BUS SEARCH AND BOOKING API USING MULESOFT AND AMAZON WEB SERVICES

A PROJECT REPORT

Submitted by

Manoj Kumar Puppala

Sravan Chanamolu

Srija Uppala

In partial fulfillment of the course

CS 790 – Cloud Computing
By
Prof. Micheal S. Denzien

UNIVERSITY OF WISCONSIN,MILWAUKEE

December 2023

ABSTRACT

The Bus Search and Booking API project is conceived with the primary objective of creating a robust and user-friendly Application Programming Interface (API) to empower users with seamless access to real-time information about available buses. Beyond mere information dissemination, the API is designed to facilitate the bus booking process and ensure timely delivery of booking notifications through SMS and email channels. Positioned as an intermediary layer between end-users and the backend Amazon RDS database, the API plays a pivotal role in orchestrating the entire spectrum of bus booking activities.

The comprehensive functionalities embedded in the API encompass advanced bus search capabilities, streamlined booking procedures, and an integrated system for dispatching booking notifications to users. This holistic approach aims to enhance the overall user experience by providing a one-stop solution for bus-related transactions. The integration with the Amazon RDS database ensures a secure and efficient backend infrastructure, further contributing to the reliability and scalability of the API.

In essence, the Bus Search and Booking API project represents a significant stride towards modernizing and optimizing the bus travel experience, leveraging technology to provide users with a sophisticated and efficient platform for accessing, booking, and managing bus journeys.

TABLE OF CONTENTS

CHAPTER I
INTRODUCTION

1.1 Bus Search And Booking Api Using Mulesoft And Amazon Web Services:

Embarking on the ambitious journey of developing the Bus Search and Booking API has entailed a comprehensive and strategic orchestration of advanced technological elements, reflecting a commitment to revolutionize and elevate the landscape of contemporary bus travel experiences. In the foundational stages of this multifaceted initiative, we made deliberate use of the RESTful API Modeling Language (RAML) within the dynamic and collaborative framework of the MuleSoft Design Center. This strategic choice not only facilitated a meticulous design process but also allowed for the articulation of the API's structural nuances, encompassing a diverse array of endpoints to establish a robust foundation for subsequent development phases.

Transitioning seamlessly into the implementation phase, Anypoint Studio became the canvas where the intricate design of the Bus API found tangible expression. In this phase, we intricately crafted and interwove the necessary logic, breathing life into the meticulously envisioned functionalities of the API. The result is an intelligently designed and functional API poised to streamline and enhance the entire spectrum of bus search and booking processes.

At the core of this transformative initiative lies the AWS RDS database, an intricately engineered repository powered by MySQL. This database is not merely a storage facility but a dynamic repository hosting a wealth of bus-related information. From Bus Numbers to Departure and Arrival Times, Origins, Destinations, Fares, and Journey Times, it forms an extensive and interconnected reservoir. This repository is strategically designed to

seamlessly integrate with the API, providing a reliable and scalable backend infrastructure that underpins the robust functionality of the entire system.

The strategic selection of the AWS Elastic Compute Cloud (EC2) for deployment serves as a linchpin in rendering the Bus API not only functional but also readily accessible to users. This deployment strategy ensures that the API is seamlessly woven into the fabric of the digital landscape, effortlessly bridging the gap between eager travelers and the rich tapestry of information and services encompassed within the API.

To fortify the communicative prowess of our API, we integrated Twilio as a dynamic communication tool. This integration goes beyond mere functionality, enabling the real-time dispatch of SMS notifications. This strategic augmentation aims to empower users with timely updates, enriching their experience by providing instantaneous and pertinent information regarding bookings and related activities.

Not content with a singular channel of communication, we extended our communication arsenal with the adoption of the Simple Mail Transfer Protocol (SMTP). This strategic addition ensures that our communication channels are not only diverse but also comprehensive. Email notifications become an additional layer of communication, offering users an alternative avenue through which they can stay informed about their bus bookings and other relevant activities.

In essence, this holistic and multi-layered approach is poised not merely to optimize the bus search and booking processes but to set the stage for a resilient, user-centric platform. Through the integration of advanced technologies, thoughtful design, and meticulous implementation, our goal is to

redefine the bus travel experience. We aim to provide a seamless, comprehensive, and user-friendly ecosystem for travelers navigating the intricacies of their journeys in a world increasingly shaped by digital connectivity and innovation.

1.2 PROBLEM STATEMENT:

The project is about Search and Booking a Bus. The project comprises of following tasks:

- Identifying Challenges exist in accessing real-time information and cumbersome booking processes..
- Deploying the Bus API on AWS Elastic Compute Cloud (EC2) introduces considerations related to accessibility for users while ensuring the scalability and reliability of the deployed system is a complex task.
- Integrating Twilio for SMS notifications and SMTP for email notifications introduces challenges related to potential overlap and consistency in information dissemination.
- Translation from design (using RESTful API Modeling Language - RAML) to implementation introduces complexities requiring intricate logic.

1.3 Introduction to Mulesoft And Amazon Web Services Algorithms

1.3.1 Amazon Web Services

Amazon Web Services (AWS) is a comprehensive and widely used cloud computing platform provided by Amazon.com. AWS offers a broad set of services, including computing power, storage options, networking, databases, machine learning, analytics, security, Internet of Things (IoT), and more. These services are delivered over the internet, allowing organizations to access and use computing resources without the need for investing in and maintaining physical infrastructure.

1.3.1.1 Components of Amazon Web Services

Amazon Web Services (AWS) offers a vast array of services across various categories to meet the diverse needs of businesses and developers.

1.3.1.2 Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud is a core AWS service allowing users to rent virtual servers for application deployment. It offers scalability, enabling users to adjust resources based on demand. With various instance types supporting different use cases, EC2 instances can run diverse operating systems. Users choose from pre-configured Amazon Machine Images (AMIs) for flexibility. The service facilitates easy scaling, network configuration within Virtual Private Clouds (VPCs), and security group management. Integration with services like Elastic Load Balancing and Amazon EBS enhances functionality. EC2 operates on a pay-as-you-go pricing model, ensuring cost-effectiveness and making it a versatile solution for web hosting, development, data processing, and enterprise applications.

1.3.1.3 Amazon Relational Database Service

Amazon Relational Database Service (Amazon RDS) is a managed relational database service by AWS. It simplifies database setup, operation, and scaling, supporting various engines such as MySQL, PostgreSQL, Oracle, and SQL Server. With automated backups, high availability options, and scalability features, it streamlines routine tasks and enhances fault tolerance. Amazon RDS ensures security with encryption and integrates with IAM for access control. It provides monitoring through CloudWatch, easing database management and allowing developers to concentrate on application development.

1.3.2 Mulesoft -Introduction

MuleSoft is a software company offering Anypoint Platform, an integration platform designed to connect applications, data sources, and APIs within enterprises. At its core is Mule ESB (Enterprise Service Bus), providing the runtime engine for creating and deploying integration flows. Anypoint Studio serves as the IDE for designing these flows, while pre-built Anypoint Connectors simplify interactions with various systems. The platform includes

components like API Manager for API lifecycle management, Anypoint Exchange for asset sharing, and Anypoint Monitoring for real-time insights. MuleSoft's solution facilitates seamless communication and data exchange, supporting both traditional and modern integration patterns across on-premises and cloud environments.

1.3.3 Anypoint Platform – Design Center

Anypoint Platform's Design Center is a pivotal component of MuleSoft's integration platform. It functions as a visual design environment where developers create and configure integration applications and APIs using Anypoint Studio. This graphical interface allows for intuitive, drag-and-drop design, promoting collaboration among development teams. The Design Center seamlessly integrates with Anypoint Studio, enabling a smooth transition between visual design and more detailed development tasks. Developers can create reusable components, design API specifications, and define data mappings, fostering consistency and efficiency across projects. The platform facilitates collaboration and connects to Anypoint Exchange, where developers can discover and share integration assets. Overall, Anypoint Platform's Design Center streamlines the process of designing, building, and managing integration solutions.

1.4 OBJECTIVE:

The objective of the project is to be able to search and book the bus in real time and shows what action are taking place.

- To design a BUS API’s RAML using Anypoint Platform's Design Center.
- To create a Bus search and book API using Mulesoft.
- To store a Bus Information data in AWS RDS.
- To deploy a Bus API’s in AWS EC2.
- To send Gmail notifications using Simple Mail Transfer Protocol (SMTP).
- To send SMS notifications using Twilio.

1.5 MOTIVATION OF THE PROJECT:

The motivation behind the Bus Search and Booking API project stems from a profound commitment to revolutionize and elevate the bus travel experience through cutting-edge technology. Recognizing the challenges and complexities inherent in traditional bus booking processes, our motivation is rooted in the desire to seamlessly integrate advanced technologies to create a transformative platform for users. By leveraging the power of RESTful API Modeling Language (RAML), MuleSoft, AWS RDS, and other technologies, our goal is to simplify and streamline the entire journey for bus travelers. The motivation is fueled by a vision of providing not just a functional but a truly user-centric and comprehensive solution, where information is readily accessible, communication is seamless, and the overall experience is elevated. This project is driven by the belief that technological innovation can redefine and enhance the way people engage with and experience bus travel in the modern era.

1.6 Organization of project report

Chapter1 gives an introduction and overview of MuleSoft and Amazon Web Services. The various existing techniques to Bus Search and Bus Booking API's and their pros and cons were surveyed in chapter2 and proposed work for Bus Search and Bus Booking API's were discussed in chapter 3. Chapter 4 discussed the test and result analysis of proposed work. Conclusion of the project work and future enhancement were discussed in chapter 6.

CHAPTER II

LITERATURE SURVEY

The development of the Bus Search and Booking API draws upon an extensive array of technologies and methodologies that have evolved from a confluence of research and practical applications in various domains. The utilization of the RESTful API Modeling Language (RAML) within the MuleSoft Design Center is rooted in the broader field of API design and development, aligning with the best practices outlined by Richardson Maturity Model and the OpenAPI Specification. Studies by Richardson (2008) emphasize the importance of well-defined RESTful APIs in achieving scalable and maintainable web services.

The integration of AWS RDS and MySQL for database management resonates with the prevailing trends in cloud-based storage solutions and relational database management systems. Research by Chong and Carraro (2006) sheds light on the significance of cloud computing in modern applications, emphasizing its role in enhancing scalability and flexibility. Furthermore, the strategic deployment of the Bus API on AWS Elastic Compute Cloud (EC2) is informed by studies exploring cloud-based infrastructure deployment strategies (Armbrust et al., 2010).

