधिसूचना सेवा के लिए एक कार्यान्वयन सर्वोत्तम अभ्यास प्रदान करता है जो जीमेल और ट्विलियो जैसे कई सिस्टमों के साथ एकीकृत होता है और एक अधिसूचना को ट्रिगर करने वाले रीस्टफुल इंटरफ़ेस को उजागर करता है।

TABLE OF CONTENTS

TABLE OF FIGURES

TITLE **PAGE NO.**

Chapter 1: INTRODUCTION

1.1 What is an E-mail notification system?

Email notifications are a type of triggered email—email that's sent in response to specific user action or other event. For SaaS applications and web sites, common examples of these app-generated emails include activation and welcome messages, activity notifications, account and security alerts, and utilitarian functions like password resets.

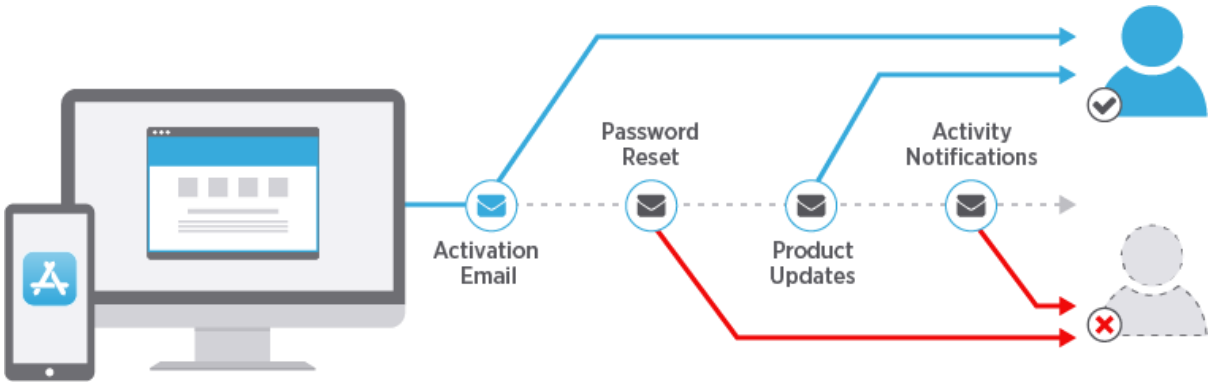

Figure1.1 E-mail notification system

These notifications serve an important purpose—alerting us when a post was shared on social media, reminding us to take action on a personal account, or asking us to approve payment for goods and services.

Beyond these are purely functional needs, notifications also are a valuable communication tool that enables product teams to directly engage with their customers. They are a persuasive instrument for drawing users back to using apps that they might have forgotten about. They help deliver a great user experience and are one of the most influential tools that product management teams have to drive conversion, retention, and growth. Additionally, and perhaps most importantly, they reinforce trust in services and help to build long-lasting relationships between a SaaS business and its customers.

1.2 Why do we need Email Notification System?

- **They have a high open-rate.** Email notifications draw more attention compared to promotional emails or newsletters since the notifications naturally contain only essential

information that influences the user directly. A higher open rate, in turn, boosts the server's reputation and ensures higher deliverability for your emails in the future.

- **Keep the customer up to date.** Changes to a website or service functionality can derail the usual routine of your visitors, better make them prepared if they need to adjust their routine for you. Sharing news on updates will minimize the possible adverse effect of changes.
- **They help with customer retention.** With regular notifications, visitors and customers can't forget about you. An email notification is a way to say, “Hey, buddy, I’m still here for you, no matter what.” Keeping your audience informed and being transparent nurtures trust and loyalty in people.

1.3 HOW MULE FLOW WORKS

On API console, when we click on “GET”

1. our request is send to the HTTP endpoint
2. where an HTTP listener forwards the request to APIKit Router.
3. APIKit Router validates the request and the method which we are calling.
4. and send the request to the relevant flow i.e. “get:/user/queryuser:tutorialsAtoZ-config”;
5. where are request is processed and we get a success response.

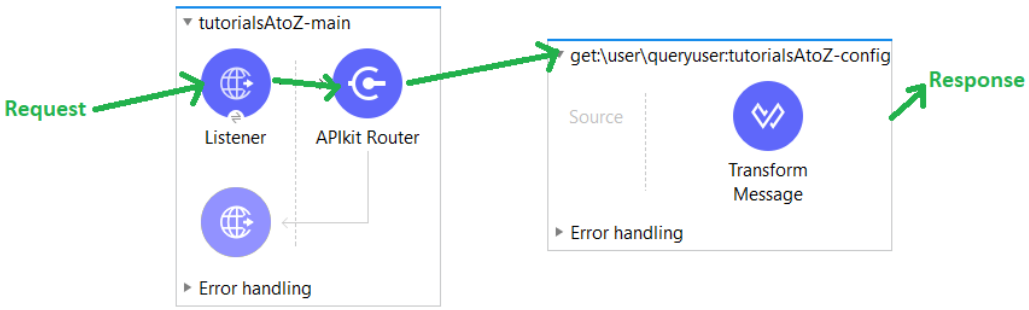

Figure 1.2 Mule flow

Chapter 2: INTRODUCTION TO MULESOFT

2.1What is MuleSoft?

MuleSoft unifies data to deliver a single view of the customer, automates business processes, and builds connected experiences. By using a modern API-led approach, each integration becomes a reusable building block.