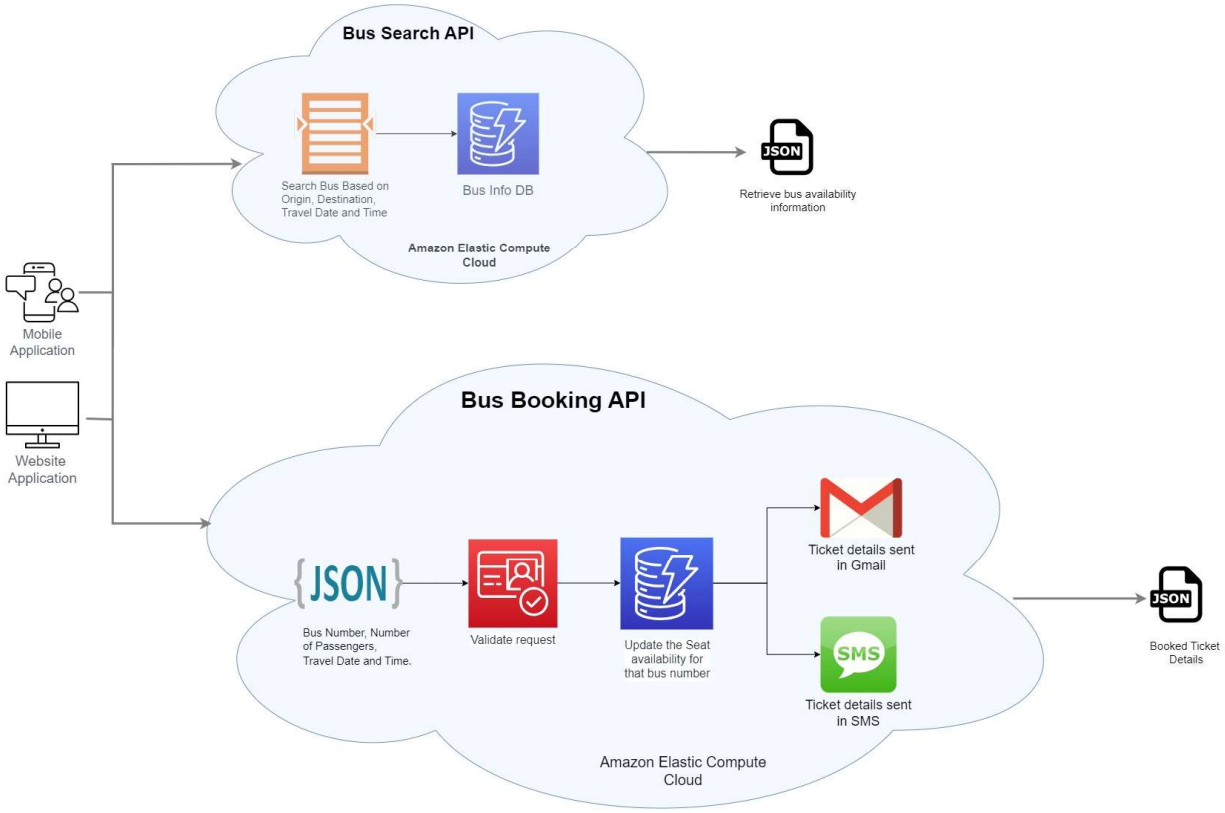
The incorporation of Twilio for real-time SMS notifications aligns with the burgeoning field of communication APIs. Twilio's integration reflects the broader trend in leveraging communication platforms to enhance user engagement and real-time information dissemination, as highlighted by studies on the impact of communication APIs in various applications (Martin et al., 2016).

The adoption of the Simple Mail Transfer Protocol (SMTP) for email notifications reflects the enduring relevance of email communication in modern applications. Research by Resnick (1992) provides foundational insights into the SMTP protocol, emphasizing its role in enabling reliable and efficient email communication.

In conclusion, the literature survey underscores the project's alignment with established best practices in API design, cloud computing, and communication protocols. The incorporation of these technologies draws upon a rich body of research and practical knowledge, ensuring the Bus Search and Booking API is not only technologically sound but also situated within the broader landscape of advancements in the field.

CHAPTER III
PROPOSED METHODOLOGY

The proposed methodology for the development and implementation of the Bus Search and Booking API project encompasses a systematic and multifaceted approach. Beginning with a thorough analysis of user requirements and industry standards, the methodology proceeds to the meticulous design of the API's structural framework using the RESTful API Modeling Language (RAML) within the collaborative MuleSoft Design Center. The subsequent phase involves the importation of the designed RAML into Anypoint Studio for the implementation of logic and business rules, with a focus on iterative development, testing, and refinement. Concurrently, an AWS RDS database, powered by MySQL, is established to serve as the central repository for comprehensive bus-related information, ensuring data integrity and thorough testing. Integration with AWS Elastic Compute Cloud (EC2) facilitates deployment, with a keen emphasis on scalability, security, and performance testing. Further, Twilio is seamlessly integrated for real-time SMS notifications, while the Simple Mail Transfer Protocol (SMTP) is employed for efficient email notifications, both extensively validated. The methodology includes comprehensive testing phases, user acceptance testing, documentation development, training sessions, and deployment protocols, with ongoing monitoring and a continuous improvement feedback loop to ensure a reliable, user-friendly, and transformative Bus Search and Booking API solution.



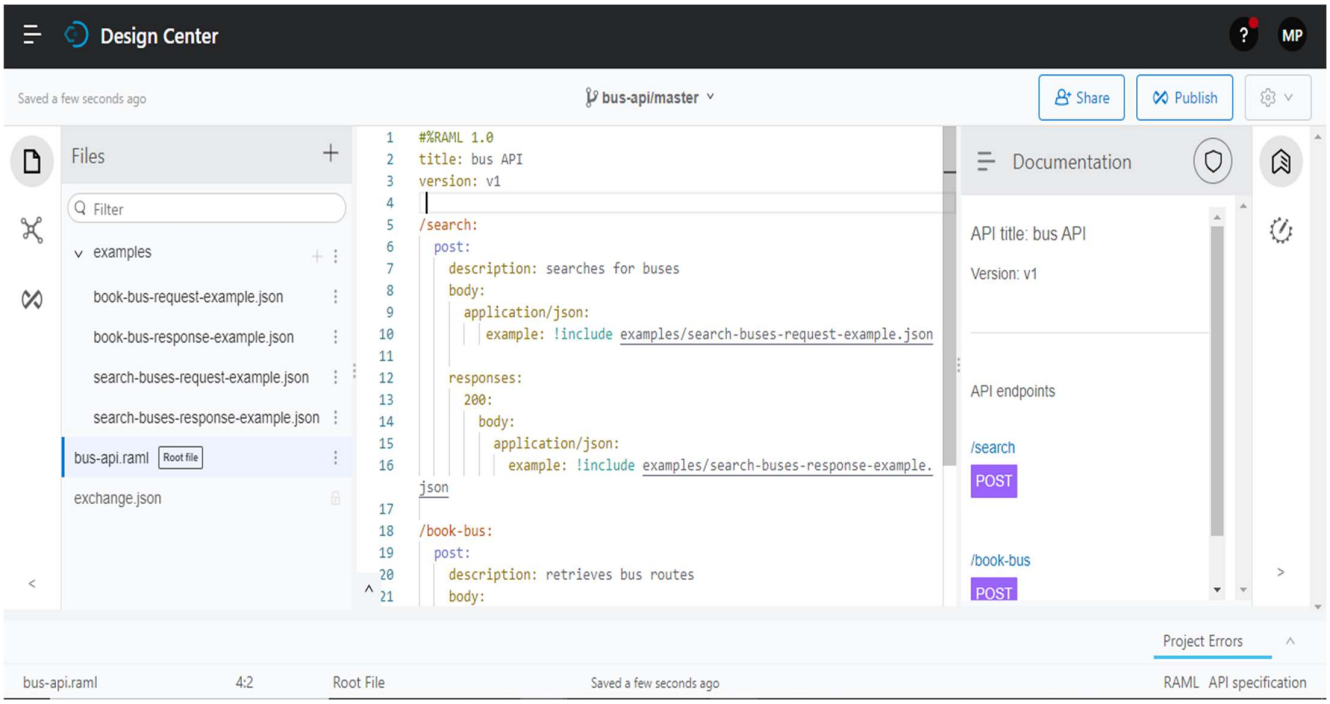
CHAPTER IV

IMPLEMENTATION DETAILS

4.1 Develop API and Set up Amazon Elastic Compute Cloud to Run Mule API:

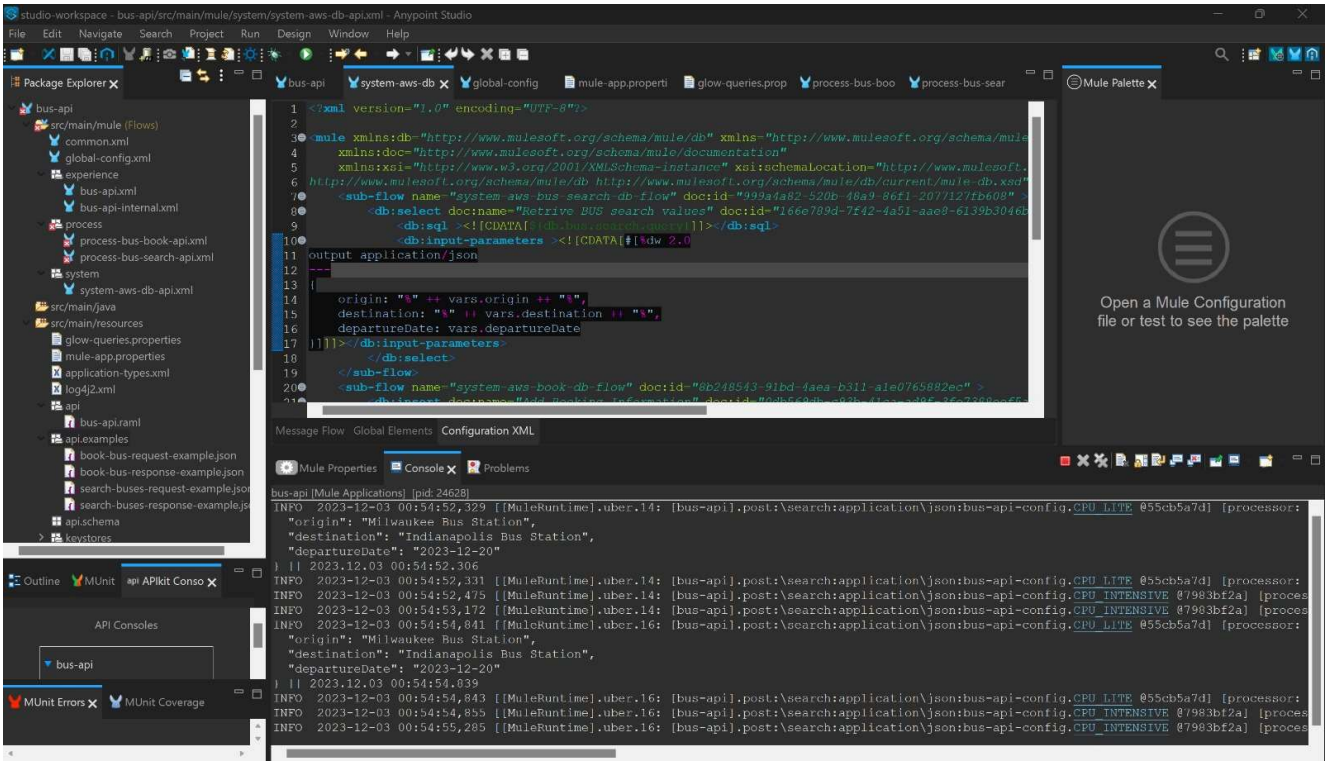
4.1.1 Design a RAML in MuleSoft Design Center :

The design process encompasses the BUS Book API within the MuleSoft Design Center. Create a distinct RAML design outlining endpoints and structures specific to the booking process. Leverage the MuleSoft Design Center to create a dedicated RAML design for the BUS Search API. Define the API's structure, specifying endpoints for searching buses based on criteria such as origin, destination, and departure times.



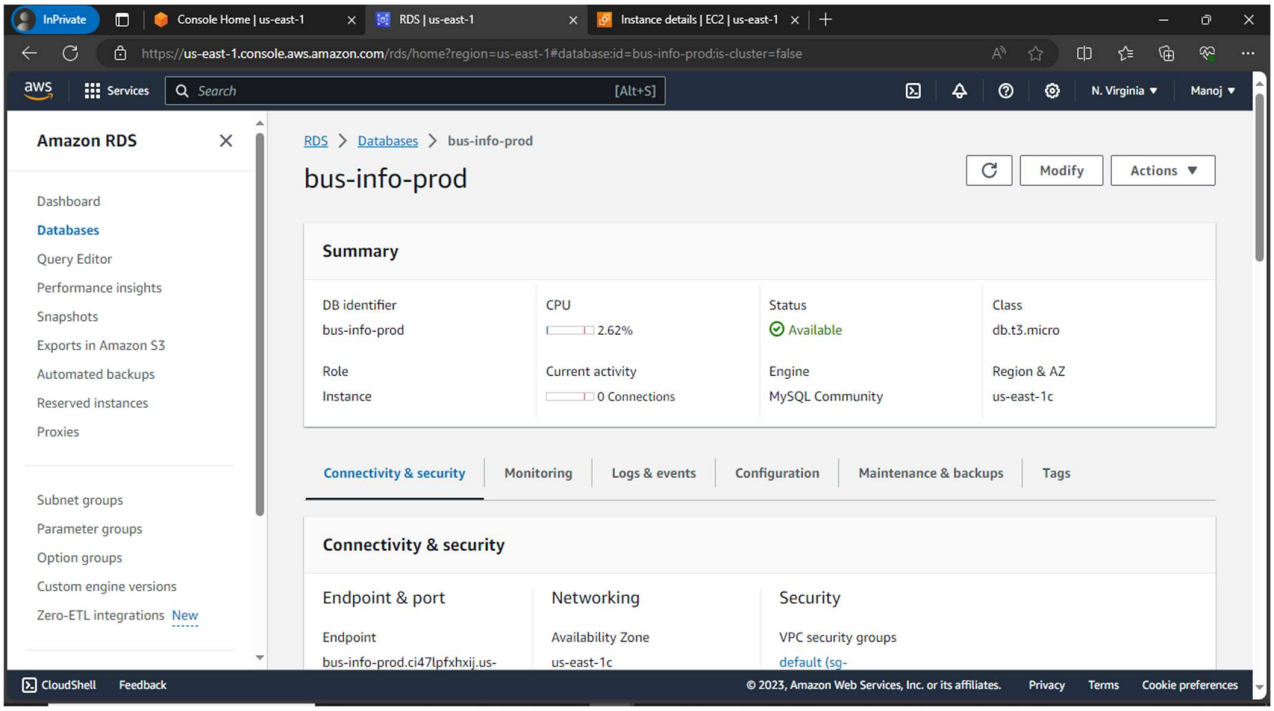
4.1.2 Develop Rest API in Anypoint Sutdio:

Import the designed RAML for the BUS Search API into Anypoint Studio. Develop the necessary logic to enable users to search for available buses based on specified criteria. Implement error handling and conduct iterative testing to validate the search functionality. Similarly, Import the designed RAML for the BUS Book API into Anypoint Studio. Implement the logic required for users to initiate and complete the bus booking process. Integrate validation checks and conduct thorough testing to ensure a seamless booking experience.

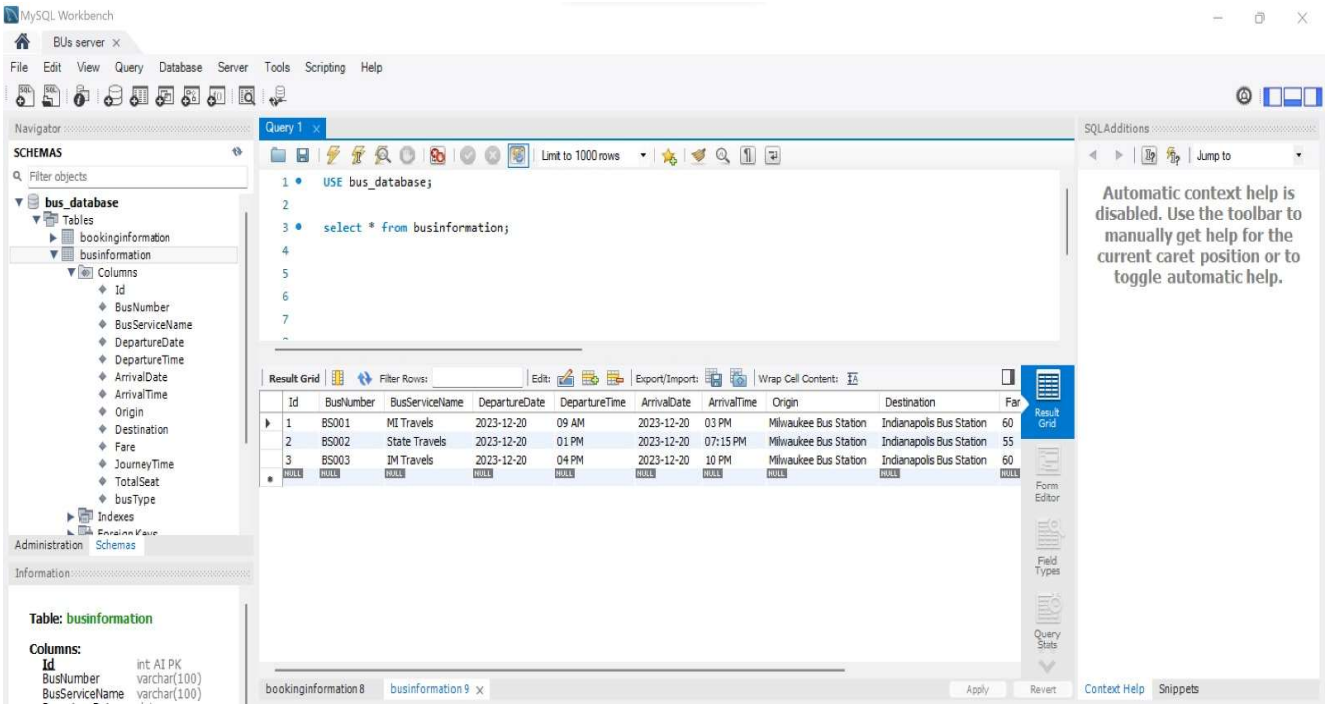


4.1.3 Create an AWS RDS database and design Tables:

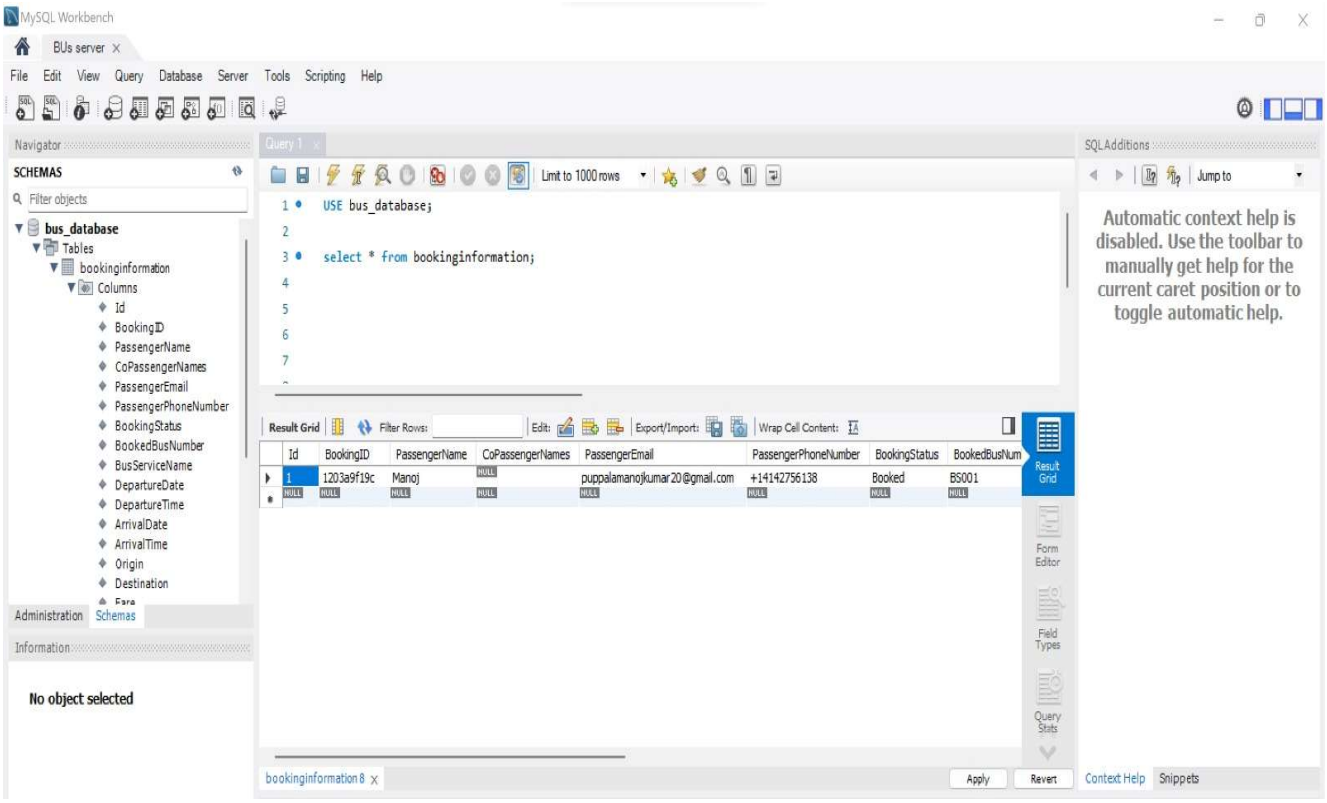
Establish a common AWS RDS database to store shared information relevant to both BUS Search and BUS Book functionalities. All bus-related information, including BusNumbers, Departure and Arrival Times, Origins, Destinations, Fares, and Journey Times, will be stored in an AWS RDS database using MySQL.



The BusInformation table serves as a core data source for the bus search API, enabling users to search for available bus services based on various parameters such as origin, destination, and departureDate.

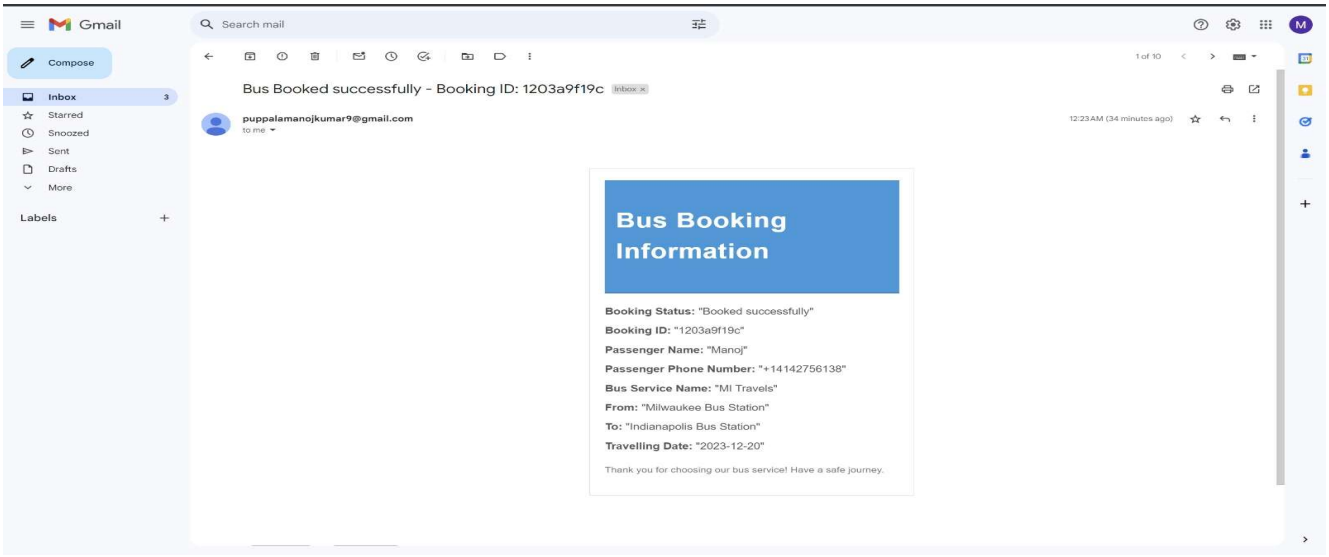


The BookingInformation table serves as the central repository for storing comprehensive details about all bus bookings made within the system. This table is crucial for the functionality of the bus booking API, recording information such as passenger details, seat reservations, journey specifics, and associated metadata.