MuleSoft is a vendor that provides an integration platform to help businesses connect data, applications and devices across on-premises and cloud computing environments. The company's platform, called Any point Platform, includes various tools to develop, manage and test application programming interfaces (APIs), which support these connections. MuleSoft, in May 2018, was acquired by Salesforce, a software as a service (SaaS) provider. Salesforce now uses MuleSoft technology as part of its Salesforce Integration Cloud.

2.1.1Components of MuleSoft-

- **API Designer** is a web-based, graphical tool that a developer can use to design and document an API, as well as share that design with team members. A developer can also choose to reuse specific components of an API, such as security schema.
- **API Manager** is an interface through which a developer can manage APIs, as well as secure them via an API gateway. With this component of the Anypoint platform, it's possible to control user access to APIs, ensure secure connections to back-end data sources and create policies around API calls and throttling.
- **Anypoint Studio** is a graphical, Java-based design environment that a developer can use to deploy APIs to on-premises and cloud environments. Studio also includes features to map, build, edit and debug data integrations.
- **Anypoint Connectors** are a set of built-in connectors that a developer can use to integrate applications with thousands of third-party REST and SOAP.
- **Anypoint Analytics** is an analytics tool to track API metrics, such as performance and usage. A developer can use this tool to create custom charts and dashboards to visualize API performance, as well as identify the root cause of any performance issues.
- **Anypoint Runtime Manager** is a central console from which a developer can provision and monitor all resources deployed on the Anypoint Platform across hybrid cloud architectures.
- **Anypoint Exchange** is a central hub that a development team can use to store and access APIs, templates, connectors, documentation and other resources.

- **Anypoint Monitoring** is a dashboard that helps a development team monitor application health.
- **Anypoint Visualizer** is a graphical tool to map APIs and their dependencies in real time.
- **Cloud Hub** is a multi-tenant integration platform as a service (iPaaS) offering. Cloud Hub is offered as a managed service, which means a development team does not need to install or operate any hardware or software to use it.

2.2What is an API?

API is the acronym for **Application Programming Interface**, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API. When you use an application on your mobile phone, the application connects to the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone. The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API.

Here is a real-life API example. You may be familiar with the process of searching flights online. Just like the restaurant, you have a variety of options to choose from, including different cities, departure and return dates, and more. Let us imagine that you're booking you are flight on an airline website. You choose a departure city and date, a return city and date, cabin class, as well as other variables. In order to book your flight, you interact with the airline's website to access their database and see if any seats are available on those dates and what the costs might be. However, what if you are not using the airline's website—a channel that has direct access to the information? What if you are using an online travel service, such as Kayak or Expedia, which aggregates information from a number of airline databases? The travel service, in this case, interacts with the airline's API. The API is the interface that, like your helpful waiter, can be asked by that online travel service to get information from the airline's database to book seats, baggage options, etc. The API then takes the airline's response to your request and delivers it right back to the online travel service, which then shows you the most updated, relevant information.

- Modern APIs adhere to standards (typically HTTP and REST), that are developer-friendly, easily accessible and understood broadly
- They are treated more like products than code. They are designed for consumption for specific audiences (e.g., mobile developers), they are documented, and they are versioned in a way that users can have certain expectations of its maintenance and lifecycle.

- Because they are much more standardized, they have a much stronger discipline for security and governance, as well as monitored and managed for performance and scale
- As any other piece of productized software, the modern API has its own software development lifecycle (SDLC) of designing, testing, building, managing, and versioning. Also, modern APIs are well documented for consumption and versioning.

2.3 What is an RAML?

RAML stands for **RESTful API Modelling Language**. A RAML provides a structure to the API which is useful for developers to start their development process and also helps client who is invoking the API to know beforehand what the API does.
A RAML contains:

1. Endpoint URL with its Query parameters and URI parameters,
2. HTTP methods to which API is listening to (GET, POST, PUT, DELETE),
3. Request and response schema and sample message,
4. HTTP response code that an API will return (e.g.: 200, 400, 404, 500).

2.4 What is POP3?

Post Office Protocol 3, or POP3, is the most commonly used protocol for receiving email over the internet. This standard protocol, which most email servers and their clients support, is used to receive emails from a remote server and send to a local client. Post Office Protocol version 3 (POP3) is a mail protocol used to retrieve mail from a remote server to a local email client. POP3 copies the mail from the remote server into the local mail client. Optionally, mail is deleted after it is downloaded from the server.

2.5 What is IMAP?

Internet Message Access Protocol (IMAP) is a protocol for accessing email or bulletin board messages from a (possibly shared) mail server or service. IMAP allows a client e-mail program to access remote message stores as if they were local.

IMAP allows you to access your email wherever you are, from any device. When you read an email message using IMAP, you aren't actually downloading or storing it on your computer; instead, you're reading it from the email service. IMAP is similar to POP3, except IMAP

supports both online and offline modes. For instance, IMAP users can leave email messages on the IMAP server until they explicitly delete them. Like POP3, IMAP cannot send email; for that, you must use an SMTP outbound endpoint. The POP3 connector implements a transport channel that enables your Mule application to retrieve email from a POP3 email server. The connector is configurable only as an inbound endpoint, that is, as a message source with a one-way exchange pattern.

2.6What is SMTP?

SMTP Stands for **Simple Mail Transfer Protocol.** SMTP is used to send and receive email. It is sometimes paired with IMAP or POP3 (for example, by a user-level application), which handles the retrieval of messages, while SMTP primarily sends messages to a server for forwarding. The SMTP is a protocol used to transfer e-mail messages and attachments. SMTP is used to transmit e-mail between e-mail servers and from e-mail clients (such as Microsoft Outlook or UNIX and Linux's sendmail) to e-mail servers (such as Microsoft Exchange).