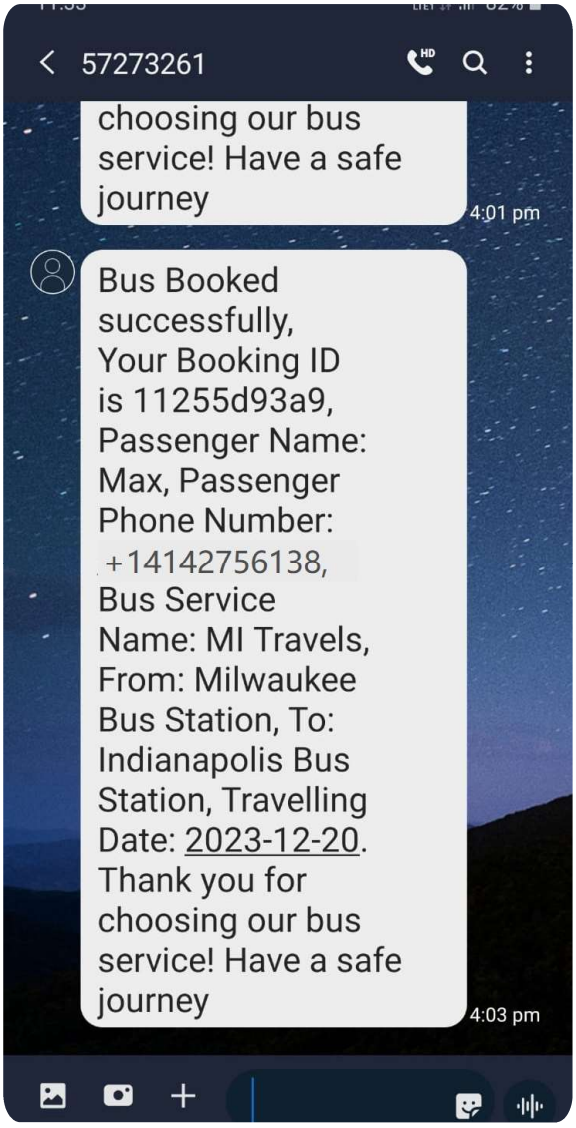
4.1.4 Integrate Simple Mail Transfer Protocol

Implement SMTP for efficient email notifications related to both BUS Search and BUS Book functionalities. Configure the API to send comprehensive email updates, such as booking confirmations or itinerary details. Conduct rigorous testing to validate the reliability and effectiveness of email notifications.



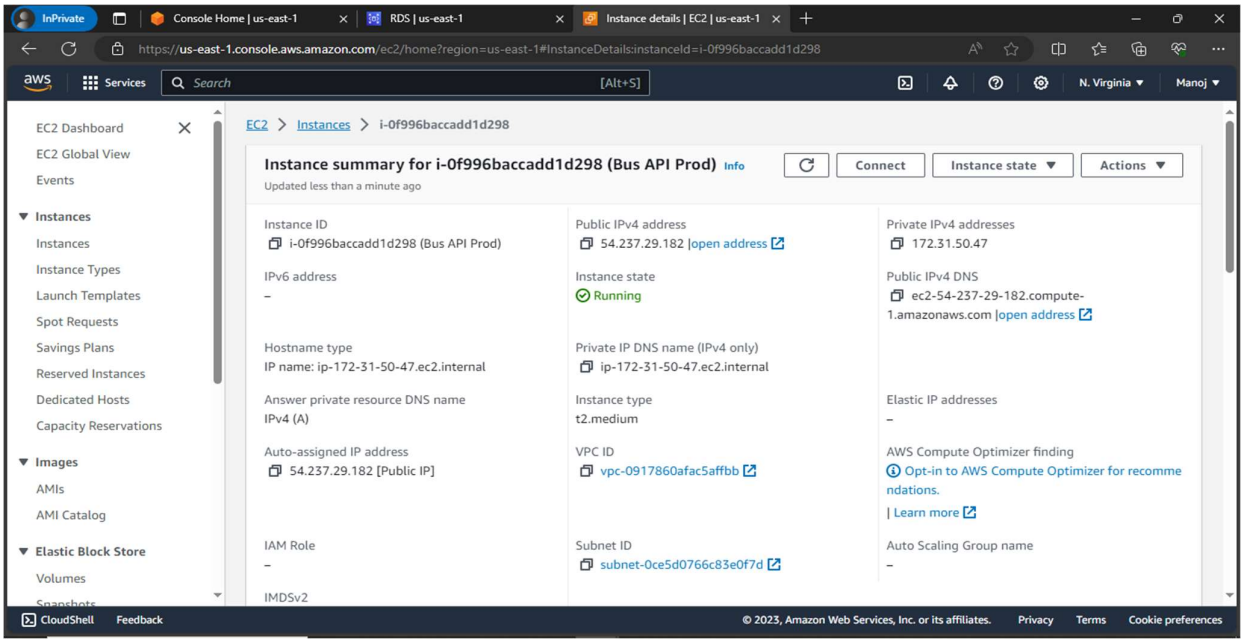
4.1.5 Create an account and connect Twilio services

Integrate Twilio to facilitate real-time SMS notifications for both BUS Search and BUS Book processes. Configure Twilio APIs to send relevant updates to users, such as search results or booking confirmations. Conduct end-to-end testing to ensure timely and accurate SMS notifications.



4.1.6 Create AWS Elastic Compute Cloud

Deploy both the BUS Search and BUS Book APIs on AWS Elastic Compute Cloud (EC2). Configure the deployment environment to ensure scalability, security, and high availability for both functionalities. Conduct performance testing to validate responsiveness under varying loads. This deployment ensures accessibility over the internet, allowing users to interact with the APIs seamlessly.



4.1.7 Install and run Java, apache maven and Mule runtime engine:

Deploying a Bus API project on an AWS EC2 instance involves several steps, including setting up the EC2 instance, configuring the environment, and deploying your Mule application. Here's a step-by-step process:

- Connect AWS EC2 Instance using SSH.

ssh ubuntu@17.60.89.24 -i busAPI-instance-prod-credentials.pem

- Install JAVA and set up JAVA_HOME environment variable.

sudo apt install openjdk-8-jdk

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

echo $JAVA_HOME

- Install apache maven and set up MAVEN_HOME environment variable.

sudo apt install maven

export MAVEN_HOME=/usr/share/maven

echo $MAVEN_HOME

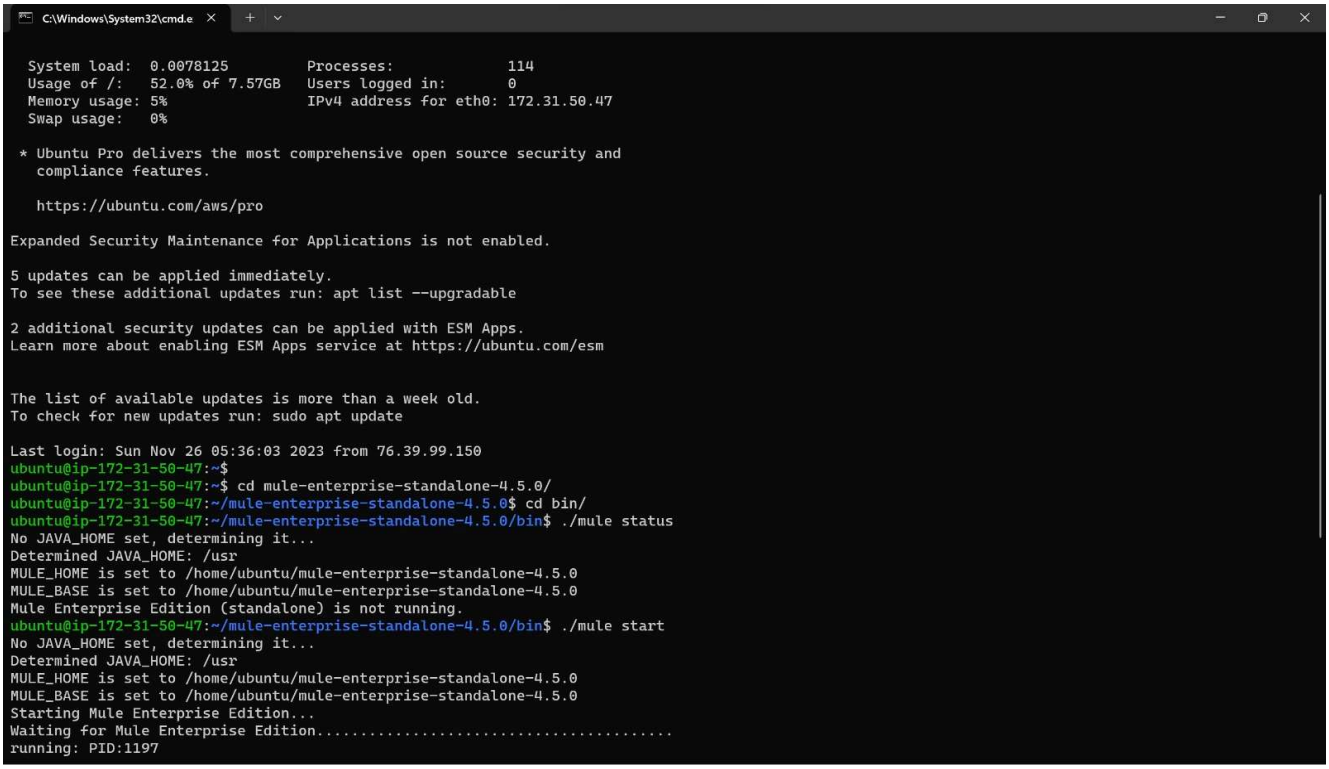
- Download Mule 4 Runtime engine.

Wget https://repository.mulesoft.org/nexus/content/repositories/releases/org/mule/distributions/mule-standalone/4.x.x/mule-standalone-4.x.x.tar.gz

tar -zxvf mule-standalone-4.x.x.tar.gz

- Upload the BUS API JAR file to apps and run Mule server in bin.

./mule start

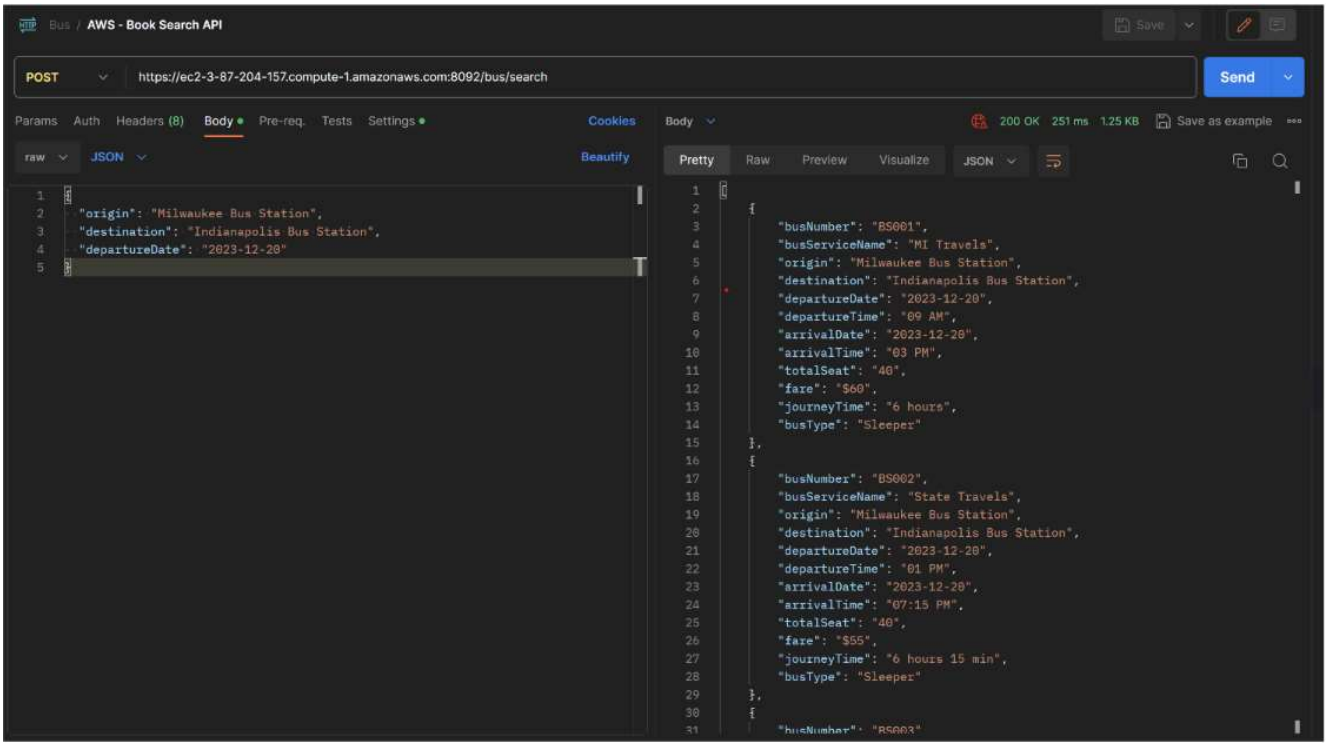


4.2 Testing API in Postman:

Testing the BUS Search and BUS Book APIs using Postman involves creating and executing test cases to ensure the functionality, reliability, and performance of the APIs.

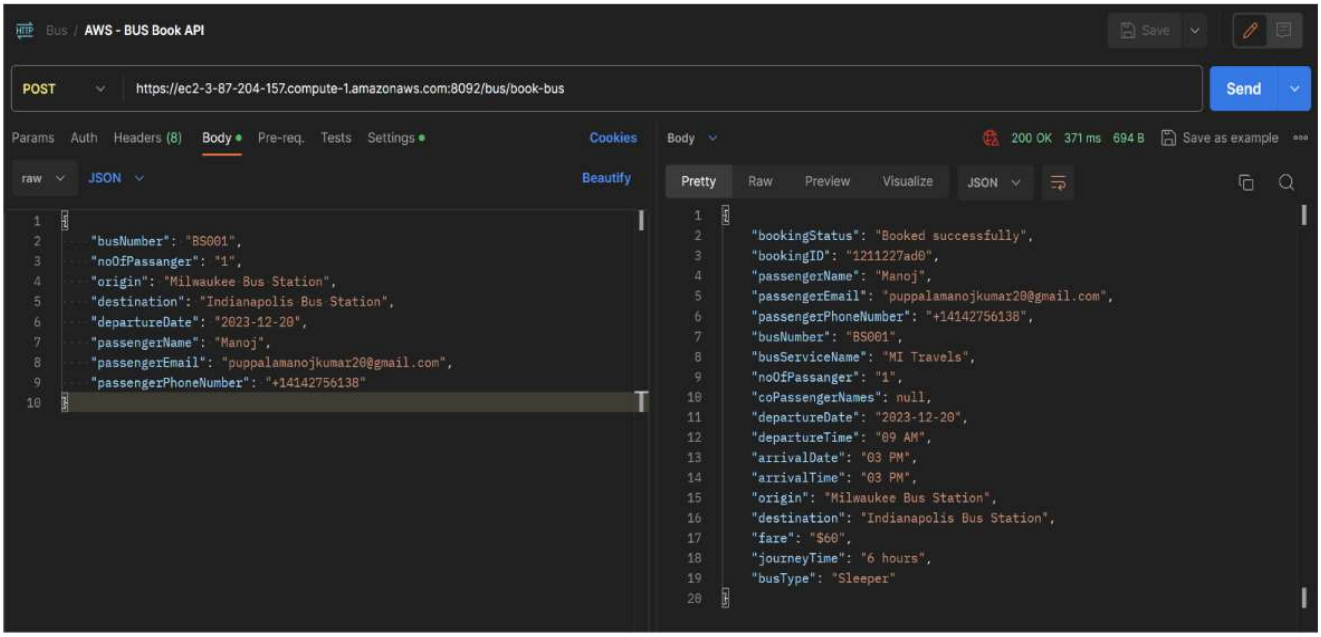
4.2.1 Bus Search API – Test suite

The BUS Search API undergoes comprehensive testing, covering positive scenarios for valid searches, negative scenarios for handling invalid inputs, and boundary testing for date and time parameters.



4.2.2 Bus Booking API – Test suite

Testing the BUS Booking API involves a series of critical scenarios to ensure its functionality, security, and performance. This encompasses verifying the accuracy of bus search results, validating the booking process, and confirming real-time information updates.



CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENTS

5.1 Conclusion

In conclusion, the Bus Search and Booking API project has been meticulously executed with a well-structured implementation plan. Leveraging the RESTful API Modeling Language (RAML) in MuleSoft Design Center, the API's structure and endpoints were defined. This blueprint was seamlessly imported into Anypoint Studio for the implementation of bus-related functionalities. Key bus information is stored securely in an AWS RDS database, while the API itself is deployed on AWS Elastic Compute Cloud (EC2) for accessibility. Integration with Twilio facilitates real-time SMS notifications, and the use of SMTP ensures comprehensive communication through email channels. The culmination of these components results in an innovative and user-friendly solution, offering a streamlined bus booking experience with robust backend support and multi-channel notification capabilities.

5.2 Future Enhancement

To further enhance the Bus Search and Booking API, consider exploring additional features and integrations. Implementing a user authentication system could provide personalized services and improve security. Integration with a payment gateway can enable seamless online transactions for bus bookings. Additionally, incorporating real-time tracking functionalities and providing a user-friendly mobile application could elevate the user experience. Continuous monitoring and optimization of the API's performance can ensure scalability and responsiveness. Furthermore, exploring integration with mapping services for route visualization and dynamic fare adjustments based on demand and occupancy could add valuable features for both users and administrators. Regular updates and feature expansions will contribute to the long-term success and relevance of the Bus API.

REFERENCES

5.2.1 https://dzone.com/articles/running-mule-application-in-aws

5.2.2 https://learning.postman.com/docs/getting-started/overview/

5.2.3 https://docs.aws.amazon.com/ec2/

5.2.4 https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_MySQL.html

5.2.5 https://docs.mulesoft.com/email-connector/latest/email-send

5.2.6 https://docs.mulesoft.com/twilio-connector/4.2/twilio-connector-reference#Create20100401AccountsAddressesjsonByAccountSid

5.2.7 https://docs.mulesoft.com/twilio-connector/4.2/

APPENDIX II

CODING

global-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns:email="http://www.mulesoft.org/schema/mule/email"
xmlns:sockets="http://www.mulesoft.org/schema/mule/sockets"
        xmlns:twilio="http://www.mulesoft.org/schema/mule/twilio"
        xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core"
xmlns:db="http://www.mulesoft.org/schema/mule/db"
xmlns:tls="http://www.mulesoft.org/schema/mule/tls"
xmlns:apikit="http://www.mulesoft.org/schema/mule/mule-apikit"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
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/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/mule-apikit
http://www.mulesoft.org/schema/mule/mule-apikit/current/mule-apikit.xsd
http://www.mulesoft.org/schema/mule/tls
http://www.mulesoft.org/schema/mule/tls/current/mule-tls.xsd
http://www.mulesoft.org/schema/mule/db
http://www.mulesoft.org/schema/mule/db/current/mule-db.xsd
http://www.mulesoft.org/schema/mule/ee/core
http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd
http://www.mulesoft.org/schema/mule/twilio
http://www.mulesoft.org/schema/mule/twilio/current/mule-twilio.xsd
http://www.mulesoft.org/schema/mule/sockets
http://www.mulesoft.org/schema/mule/sockets/current/mule-sockets.xsd
```

```
http://www.mulesoft.org/schema/mule/email
http://www.mulesoft.org/schema/mule/email/current/mule-email.xsd">
        <configuration-properties doc:name="Configuration properties"
doc:id="7c3d0782-a19f-43ba-81dd-8cfb628631e2" file="mule-app.properties" />
        <configuration-properties doc:name="Configuration properties"
doc:id="3cec49e9-f4f8-4ba4-92a5-04f6f9992110" file="glow-queries.properties" />
        <apikit:config name="bus-api-config" api="bus-api.raml"
outboundHeadersMapName="outboundHeaders" httpStatusVarName="httpStatus" >
            <apikit:flow-mappings >
                <apikit:flow-mapping resource="/search" action="post" content-
type="application/json" flow-ref="post:₩search:application₩json:bus-api-config" />
                <apikit:flow-mapping resource="/book-bus" action="post" content-
type="application/json" flow-ref="post:₩book-bus:application₩json:bus-api-config" />
            </apikit:flow-mappings>
        </apikit:config>
        <http:listener-config name="httpListenerConfig" doc:name="HTTP Listener config"
doc:id="34ab61c9-3cb6-4864-851b-0ba045ac3403" >
            <http:listener-connection host="0.0.0.0" port="${http.private.port}" />
        </http:listener-config>
        <http:listener-config name="httpsListenerConfig" doc:name="HTTP Listener
config" doc:id="66cb9c39-02aa-4cab-a226-573d2598f5a6" >
            <http:listener-connection protocol="HTTPS" host="0.0.0.0"
port="${https.private.port}" >
                <tls:context >
                    <tls:key-store type="jks" path="keystores/keystore.jks"
alias="mule" keyPassword="${ssl.keystore.password}" password="${ssl.keystore.password}"
/>
                </tls:context>
            </http:listener-connection>
        </http:listener-config>
        <db:config name="bus_Database_Config" doc:name="Database Config"
doc:id="62156fd6-bf7e-4816-985f-f0733a64f95d" >
            <db:my-sql-connection host="${db.bus.mysql.host}"
```