Chapter 3: EMAIL CONNECTORS

Anypoint Connector for Email (Email Connector) sends and retrieves email messages over standard email protocols. Email Connector configurations share a basic set of parameters that require a connection over the protocols you use.

3.1 Pre-Requisites

To use this connector, you must be familiar with:

- Anypoint Connectors
- Mule runtime engine (Mule)
- Elements and global elements in a Mule flow
- Creating a Mule app using Anypoint Studio (Studio)

3.2 Use of connectors

Email Connector enables you to:

- Retrieve emails from POP3 mailboxes
- Retrieve, delete, and store emails from IMAP mailboxes
- Send emails over the SMTP protocol
- Support secure connections for all protocols over Transport Layer Security (TLS)

Chapter 4: WORKING AND CODING

4.1Create a Mule Project

In Studio, create a new Mule project in which to add and configure the connector:

1. In Studio, select **File > New > Mule Project**.
2. Enter a name for your Mule project and click **Finish**.

4.2 Add the Connector to Your Mule Project

Add Email Connector to your Mule project to automatically populate the XML code with the connector's namespace and schema location and to add the required dependencies to the project's `pom.xml` file:

1. In the **Mule Palette** view, click **(X) Search in Exchange**.
2. In the **Add Dependencies to Project** window, type `email` in the search field.
3. Click **Email Connector** in **Available modules**.
4. Click **Add**.
5. Click **Finish**.

Adding a connector to a Mule project in Studio does not make that connector available to other projects in your studio workspace.

4.3 Configure a Source

A source initiates a flow when a specified condition is met. You can configure one of these input sources to use with Email Connector:

- **On New Email - IMAP**
  Initiates a flow by retrieving all the emails from an IMAP mailbox folder, watermark can be enabled for polled items
- **On New Email - POP3**
  Initiates a flow by retrieving all the emails from an POP3 mailbox folder
- **HTTP > Listener**
  Initiates a flow each time it receives a request on the configured host and port
- **Scheduler**
  Initiates a flow when a time-based condition is met

For example, to configure the **On New Email - IMAP** source, follow these steps:

1. In the **Mule Palette** view, select **Email > On New Email - IMAP**.
2. Drag **On New Email - IMAP** to the Studio canvas.
3. On the **On New Email - IMAP** configuration screen, optionally change the value of the **Display Name** field.
4. Click the plus sign (+) next to the **Connector configuration** field to configure a global element that can be used by all instances of the source in the app.
5. On the **Email IMAP** window, for **Connection** select any of the connection types to provide to this configuration:
    - **IMAP Connection**
    - **IMAPS Connection**

    1. On the **General** tab, specify the connection information for the connector, such as **Host**, **Port** and **TLS configuration**.
    2. On the **Advanced** tab, optionally specify timeout configuration and reconnection information, including a reconnection strategy.
    3. Click **OK**.
    4. On the **On New Email - IMAP** configuration screen, in the **General** section, select a **Scheduling Strategy** to configure the scheduler that triggers the polling.

4.4 Add a Connector Operation to the Flow

When you add a connector operation to your flow, you immediately define a specific operation for that connector to perform.

To add an operation for Email Connector, follow these steps:

1. In the **Mule Palette** view, select **Email** and then select the desired operation.
2. Drag the operation onto the Studio canvas and to the right of the input source.

The following screenshot shows the Email Connector operations in the Mule Palette of Anypoint Studio:

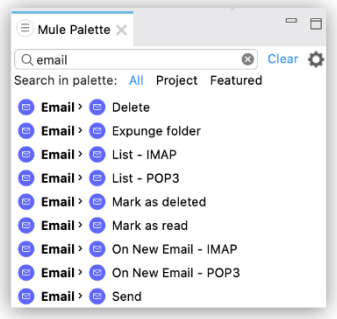

Figure 4.1 Email Connector Operations

4.5 Configure a Global Element for the Connector

When you configure a connector, it's best to configure a global element that all instances of that connector in the app can use. Configuring a global element for Email Connector requires you to set a configuration based on the supported protocols for the connector operations:

Global Element Configuration	Connector Operation/Sources
IMAP (Internet Message Access Protocol)	○ **Delete** ○ **Expunge Folder** ○ **List - IMAP** ○ **Mark As Deleted** ○ **Mark As Read** ○ **On New Email - IMAP**
POP3 (Post Office Protocol 3)	○ **List POP3** ○ **On New Email - POP3**
SMTP (Simple Mail Transfer Protocol)	○ **Send**

For example, to configure a POP3 global element for the **List POP3** operation, follow these steps:

1. Select the name of the connector in the Studio canvas.
2. Select the **List POP3** operation in the Studio canvas.
3. In the **List POP3** configuration screen for the operation, click the plus sign (+) next to the **Connector configuration** field to access the global element configuration fields. On the **Email POP3** window, for **Connection** select any of the connection types to provide to this configuration:
    - **POP3 Connection**

- **POP3S Connection**

1. On the **General** tab, specify the connection information for the connector, such as **Host**, **Port** and **TLS configuration**.
2. On the **Advanced** tab, optionally specify timeout configuration and reconnection information, including a reconnection strategy.
3. Click **OK**.

The following screenshot shows the Email Connector Global Element Configuration window in Anypoint Studio:

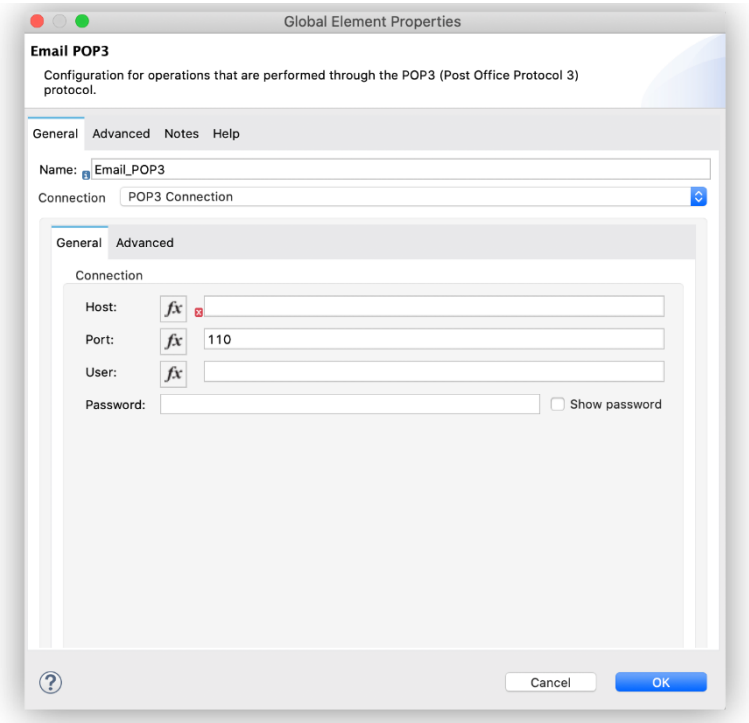

Figure 4.2 Email Connector Global Element Configuration

4.6 Configure Additional Connector Fields

For IMAPS, POP3S, and SMTPS protocol connections, you can use Transport Layer Security (TLS) and configure email by providing a key store with your certificate. Additionally, you can also enable two-way authentication by providing a trust store.

4.7 View the App Log

To check for problems, you can view the app log as follows:

- If you're running the app from Anypoint Platform, the output is visible in the Anypoint Studio console window.

- If you're running the app using Mule from the command line, the app log is visible in your OS console.

Unless the log file path is customized in the app's log file (`log4j2.xml`), you can also view the app log in the default location `MULE_HOME/logs/<app-name>.log` .

4.8 Send Emails with Email Connector Examples - Mule 4

Use Anypoint Connector for Email (Email Connector) to send messages over SMTP and SMTPS servers by using the **Send** operation. The following examples show how to configure Email Connector to send emails and attachments, in both Anypoint Studio and the XML editor:

- **Send Emails**
  Configure the Email Connector **Send** operation to send emails over an SMTPS server.
- **Send an Attachment**
  Configure the File Connector **Read** operation to read a JSON file and the Email Connector **Send** operation to send the file as an attachment over an SMTP server.
- **Send Multiple Attachments**
  Configure **Set Variable** components to define multiple media type attachments, and send these attachments using the Email Connector **Send** operation. Then log the result message using a **Logger** component.

4.9 Send Emails

The following example illustrates how to send emails over the SMTPS server. The flow initiates with a **Scheduler** component, and then the **Send** operation sends the email. The SMTPS connection type enables SSL or TLS encryption and sends encrypted messages over the secured version of the SMTP server.

The following screenshot shows the app flow for this example:

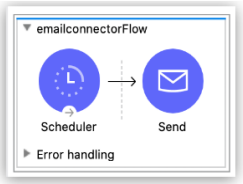

Figure 4.3 Email Connector Send messages over SMTPS

To create the flow:

1. Create a new Mule project in Studio.

2. In the **Mule Palette** view, select the **Scheduler** component and drag it onto the canvas. This source initiates a flow when a time-based condition is met.
3. Drag the Email **Send** operation to the right of the **Scheduler** source.
4. On the **Send** configuration screen, click the plus sign (+) next to the **Connector configuration** field to configure a global element for the operation.
5. In the **Connection** field, select **SMTPS Connection**.
6. In the **General** tab, enter the following values:
    - **Host**
      `pop.gmail.com`
    - **Port**
      `995`
    - **User**
      `user3@domain.com`
    - **Password**
      `passwordconnection`
7. In the **TLS Configuration** section, select **Edit inline**.
8. In the **Trust Store Configuration** section, enter the following values:
    - **Path**
      `aTruststore.jks`
    - **Password**
      `passwordtrust`
9. In the **Key Store Configuration** section, enter the following values:
    - **Path**
      `aKeystore`
    - **Password** `passwordkey`
10. In the **Advanced** section, set **Enabled Protocols** to `TLSv1.2,SSLv3`.