```
port="${db.bus.mysql.port}" user="${db.bus.mysql.user}"
password="${db.bus.mysql.password}" database="${db.bus.mysql.database}"/>
		</db:config>
		<twilio:config name="Twilio_Connector_Config" doc:name="Twilio Connector
Config" doc:id="8ff1fc24-c60e-4312-8240-1c65959eda8d" >
			<twilio:account-sid-auth-token-connection username="${Twilio.username}"
password="${Twilio.password}" baseUri="https://api.twilio.com/"/>
		</twilio:config>
		<email:smtp-config name="Email_SMTP" doc:name="Email SMTP"
doc:id="f0b76a88-c284-4302-a462-6f3efc271f76" >
			<email:smtp-connection host="smtp.gmail.com" port="587"
user="${gmail.username}" password="${gmail.password}" >
				<email:properties >
					<email:property key="mail.smtp.starttls.enable" value="true" />
				</email:properties>
			</email:smtp-connection>
		</email:smtp-config>
		<error-handler name="Bus-API-Error-Handler" doc:id="97606642-fba4-42e1-
b8bb-5b5c1cb02de1" >
			<on-error-propagate enableNotifications="true" logException="true"
doc:name="Generic Error" doc:id="97b8531c-1e3b-4eec-9342-ac2c30a12be8" >
				<ee:transform doc:name="Frame Generic Error Response"
doc:id="5f664aa1-cd55-479f-af5e-6781e3b4a8d6" >
					<ee:message >
						<ee:set-payload ><![CDATA[%dw 2.0
output application/json
---
{
	errorMessage: error.description default "Internal Server Error"
}]]></ee:set-payload>
					</ee:message>
					<ee:variables >
						<ee:set-variable variableName="httpStatus"
```

><![CDATA[%dw 2.0
output application/java

500]]></ee:set-variable>
</ee:variables>
</ee:transform>
</on-error-propagate>
</error-handler>
</mule>

bus-api.xml

<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:apikit="http://www.mulesoft.org/schema/mule/mule-apikit"
xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
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/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/mule-apikit
http://www.mulesoft.org/schema/mule/mule-apikit/current/mule-apikit.xsd
http://www.mulesoft.org/schema/mule/ee/core
http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd ">

<flow name="bus-api-main">
<http:listener path="${api.bus.services.name.path}" config-ref="httpsListenerConfig">
<http:response statusCode="#[vars.httpStatus default 200]">
<http:headers><![CDATA[#[vars.outboundHeaders default
{}]]]></http:headers>

```
            </http:response>
            <http:error-response statusCode="#[vars.httpStatus default 500]">
                <http:body><![CDATA[#[payload]]]></http:body>
                <http:headers><![CDATA[#[vars.outboundHeaders default
{}]]]></http:headers>
            </http:error-response>
        </http:listener>
        <apikit:router config-ref="bus-api-config" />
        <error-handler>
            <on-error-propagate type="APIKIT:BAD_REQUEST">
                <ee:transform doc:name="Bad Request">
                    <ee:message>
                        <ee:set-payload><![CDATA[%dw 2.0
output application/json
---
{message: "Bad request"}]]></ee:set-payload>
                    </ee:message>
                    <ee:variables>
                        <ee:set-variable variableName="httpStatus">400</ee:set-variable>
                    </ee:variables>
                </ee:transform>
            </on-error-propagate>
            <on-error-propagate type="APIKIT:NOT_FOUND">
                <ee:transform doc:name="Resource Not Found">
                    <ee:message>
                        <ee:set-payload><![CDATA[%dw 2.0
output application/json
---
{message: "Resource not found"}]]></ee:set-payload>
                    </ee:message>
                    <ee:variables>
                        <ee:set-variable variableName="httpStatus">404</ee:set-variable>
                    </ee:variables>
```

```
            </ee:transform>
        </on-error-propagate>
        <on-error-propagate type="APIKIT:METHOD_NOT_ALLOWED">
            <ee:transform>
                <ee:message>
                    <ee:set-payload><![CDATA[%dw 2.0
output application/json
---
{message: "Method not allowed"}]]></ee:set-payload>
                </ee:message>
                <ee:variables>
                    <ee:set-variable variableName="httpStatus">405</ee:set-variable>
                </ee:variables>
            </ee:transform>
        </on-error-propagate>
        <on-error-propagate type="APIKIT:NOT_ACCEPTABLE">
            <ee:transform doc:name="Not Acceptable">
                <ee:message>
                    <ee:set-payload><![CDATA[%dw 2.0
output application/json
---
{message: "Not acceptable"}]]></ee:set-payload>
                </ee:message>
                <ee:variables>
                    <ee:set-variable variableName="httpStatus">406</ee:set-variable>
                </ee:variables>
            </ee:transform>
        </on-error-propagate>
        <on-error-propagate type="APIKIT:UNSUPPORTED_MEDIA_TYPE">
            <ee:transform doc:name="Unsupported Media Type">
                <ee:message>
                    <ee:set-payload><![CDATA[%dw 2.0
output application/json
```

```
---
{message: "Unsupported media type"}]]></ee:set-payload>
                    </ee:message>
                    <ee:variables>
                        <ee:set-variable variableName="httpStatus">415</ee:set-variable>
                    </ee:variables>
                </ee:transform>
            </on-error-propagate>
            <on-error-propagate type="APIKIT:NOT_IMPLEMENTED">
                <ee:transform doc:name="Not Implemented">
                    <ee:message>
                        <ee:set-payload><![CDATA[%dw 2.0
output application/json
---
{message: "Not Implemented"}]]></ee:set-payload>
                    </ee:message>
                    <ee:variables>
                        <ee:set-variable variableName="httpStatus">501</ee:set-variable>
                    </ee:variables>
                </ee:transform>
            </on-error-propagate>
        </error-handler>
    </flow>
    <flow name="bus-api-console">
        <http:listener config-ref="httpListenerConfig" path="/console/*">
            <http:response statusCode="#[vars.httpStatus default 200]">
                <http:headers><![CDATA[#[vars.outboundHeaders default
{}]]]></http:headers>
            </http:response>
            <http:error-response statusCode="#[vars.httpStatus default 500]">
                <http:body><![CDATA[#[payload]]]></http:body>
                <http:headers><![CDATA[#[vars.outboundHeaders default
{}]]]></http:headers>
```

```
            </http:error-response>
        </http:listener>
        <apikit:console config-ref="bus-api-config" />
        <error-handler>
            <on-error-propagate type="APIKIT:NOT_FOUND">
                <ee:transform>
                    <ee:message>
                        <ee:set-payload><![CDATA[%dw 2.0
output application/json
---
{message: "Resource not found"}]]></ee:set-payload>
                    </ee:message>
                    <ee:variables>
                        <ee:set-variable variableName="httpStatus">404</ee:set-variable>
                    </ee:variables>
                </ee:transform>
            </on-error-propagate>
        </error-handler>
    </flow>
    <flow name="post:₩search:application₩json:bus-api-config" doc:id="7254a81e-22ba-
4f97-a281-cbb0d4994d3a">
        <logger level="INFO" doc:name="Initial Logger" doc:id="5e017621-3cff-47d6-b2cd-
a2e9d38f6b73" message='Message Id : #[correlationId] || Stage 1 : Request received for Bus
Search API || #[payload] || #[now() as String {format:"yyyy.MM.dd HH:mm:ss.SSS" }]' />
                <flow-ref doc:name="Call process-bus-search-flow" doc:id="7c30efe6-5974-
461b-8ca5-e00f2e16a997" name="process-bus-search-flow"/>
                <logger level="INFO" doc:name="Final Logger" doc:id="71a360b4-cc38-4e46-
9cb1-25e242f1760a" message='Message Id : #[correlationId] || BUS Search API End ||
#[now() as String {format:"yyyy.MM.dd HH:mm:ss.SSS" }]' />
                <error-handler ref="Bus-API-Error-Handler" />
    </flow>
        <flow name="post:₩book-bus:application₩json:bus-api-config"
doc:id="6110b58b-e135-4a3c-bac8-f55c8096599b" >
```

```
        <logger level="INFO" doc:name="Initial Logger" doc:id="0d8b6bdd-e71d-
4849-8aee-d007ee22aaa4" message='Message Id : #[correlationId] || Stage 1 : Request
received for Bus Book API || #[payload] || #[now() as String {format:"yyyy.MM.dd
HH:mm:ss.SSS" }]' />
        <flow-ref doc:name="Call process-bus-book-flow" doc:id="f7aceaa1-78b2-
462d-b9d9-78b3f5fc4acf" name="process-bus-book-flow"/>
        <logger level="INFO" doc:name="Final Logger" doc:id="f3c6d19d-6867-492f-
8a43-6ea518fb2245" message='Message Id : #[correlationId] || BUS Book API End || #[now()
as String {format:"yyyy.MM.dd HH:mm:ss.SSS" }]' />
        <error-handler ref="Bus-API-Error-Handler" />
    </flow>
</mule>
```

process-bus-search-api.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core"
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/ee/core
http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd">
    <sub-flow name="process-bus-search-flow" doc:id="f005c28c-8623-4c58-a93e-
9234348d93d6" >
        <logger level="INFO" doc:name="Stage Logger" doc:id="3a05f567-37ca-4a09-9426-
a4fe8b16cb71" message='Message Id : #[correlationId] ||  Process BUS search API || #[now() as
String {format:"yyyy.MM.dd HH:mm:ss.SSS" }]' />
        <ee:transform doc:name="Store Required Values" doc:id="795e603e-828c-4674-
935d-6f8fe679df09" >
            <ee:message >
            </ee:message>
            <ee:variables >
                <ee:set-variable variableName="origin" ><![CDATA[%dw 2.0
```

```
output application/java
---
payload.origin]]></ee:set-variable>
                        <ee:set-variable variableName="destination" ><![CDATA[%dw 2.0
output application/java
---
payload.destination]]></ee:set-variable>
                        <ee:set-variable variableName="departureDate" ><![CDATA[%dw 2.0
output application/java
---
payload.departureDate]]></ee:set-variable>
                    </ee:variables>
                </ee:transform>
                <ee:transform doc:name="Set Search Request Error Message" doc:id="1bac3b09-
b0f6-458a-9cee-8e70c7562b16" >
                    <ee:message >
                    </ee:message>
                    <ee:variables >
                        <ee:set-variable variableName="error" ><![CDATA[%dw 2.0
output application/json
---
[
        ({
            "errorMessage": "Required Value is Missing for [origin]"
        }) if (isEmpty(vars.origin)),
        ({
            "errorMessage": "Invalid Value Received for [origin]. Allowed String"
        }) if (!isEmpty(vars.origin) and (typeOf(vars.origin) as String != "String")),
   ({
            "errorMessage": "Required Value is Missing for [destination]"
        }) if (isEmpty(vars.destination)),
        ({
            "errorMessage": "Invalid Value Received for [destination]. Allowed String"
        }) if (!isEmpty(vars.destination) and (typeOf(vars.destination) as String != "String")),
   ({
            "errorMessage": "Required Value is Missing for [departureDate]"
```

```
        }) if (isEmpty(vars.departureDate)),
        ({
            "errorMessage": "Invalid Value Received for [departureDate]. Allowed : YYYY-MM-DD"
        }) if (!isEmpty(vars.departureDate) and (typeOf(vars.departureDate) as String != "String"))
]]]></ee:set-variable>
                    </ee:variables>
                </ee:transform>
                <choice doc:name="Validate ErrorMessage" doc:id="f16f663f-ffd3-44f6-8054-
b4ee35409f31" >
                    <when expression="#[sizeOf(vars.error) == 0]">
                        <logger level="INFO" doc:name="Success Logger" doc:id="01b32a7b-
5701-43bf-bbc9-4403452e7654" message='Message Id : #[correlationId] || BUS search Request
Validation succes || #[now() as String {format:"yyyy.MM.dd HH:mm:ss.SSS" }]'/>
                        <flow-ref doc:name="Call system-aws-bus-search-db-flow"
doc:id="d2fa4959-d987-4f6f-b3d1-f1b2537c6af2" name="system-aws-bus-search-db-flow"/>
                        <choice doc:name="Check for BUS Response" doc:id="48252183-a806-
484e-9c91-0cbaab5debd3" >
                            <when expression="#[!isEmpty(payload)]">
                                <ee:transform doc:name="Frame BUS Search Response"
doc:id="109900ba-7737-4402-99a0-d283b62db989">
                            <ee:message>
                                <ee:set-payload><![CDATA[%dw 2.0
output application/json
fun formateDate(dateValue) = if (!isEmpty(dateValue)) (dateValue as DateTime as String
{format:"yyyy-MM-dd"}) else dateValue
---
(payload default []) map ((busValue, busIndex) -> {
    "busNumber":busValue.BusNumber,
    "busServiceName":busValue.BusServiceName,
    "origin":busValue.Origin,
    "destination":busValue.Destination,
    "departureDate":formateDate(busValue.DepartureDate),
    "departureTime":busValue.DepartureTime,
    "arrivalDate":formateDate(busValue.ArrivalDate),
    "arrivalTime":busValue.ArrivalTime,
    "totalSeat":busValue.TotalSeat,
```

```
    "fare": "₩$" ++ (busValue.Fare  default "0"),
    "journeyTime": busValue.JourneyTime,
    "busType":busValue.busType
})]]></ee:set-payload>
                                        </ee:message>
                                    </ee:transform>
                                        </when>
                                        <otherwise >
                                            <ee:transform doc:name="Set BUS Search API Error
Response" doc:id="4acbf736-6e54-4fc3-90fd-3df057703599" >
                                                <ee:message >
                                                    <ee:set-payload ><![CDATA[%dw 2.0
output application/json
---
{
            errorMessage: "No Bus Found! For The Provided Information"
}]]></ee:set-payload>
                                                </ee:message>
                                                <ee:variables >
                                                    <ee:set-variable
variableName="httpStatus" ><![CDATA[%dw 2.0
output application/java
---
400]]></ee:set-variable>
                                                </ee:variables>
                                            </ee:transform>
                                        </otherwise>
                                    </choice>
                                </when>
                                <otherwise >
                                    <logger level="INFO" doc:name="Failure Logger" doc:id="baa705eb-
66e1-46aa-8dca-844b807728fe" message='Message Id : #[correlationId] ||  Error in BUS search API
Request || #[now() as String {format:"yyyy.MM.dd HH:mm:ss.SSS" }]'/>
                                    <ee:transform doc:name="Set BUS Search API Error Response"
doc:id="4d27f9d7-20ca-43dc-960f-51506d8a8cb4" >
                                        <ee:message >
```

```
                                        <ee:set-payload ><![CDATA[%dw 2.0
output application/json
---
{
        errorMessage: vars.error.errorMessage
}]]></ee:set-payload>
                                    </ee:message>
                                    <ee:variables >
                                        <ee:set-variable variableName="httpStatus"
><![CDATA[%dw 2.0
output application/java
---
400]]></ee:set-variable>
                                    </ee:variables>
                                </ee:transform>
                            </otherwise>
                        </choice>
                    </sub-flow>
</mule>
```

process-bus-book-api.xml

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

<mule xmlns:email="http://www.mulesoft.org/schema/mule/email"
xmlns:twilio="http://www.mulesoft.org/schema/mule/twilio"
        xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core"
        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/ee/core
http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd
http://www.mulesoft.org/schema/mule/twilio
http://www.mulesoft.org/schema/mule/twilio/current/mule-twilio.xsd
```

```
http://www.mulesoft.org/schema/mule/email
http://www.mulesoft.org/schema/mule/email/current/mule-email.xsd">
        <sub-flow name="process-bus-book-flow" doc:id="a6bf99cd-d81a-457e-aea0-
94a25441b12f" >
            <ee:transform doc:name="Store Required Values" doc:id="78a7630a-6ed3-44a5-
a02e-5fe61dc66ff6" >
                <ee:message >
                </ee:message>
                <ee:variables >
                    <ee:set-variable variableName="busNumber" ><![CDATA[%dw 2.0
output application/java
---
payload.busNumber]]></ee:set-variable>
                    <ee:set-variable variableName="noOfPassanger" ><![CDATA[%dw 2.0
output application/java
---
payload.noOfPassanger]]></ee:set-variable>
                    <ee:set-variable variableName="passengerName" ><![CDATA[%dw 2.0
output application/java
---
payload.passengerName]]></ee:set-variable>
                    <ee:set-variable variableName="passengerEmail" ><![CDATA[%dw 2.0
output application/java
---
payload.passengerEmail]]></ee:set-variable>
                    <ee:set-variable variableName="passengerPhoneNumber"
><![CDATA[%dw 2.0
output application/java
---
payload.passengerPhoneNumber]]></ee:set-variable>
                    <ee:set-variable variableName="origin" ><![CDATA[%dw 2.0
output application/java
---
payload.origin]]></ee:set-variable>
                    <ee:set-variable variableName="destination" ><![CDATA[%dw 2.0
output application/java
```

```
---
payload.destination]]></ee:set-variable>
                                <ee:set-variable variableName="departureDate" ><![CDATA[%dw 2.0
output application/java
---
payload.departureDate]]></ee:set-variable>
                                <ee:set-variable variableName="coPassengerNames" ><![CDATA[%dw
2.0
output application/java
---
payload.CoPassengerNames]]></ee:set-variable>
                        </ee:variables>
                </ee:transform>
                <ee:transform doc:name="Set Search Request Error Message" doc:id="cecf52fb-a014-
42bc-a476-d8bc060538c6" >
                        <ee:message />
                        <ee:variables >
                                <ee:set-variable variableName="error" ><![CDATA[%dw 2.0
output application/json
---
[
        ({
            "errorMessage": "Required Value is Missing for [busNumber]"
        }) if (isEmpty(vars.busNumber)),
        ({
            "errorMessage": "Invalid Value Received for [busNumber]. Allowed String"
        }) if (!isEmpty(vars.busNumber) and (typeOf(vars.busNumber) as String != "String")),
        ({
            "errorMessage": "Required Value is Missing for [noOfPassanger]"
        }) if (isEmpty(vars.noOfPassanger)),
        ({
            "errorMessage": "Invalid Value Received for [noOfPassanger]. Allowed String"
        }) if (!isEmpty(vars.noOfPassanger) and (typeOf(vars.noOfPassanger) as String != "String")),
        ({
            "errorMessage": "Required Value is Missing for [passengerName]"
        }) if (isEmpty(vars.passengerName)),
```

```
        ({
            "errorMessage": "Invalid Value Received for [passengerName]. Allowed String"
        }) if (!isEmpty(vars.passengerName) and (typeOf(vars.passengerName) as String !=
"String")),
        ({
            "errorMessage": "Required Value is Missing for [passengerEmail]"
        }) if (isEmpty(vars.passengerEmail)),
        ({
            "errorMessage": "Invalid Value Received for [passengerEmail]. Allowed String"
        }) if (!isEmpty(vars.passengerEmail) and (typeOf(vars.passengerEmail) as String !=
"String")),
        ({
            "errorMessage": "Required Value is Missing for [passengerPhoneNumber]"
        }) if (isEmpty(vars.passengerPhoneNumber)),
        ({
            "errorMessage": "Invalid Value Received for [passengerPhoneNumber]. Allowed
String"
        }) if (!isEmpty(vars.passengerPhoneNumber) and (typeOf(vars.passengerPhoneNumber) as
String != "String")),
        ({
            "errorMessage": "Required Value is Missing for [origin]"
        }) if (isEmpty(vars.origin)),
        ({
            "errorMessage": "Invalid Value Received for [origin]. Allowed String"
        }) if (!isEmpty(vars.origin) and (typeOf(vars.origin) as String != "String")),
    ({
            "errorMessage": "Required Value is Missing for [destination]"
        }) if (isEmpty(vars.destination)),
        ({
            "errorMessage": "Invalid Value Received for [destination]. Allowed String"
        }) if (!isEmpty(vars.destination) and (typeOf(vars.destination) as String != "String")),
    ({
            "errorMessage": "Required Value is Missing for [departureDate]"
        }) if (isEmpty(vars.departureDate)),
        ({
            "errorMessage": "Invalid Value Received for [departureDate]. Allowed : YYYY-MM-DD"
```

```
        }) if (!isEmpty(vars.departureDate) and (typeOf(vars.departureDate) as String != "String"))
]]]></ee:set-variable>
                        </ee:variables>
                    </ee:transform>
                    <choice doc:name="Validate ErrorMessage" doc:id="fdbf789f-746d-4c5b-b49a-
c1f6640b780e" >
                        <when expression="#[sizeOf(vars.error) == 0]" >
                            <logger level="INFO" doc:name="Success Logger" doc:id="ffb04a7b-
d05b-48ec-8930-f848475d08af" message='Message Id : #[correlationId] || BUS Book Request
Validation succes || #[now() as String {format:"yyyy.MM.dd HH:mm:ss.SSS" }]' />
                            <ee:transform doc:name="Frame BUS Search API Request"
doc:id="b70d6527-ed12-41fd-be16-f5323ed3c550">
                                <ee:message>
                                    <ee:set-payload><![CDATA[%dw 2.0
output application/json
---
{
  "origin": vars.origin,
  "destination": vars.destination,
  "departureDate": vars.departureDate
}]]></ee:set-payload>
                                </ee:message>
                            </ee:transform>
                            <flow-ref doc:name="Call post:₩search:application₩json:bus-api-
config" doc:id="a1163f4f-680b-47cf-aa87-b39633a8a5e6"
name="post:₩search:application₩json:bus-api-config" />
                            <ee:transform doc:name="Filter based on BusNumber"
doc:id="71db8a5a-88dc-4c92-a8e2-f17fde7258e1">
                                <ee:message>
                                    <ee:set-payload><![CDATA[%dw 2.0
output application/json
---
payload default [] filter ($.busNumber == vars.busNumber)]]></ee:set-payload>
                                </ee:message>
                            </ee:transform>
                            <choice doc:name="Check valid bus information" doc:id="fc6a2d72-
```

```
45f9-4994-9dfe-e34e107cdd07" >
                                    <when expression="#[!isEmpty(payload)]">
                                        <ee:transform doc:name="Store Bus Search Response"
doc:id="46c94df5-0d75-4ad7-8186-fc436c821b40" >
                                            <ee:message >
                                            </ee:message>
                                            <ee:variables >
                                                <ee:set-variable
variableName="busServiceName" ><![CDATA[%dw 2.0
output application/java
---
payload[0].busServiceName]]></ee:set-variable>
                                                <ee:set-variable
variableName="departureTime" ><![CDATA[%dw 2.0
output application/java
---
payload[0].departureTime]]></ee:set-variable>
                                                <ee:set-variable
variableName="arrivalDate" ><![CDATA[%dw 2.0
output application/java
---
payload[0].arrivalDate]]></ee:set-variable>
                                                <ee:set-variable
variableName="arrivalTime" ><![CDATA[%dw 2.0
output application/java
---
payload[0].arrivalTime]]></ee:set-variable>
                                                <ee:set-variable variableName="fare"
><![CDATA[%dw 2.0
output application/java
---
payload[0].fare]]></ee:set-variable>
                                                <ee:set-variable
variableName="JourneyTime" ><![CDATA[%dw 2.0
output application/java
---
```