11. Click **OK** to close the configuration window.

The following screenshot shows the **Email SMTPS** configuration:

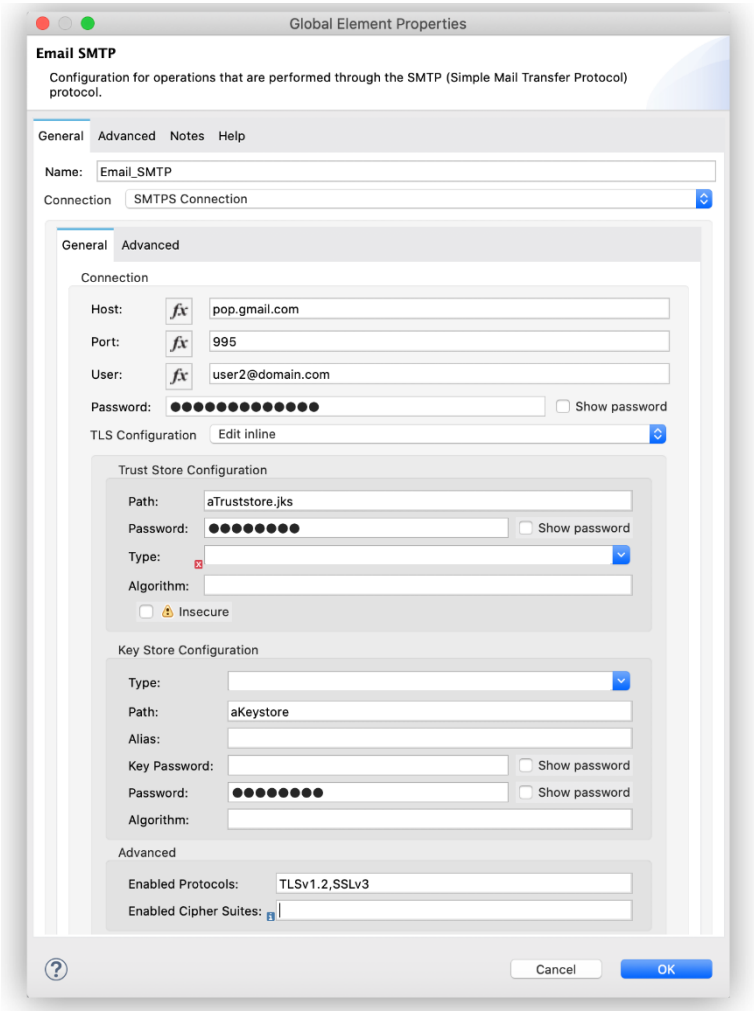

Figure 4.4 Email SMTPS configuration

1. In the **Send** operation configuration screen, set the following values:
    - **From address**

      `user1@domain.com`
    - **To addresses**

      `Edit inline`
2. Click the plus sign (+) to add new emails.
3. Set the **Value** field to `user2@domain.com`.
4. Repeat the previous two actions and add another email as `user3@domain.com`.

5. Set the **Subject** field to `IMPORTANT.`

6. Set the **Content** field to `"<h1>Hello this is an important message</h1>"` and the **ContentType** field to `text/html`.

   If **Content** is not empty, the content of the message payload is used by default. If that payload is not `text`, the operation fails with an `EMAIL:SEND` error.

7. Save and run the app.

The following screenshot shows the **Send** operation configuration:

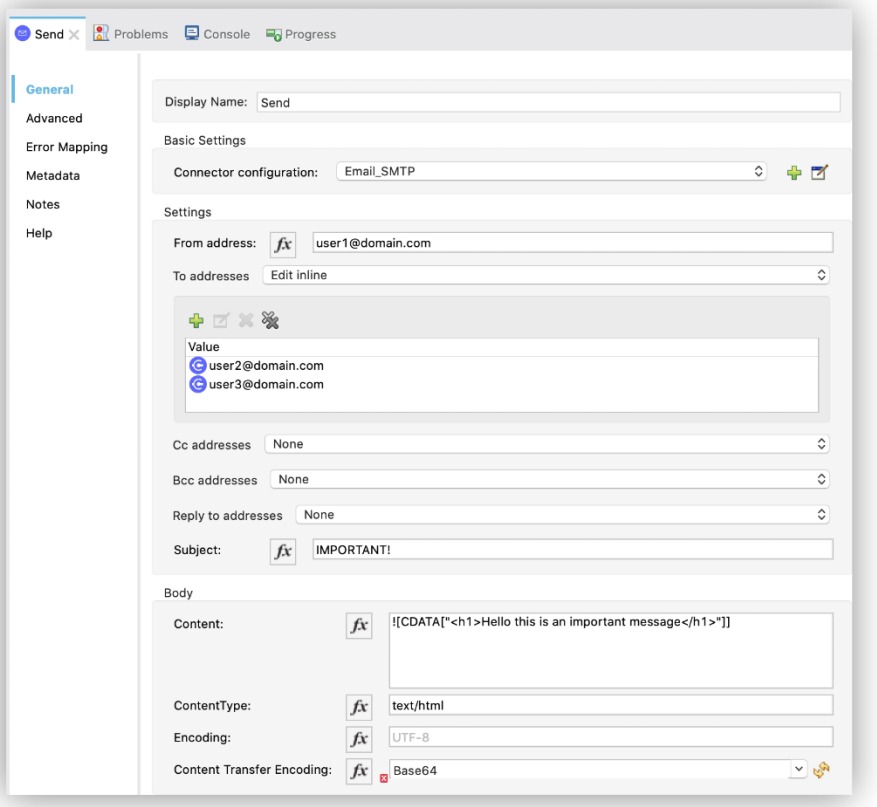

Figure 4.5 Email Send operation configuration

4.10 XML for Sending Emails

Paste this code into the Studio XML editor to quickly load the flow for this example into your Mule app:

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:tls="http://www.mulesoft.org/schema/mule/tls"
xmlns:email="http://www.mulesoft.org/schema/mule/email"
        xmlns="http://www.mulesoft.org/schema/mule/core"
        xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/email
http://www.mulesoft.org/schema/mule/email/current/mule-email.xsd
http://www.mulesoft.org/schema/mule/tls http://www.mulesoft.org/schema/mule/tls/current/mule-
tls.xsd">
        <email:smtp-config name="Email_SMTP" >
                <email:smtps-connection host="pop.gmail.com" port="995"
user="user1@domain.com" password="passwordvalue" >
                        <tls:context enabledProtocols="TLSv1.2,SSLv3" >
                                <tls:trust-store path="aTruststore.jks" password="changeit"
/>
                                <tls:key-store path="aKeystore" password="password" />
                        </tls:context>
                </email:smtps-connection>
        </email:smtp-config>
        <flow name="emailconnectorFlow">
                <scheduler>
                        <scheduling-strategy >
                                <fixed-frequency />
                        </scheduling-strategy>
                </scheduler>
                <email:send config-ref="Email_SMTP" fromAddress="user1@domain.com"
subject="IMPORTANT!">
                        <email:to-addresses >
                                <email:to-address value="user3@domain.com" />
    <email:to-address value="user2@domain.com" />
                        </email:to-addresses>
                        <email:body contentType="text/html" >
```

```
                                        <email:content ><![CDATA["<h1>Hello this is an
important message</h1>"]]></email:content>
                                    </email:body>
                            </email:send>
                    </flow>
</mule>
```

4.11 Send an Attachment

The following example illustrates how to send emails and attachments over the SMTP server. Use DataWeave to manage the attachments. The flow reads a JSON file using the File Connector **Read** operation and then uses the Email Connector **Send** operation to send the content of the file as an attachment:

The following screenshot shows the Anypoint Studio app flow for this example:

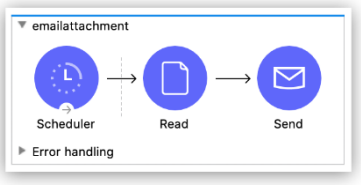

Figure 4.6 Email Connector Send attachments over SMTP

To create the flow:

1. Create a new Mule project in Studio.
2. In the **Mule Palette** view, select the **Scheduler** component and drag it onto the canvas. The source initiates a flow when a time-based condition is met.
3. Drag the File Connector **Read** operation to the right of the **Scheduler** component.
4. Set the **File Path** field to `file.json`.
5. Drag the **Send** operation to the right of the **Read** operation.
6. On the **Send** configuration screen, click the plus sign (+) next to the **Connector configuration** field to configure a global element for the operation.
7. In the **Connection** field, select **SMTP Connection**.
8. In the **General** tab, enter the following values:

- **Host**
  `pop.gmail.com`
- **Port**
  `995`
- **User**
  `user1@domain.com`
- **Password**
  `password`

9. Click **OK**.
10. In the **Send** operation configuration screen, set the **To addresses** field to `Edit inline`.
11. Click the plus sign (+) to add a new email **Value** as `example@domain.com`.
12. Set the **Subject** field to `Attachment test`.
13. Set the **Content** field to `"<h1>Hello this is an important message</h1>"`.
14. In the **Attachments** field, enter the following DataWeave expression as a new attachment element:

```
{
  'json-attachment' : payload
}
```

Note that `payload` is the content of the JSON file read by the File Connector **Read** operation.

17. Save and run the app.

The following screenshot shows the **Send** operation configuration:

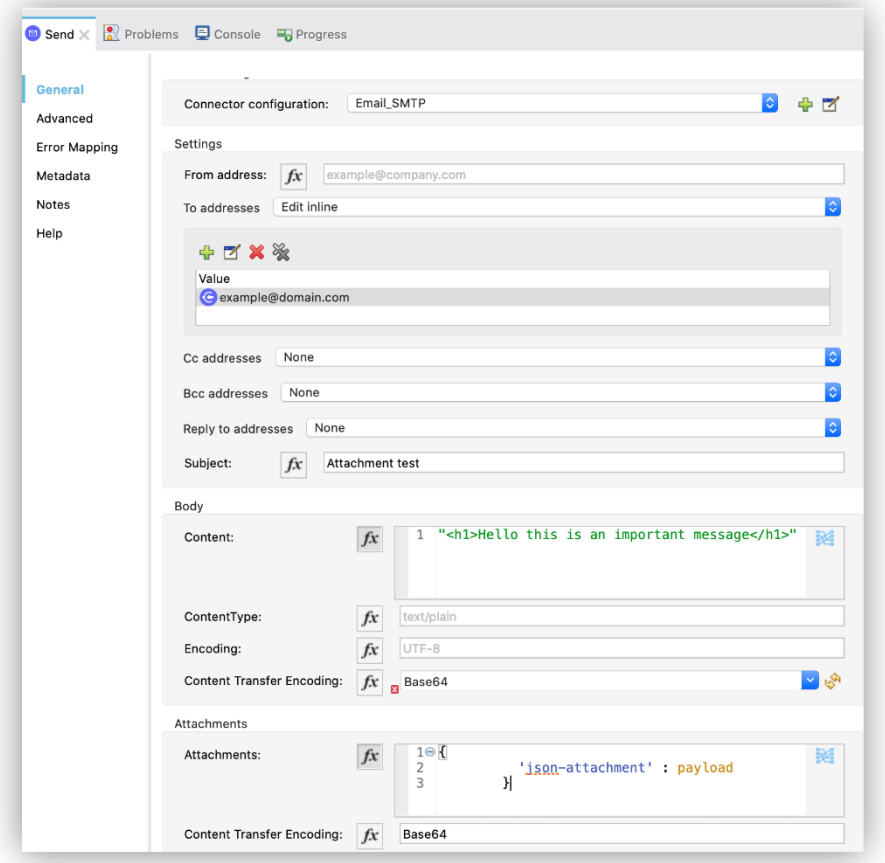

Figure 4.7 Email Send Attachment operation configuration

4.12 Send Multiple Attachments

The following example illustrates how to send emails and multiple attachments over the SMTP server. The flow initiates with a **Scheduler** component. Then, **Set Variable** components define each attachment media type JSON, text and file. Subsequently, the Email Connector **Send** operation sends the attachments in the email, after which you the result message using a **Logger** component.

The following screenshot shows the Anypoint Studio app flow for this example:

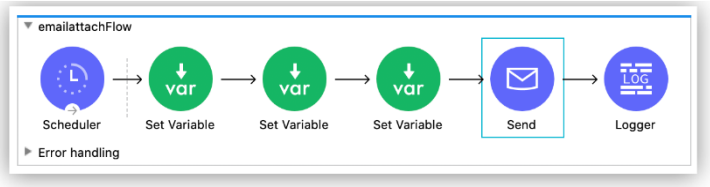

Figure 4.8 Email Connector Multiple attachments

To create the flow:

1. Create a new Mule project in Studio.

2. In the **Mule Palette** view, select the **Scheduler** component and drag it onto the canvas. The source initiates a flow when a time-based condition is met and sends the emails.
3. Drag a **Set Variable** component to the right of the **Scheduler** component.
4. In the **Set Variable** configuration screen, set the following parameters:
    - **Name**

      `json`
    - **Value**

      `output application/json --- {address: '221B Baker Street'}`
    - **MIME Type**

      `application/json`
5. Drag another **Set Variable** component to the right of the first **Set Variable** component, and set the following parameters:
    - **Name**

      `textPlain`
    - **Value**

      `This is the email text attachment for John Watson`
    - **MIME Type**

      `text/plain`
6. Drag another **Set Variable** component to the right of the second **Set Variable** component, and set the following parameters:
    - **Name**

      `octectStream`
    - **Value**

      `vars.textPlain`
    - **MIME Type**

      `application/octet-stream`
7. Drag the Email **Send** operation to the right of the third **Set Variable** component.

8. On the **Send** configuration screen, click the plus sign (+) next to the **Connector configuration** field to configure a global element for the operation.
9. In the **Connection** field, select **SMTP Connection**.
10. In the **General** tab, enter the following values:
    - **Host**
      `pop.gmail.com`
    - **Port**
      `995`
    - **User**
      `user1@domain.com`
    - **Password**
      `password`
11. Click **OK**.
12. In the **Send** operation configuration screen, set the **To addresses** field to `Edit inline`
13. Click the plus sign (+) to add a new email **Value** as `user4@domain.com`.
14. Set the **Content** field to `Email Content` and the **ContentType** field to `text/plain`.
15. In the **Attachments** field, enter the following DataWeave expression:

```
{
        'text-attachment' : vars.textPlain,
        'json-attachment' : vars.json,
        'stream-attachment' : vars.octetStream
}
```

20. Save and run the app.

The following screenshot shows the **Send** operation configuration:

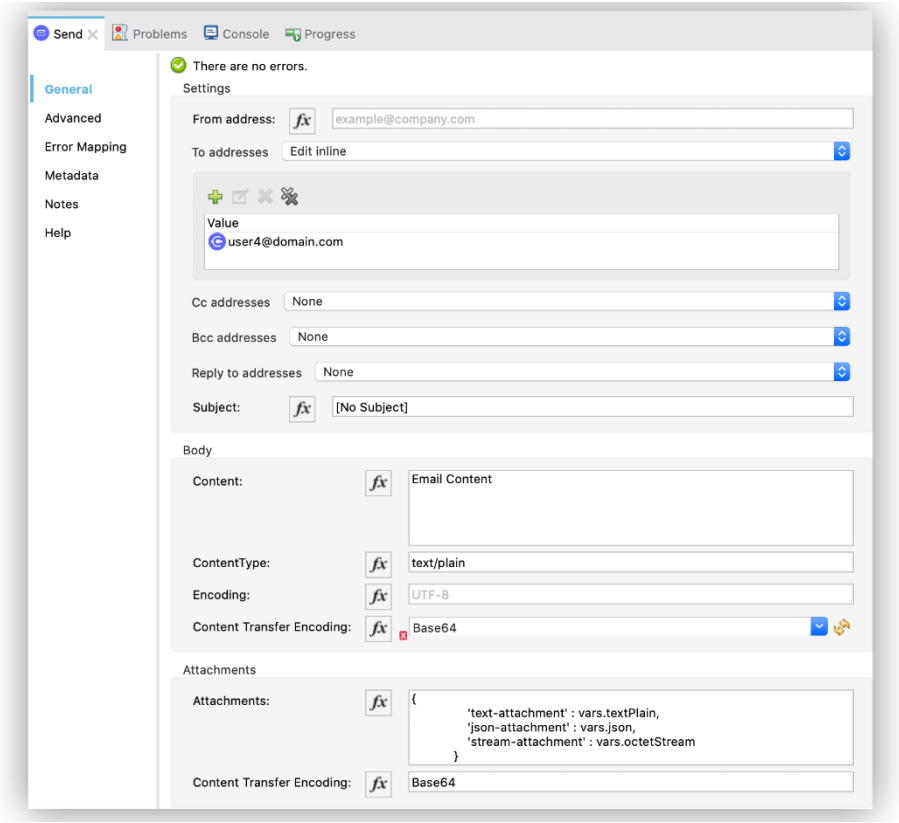

Figure 4.9 Email Send Multiple Attachment configuration

XML for Sending Multiple Attachments

Paste this code into the Studio XML editor to quickly load the flow for this example into your Mule app:

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:email="http://www.mulesoft.org/schema/mule/email"
        xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
http://www.mulesoft.org/schema/mule/email
http://www.mulesoft.org/schema/mule/email/current/mule-email.xsd
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd">
        <email:smtp-config name="Email_SMTP">
                <email:smtp-connection host="pop.gmail.com" port="995"
user="user1@domain.com" password="password" />
```

```
        </email:smtp-config>
        <flow name="emailattachFlow">
                <scheduler>
                        <scheduling-strategy >
                                <fixed-frequency />
                        </scheduling-strategy>
                </scheduler>
                <set-variable value="#[output application/json --- {address: '221B Baker
Street'}]" variableName="json" mimeType="application/json"/>
    <set-variable value="This is the email text attachment for John Watson"
variableName="textPlain" mimeType="text/plain"/>
    <set-variable value="#[vars.textPlain]" variableName="octetStream"
mimeType="application/octet-stream"/>
    <email:send config-ref="config">
      <email:to-addresses >
        <email:to-address value="user4@domain.com" />
      </email:to-addresses>
      <email:body contentType="text/plain" >
        <email:content>Email Content</email:content>
      </email:body>
      <email:attachments>#[{
        'text-attachment' : vars.textPlain,
        'json-attachment' : vars.json,
        'stream-attachment' : vars.octetStream
      }]</email:attachments>
    </email:send>
                <logger level="INFO" doc:name="Logger" message="#['Message Id ' ++
correlationId as String]"/>
        </flow>

</mule>
```

4.13 Connect to Gmail with Email Connector Examples - Mule 4

To connect Anypoint Connector for Email (Email Connector) to your Gmail account, enable **Less Secure Apps** in your account and configure a global element for the server connection IMAPS, POP3S, or SMTPS you want to use.

Enable Less Secure Apps in Your Gmail Account

Google might block third-party mail clients if you do not enable the **Less Secure Apps** setting for your Gmail account, because Google blocks any apps that are not using security protocols that Google deems mandatory unless you allow less secure apps to access your Gmail account.

When you try to connect to your Gmail for the first time, you will receive an email with a link for enabling less secure apps in your Gmail account.

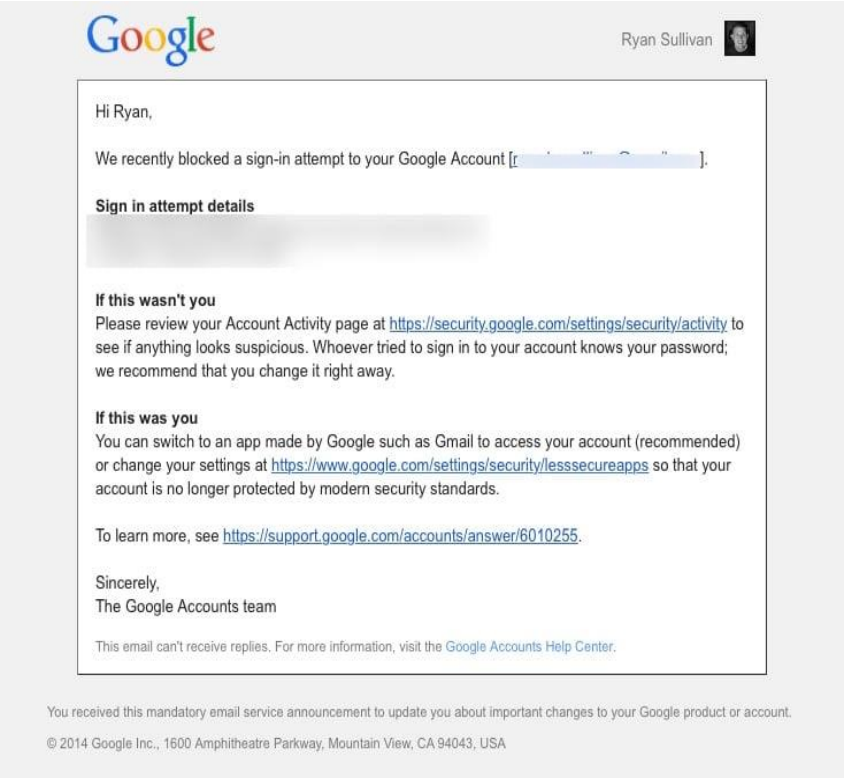

Figure 4.10 Email connector example

Chapter 5 Conclusion And Future Scope

5.1 Conclusion

Underlying all IT architectures are core systems of records that are often not readily available due to its complexity and connectivity concerns. System APIs provide a means of hiding that complexity from the user while exposing data and providing downstream insulation from any interface changes or rationalization of those systems. This API provides an implementation best practice for a notification service that integrates with multiple systems like Gmail and Twilio and exposes a RESTful interface that triggers a notification.

5.2 Future Scope

A MuleSoft Developer is familiar with end-to-end integration solutions designing and development. With Mule being included in Salesforce ecosystem, a significant growth can be seen. But all tool-based solutions have high chances of being replaced in few years. Also, there is a lot of scope of learning and growth with associated systems. It is easier for a person with said qualifications to work on any integrations.

References

MuleSoft Documentation-

https://docs.mulesoft.com/general/

YouTube Reference

https://www.youtube.com/watch?v=taxnRe2FXRY&t=186s

About MuleSoft and Training

https://training.mulesoft.com/

https://www.mulesoft.com/about