```
payload[0].journeyTime]]></ee:set-variable>
                                                <ee:set-variable variableName="busType"
><![CDATA[%dw 2.0
output application/java
---
payload[0].busType]]></ee:set-variable>
                                                <ee:set-variable
variableName="bookingID" ><![CDATA[%dw 2.0
output application/java
---
now() as String {format: "MMdd"} ++ (uuid() replace ("-") with "")[0 to 5]]]></ee:set-variable>
                                                <ee:set-variable
variableName="bookingStatus" ><![CDATA[%dw 2.0
output application/java
---
"Booked"]]></ee:set-variable>
                                                <ee:set-variable
variableName="bookedBusNumber" ><![CDATA[%dw 2.0
output application/java
---
payload[0].busNumber]]></ee:set-variable>
                                        </ee:variables>
                                    </ee:transform>
                                    <flow-ref doc:name="Call system-aws-book-db-flow"
doc:id="1720bd27-194e-45d9-9b07-063125ddfbe4" name="system-aws-book-db-flow"/>
                                    <ee:transform doc:name="Bus Book Response"
doc:id="e3623895-af7a-4516-94b5-d1d70ab9e395" >
                                        <ee:message >
                                            <ee:set-payload ><![CDATA[%dw 2.0
output application/json
---
payload]]></ee:set-payload>
                                        </ee:message>
                                    </ee:transform>
                                    <flow-ref doc:name="Call process-bus-book-response-
notification-flow" doc:id="69524eae-7bfb-4988-812a-7a72c81c3707" name="process-bus-book-
```

```
response-notification-flow"/>
                                        </when>
                                        <otherwise >
                                                <logger level="INFO" doc:name="Failure Logger"
doc:id="7cf9c058-a520-40c9-b212-7a87dbb8bb1f" message='Message Id : #[correlationId] ||  Error
in BUS Book API Request || #[now() as String {format:"yyyy.MM.dd HH:mm:ss.SSS" }]' />
                                                <ee:transform doc:name="Set BUS Book API Error
Response" doc:id="ec81587a-9d81-425b-b8ea-0ce259405f29" >
                                                        <ee:message >
                                                                <ee:set-payload ><![CDATA[%dw 2.0
output application/json
---
{
        errorMessage: "Invalid Bus Information Provided"
}]]></ee:set-payload>
                                                        </ee:message>
                                                        <ee:variables >
                                                                <ee:set-variable
variableName="httpStatus" ><![CDATA[%dw 2.0
output application/java
---
400]]></ee:set-variable>
                                                        </ee:variables>
                                                </ee:transform>
                                        </otherwise>
                                </choice>
                        </when>
                        <otherwise >
                                <logger level="INFO" doc:name="Failure Logger" doc:id="98f7a0eb-
6a3c-47ba-90e3-8c411721822d" message='Message Id : #[correlationId] ||  Error in BUS Book API
Request || #[now() as String {format:"yyyy.MM.dd HH:mm:ss.SSS" }]' />
                                <ee:transform doc:name="Set BUS Book API Error Response"
doc:id="41307fe5-afe1-4dc5-9c3e-9e036a15c44c" >
                                        <ee:message >
                                                <ee:set-payload ><![CDATA[%dw 2.0
output application/json
```

```
---
{
	errorMessage: vars.error.errorMessage
}]]></ee:set-payload>
                                    </ee:message>
                                    <ee:variables >
                                        <ee:set-variable variableName="httpStatus"
><![CDATA[%dw 2.0
output application/java
---
400]]></ee:set-variable>
                                    </ee:variables>
                                </ee:transform>
                            </otherwise>
                        </choice>
                    </sub-flow>
                    <sub-flow name="process-bus-book-response-notification-flow" doc:id="6d27c3be-89f9-
45f6-9429-6d686547f202" >
                        <logger level="INFO" doc:name="Stage Logger" doc:id="2e1ee28f-7452-4441-b0c4-
69c1f46128ca" message='Message Id : #[correlationId] ||  Bus Book API - Response and Notification
|| #[now() as String {format:"yyyy.MM.dd HH:mm:ss.SSS" }]' />
                        <ee:transform doc:name="Frame BUS Book Response API" doc:id="6bc41111-091c-
4b74-9a53-62e3baf9310d" >
                            <ee:message >
                                <ee:set-payload ><![CDATA[%dw 2.0
output application/json
---
{
	"bookingStatus": "Booked successfully",
	"bookingID": vars.bookingID,
	"passengerName": vars.passengerName,
	"passengerEmail": vars.passengerEmail,
	"passengerPhoneNumber": vars.passengerPhoneNumber,
	"busNumber": vars.bookedBusNumber,
	"busServiceName": vars.busServiceName,
	"noOfPassanger": vars.noOfPassanger,
```

```
            "coPassengerNames": vars.coPassengerNames,
            "departureDate": vars.departureDate,
            "departureTime": vars.departureTime,
            "arrivalDate": vars.arrivalTime,
            "arrivalTime": vars.arrivalTime,
            "origin": vars.origin,
            "destination": vars.destination,
            "fare": vars.fare,
            "journeyTime": vars.journeyTime,
            "busType": vars.busType
}]]></ee:set-payload>
                        </ee:message>
                </ee:transform>
                <async doc:name="Async" doc:id="89305c92-904e-4e9c-94b7-37c818e59312" >
                        <flow-ref doc:name="Call process-bus-book-sms-notification-flow"
doc:id="6043fe69-3815-4ce2-81ed-2f28f7a25873" name="process-bus-book-sms-notification-
flow"/>
                </async>
                <async doc:name="Async" doc:id="6cfce7f5-e91f-4699-bd83-999ed3c31fc9" >
                        <flow-ref doc:name="Call process-bus-book-gmail-notification-service-flow"
doc:id="413aeceb-56b4-4ae2-a0f5-17bd6b7273cd" name="process-bus-book-gmail-notification-
service-flow"/>
                </async>
            </sub-flow>
            <sub-flow name="process-bus-book-sms-notification-flow" doc:id="b6c5fcfc-e6a2-4ffe-
a976-4c3c2eda3287" >
                <ee:transform doc:name="Json Payload" doc:id="7d77318d-5474-4b8c-858a-
c209e5733814" >
                        <ee:message >
                                <ee:set-payload ><![CDATA[%dw 2.0
output application/json
---
"Bus Booked successfully, Your Booking ID is " ++ payload.bookingID ++ ", Passenger Name: " ++
payload.passengerName  ++ ", Passenger Phone Number: " ++ payload.passengerPhoneNumber
++ ", Bus Service Name: " ++ payload.busServiceName ++ ", From: " ++ payload.origin ++ ", To: "
++ payload.destination ++ ", Travelling Date: " ++ payload.departureDate ++ ". Thank you for
```

```
choosing our bus service! Have a safe journey. "]]></ee:set-payload>
                        </ee:message>
                </ee:transform>
                <ee:transform doc:name="Frame Twilio Message Request" doc:id="1ad9e3f8-46c1-
477b-8efa-ecdcc507d9fd" >
                        <ee:message >
                                <ee:set-payload ><![CDATA[%dw 2.0
output application/x-www-form-urlencoded
---
{
        From: "+16784006109",
        Body: payload as String ,
        To: vars.passengerPhoneNumber,
}]]></ee:set-payload>
                        </ee:message>
                </ee:transform>
                <logger level="INFO" doc:name="Stage Logger" doc:id="ade15256-c8c1-49ea-ba49-
3227a9f0a21f" message="mssageId : #[correlationId] || Text Message Request: #[payload]"/>
                <twilio:create20100401-accounts-messagesjson-by-account-sid doc:name="Send SMS
notification" doc:id="e0784563-c8d1-4bb9-bb43-0c147a473c38" config-
ref="Twilio_Connector_Config" accountSid="${Twilio.username}"/>
        </sub-flow>
        <sub-flow name="process-bus-book-gmail-notification-service-flow" doc:id="43202cb1-
754e-46d1-a5e1-1a37bde7fdf1" >
                <ee:transform doc:name="Json Payload" doc:id="bf072981-72a4-48dc-88f6-
d4211bdd173a" >
                        <ee:message >
                                <ee:set-payload ><![CDATA[%dw 2.0
output application/json
---
payload]]></ee:set-payload>
                        </ee:message>
                </ee:transform>
                <logger level="INFO" doc:name="Stage Logger" doc:id="f604e0df-d8b2-48d9-8e7a-
2cbb0d85c8b2" message="mssageId : #[correlationId] || Gmail Notification flow" />
                <parse-template doc:name="Send Booking Information Template" doc:id="637df95a-
```

```
84ac-470a-a77f-026710dde757" location="template₩busTicket.html"/>
                <email:send doc:id="337848c2-bed7-4716-a182-d0150c56e112" config-
ref="Email_SMTP" fromAddress="${gmail.username}" subject='#["Bus Booked successfully - Booking
ID: " ++ vars.bookingID]' doc:name="Send Gmail Notification">
                        <email:to-addresses >
                                <email:to-address value="#[vars.passengerEmail]" />
                        </email:to-addresses>
                        <email:body contentType="text/html" >
                        </email:body>
                </email:send>
        </sub-flow>
</mule>
```

system-aws-db-api.xml

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

<mule xmlns:db="http://www.mulesoft.org/schema/mule/db"
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/db http://www.mulesoft.org/schema/mule/db/current/mule-
db.xsd">
        <sub-flow name="system-aws-bus-search-db-flow" doc:id="999a4a82-520b-48a9-86f1-
2077127fb608" >
                <db:select doc:name="Retrive BUS search values" doc:id="166e789d-7f42-4a51-aae8-
6139b3046b95" config-ref="bus_Database_Config">
                        <db:sql ><![CDATA[${db.bus.search.query}]]></db:sql>
                        <db:input-parameters ><![CDATA[#[%dw 2.0
output application/json
---
{
        origin: "%" ++ vars.origin ++ "%",
        destination: "%" ++ vars.destination ++ "%",
```

```
            departureDate: vars.departureDate
}]]]></db:input-parameters>
                    </db:select>
            </sub-flow>
            <sub-flow name="system-aws-book-db-flow" doc:id="8b248543-91bd-4aea-b311-
a1e0765882ec" >
                    <db:insert doc:name="Add Booking Information" doc:id="0db569db-c93b-41ca-ad9f-
3fe7388eef5a" config-ref="bus_Database_Config">
                            <db:sql ><![CDATA[${db.bus.book.insert.query}]]></db:sql>
                            <db:input-parameters ><![CDATA[#[{
bookingID: vars.bookingID,
passengerName:vars.passengerName,
coPassengerNames:vars.coPassengerNames,
passengerEmail:vars.passengerEmail,
passengerPhoneNumber:vars.passengerPhoneNumber,
bookingStatus:vars.bookingStatus,
bookedBusNumber:vars.bookedBusNumber,
busServiceName:vars.busServiceName,
departureDate:vars.departureDate,
departureTime:vars.departureTime,
arrivalDate:vars.arrivalDate,
arrivalTime:vars.arrivalTime,
origin:vars.origin,
destination:vars.destination,
fare:vars.fare,
journeyTime:vars.journeyTime,
busType:vars.busType
}]]]></db:input-parameters>
                    </db:insert>
            </sub-flow>
</mule
```