



PARSHVANATH CHARITABLE TRUST'S
A.P. Shah Institute of Technology
Thane, 400615

Academic Year: 2021-22
Department of Computer Engineering

CSL605 SKILL BASED LAB COURSE: CLOUD COMPUTING

Mini Project Report

- **Title of Project** :Handyman Service Booking Bot
- **Year and Semester** : T.E. (Sem VI)
- **Group Members Name and Roll No.** :
JANHAVI ANAP(03)
NIDHI SINGH(69)
ISHANEE REVANKAR(54)
NIDHI HENIYA(21)

Table of Contents

Sr. No.	Topic	Page No.
1.	Problem Definition	3
2.	Introduction	3
3.	Description	4
4.	Implementation details with screen-shots (stepwise)	7
5.	Learning Outcome	9

Problem Definition:

Booking any type of service for any problem that one faces is tedious since one has to physically visit the shop to book a handyman or a worker. To ease this out, this project aims to build a system or a platform which is a web-based Chatbot for Handyman service booking. Handyman Service Booking Chatbot is designed to let a user book a handyman service in just one click.

Introduction:

A **handyman**, also known as a **fixer**, **handyperson** or **handyworker**, is a person skilled at a wide range of repairs, typically around the home. These tasks include trade skills, repair work, maintenance work, are both interior and exterior, and are sometimes described as "side work", "odd jobs" or "fix-up tasks". Specifically, these jobs could be light plumbing jobs such as fixing a leaky toilet or light electric jobs such as changing a light fixture or bulb. The term *handyman* increasingly describes a paid worker, but it also includes non-paid homeowners or do-it-yourselfers. Many people can do common household repairs. Handyman service booking bot have a User Interface through which users interact with the chatbot. Once the user sends respective service request or utterance the chatbot replies accordingly. It asks for information like name, address, phone number and pin code for the purpose of user booking confirmation. Further it asks for the problem faced by the user. Once the user inputs their problem, related service is booked (on the basis of problem type) after confirmation from the user. Bot books the service via our interactive frontend.

The cloud services used in this project are:

- Amazon Lex
- Amazon EBS

Kommunicate

We are integrating our interactive frontend with the chatbot using Kommunicate

Description:

Cloud services used:

Amazon Lex
AWS Elastic Beanstalk

Software requirements:

For frontend:
HTML
CSS

Integration Tool:

Kommunicate

Amazon Lex:

Amazon Lex is an AWS service for building conversational interfaces for applications using voice and text. Amazon Lex provides the deep functionality and flexibility of natural language understanding (NLU) and automatic speech recognition (ASR) .

It helps to build conversational chatbots quickly. We just have to specify the basic conversation flow in the Amazon Lex console, without the knowledge of deep learning. Amazon Lex manages the dialogue and dynamically adjusts the responses in the conversation. Using the console, you can build, test, and publish your text or voice chatbot. This bots can be later integrated into web applications, mobile applications, etc.

Typical steps performed when working with Amazon Lex:

1. Create a bot and configure it with one or more intents that you want to support. You add the configuration so that the bot is able to understand the user's goal (intent), engage in conversation with the user to elicit information, and, after the user provides the necessary data, fulfill the user's intent.
2. Test the bot. You can use the test window client provided by the Amazon Lex console.
3. Publish a version and create an alias.
4. Deploy the bot. You can deploy the bot on platforms such as mobile applications or messaging platforms such as Facebook Messenger.

Screenshot of the Amazon Lex homepage:

The page features a search bar at the top with the placeholder "Search for services, features, blogs, docs, and more". Below the search bar is a "Get Started" button and a "Getting Started Guide" link.

Amazon Lex

Amazon Lex is a service for building conversational interfaces using voice and text. With Lex, the same deep learning engine that powers Alexa is now available to any developer, enabling you to bring sophisticated, natural language chatbots to your new and existing applications.

High Quality Deep Learning Technologies

Powered by the same technology as Alexa, Lex provides both automatic speech recognition (ASR) and natural language understanding (NLU) technologies to create a Speech Language Understanding (SLU) system. Through SLU, Amazon Lex takes natural language speech and text input, understands the intent, and fulfills the intent of the user.

[Learn more](#)

Seamlessly Deploy and Scale

You can build, test, and deploy your chatbots directly from the AWS Management Console. Lex allows you to easily publish your voice or text chatbots, so you can access them from mobile apps, web apps, and multiple chat services, like Facebook Messenger. Amazon Lex scales automatically so you don't have to worry about scaling your bots.

[Learn more](#)

Built-in Integration with the AWS Platform

Amazon Lex has native interoperability with several AWS services such as Amazon Cognito, AWS Lambda, Amazon DynamoDB, Amazon CloudWatch, and AWS Mobile Hub, so you can take advantage of the power of the AWS platform for security, monitoring, user authentication, business logic, storage and mobile app development.

[Learn more](#)

Amazon Lex documentation & support

[Getting Started Guide](#) | [FAQ](#) | [Developer Guide](#) | [Forums](#) | [Report an Issue](#)

AWS Elastic Beanstalk:

AWS Elastic Beanstalk is used for deploying web applications and services. It handles all the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring. AWS Elastic Beanstalk allows you to quickly deploy applications and services without having to worry about configuring underlying resources, services, operating systems or web servers.

It is pay only for the resources needed to store and run your applications.

Screenshot of the Amazon Elastic Beanstalk homepage:

The page features a search bar at the top with the placeholder "Search for services, features, blogs, docs, and more".

Amazon Elastic Beanstalk

End-to-end web application management.

How it works

You simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, and automatic scaling to web application health monitoring, with ongoing fully managed patch and security updates. [Learn more](#)

Benefits and features

- Easy to get started**: Elastic Beanstalk is the simplest way to deploy and run your web application on Amazon Web Services. Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, automatic scaling, and web
- Complete resource control**: You have the freedom to select the Amazon Web Services resources, such as Amazon EC2 instance types, that are optimal for your web application. Additionally, Elastic Beanstalk lets you manage and retain full control over the Amazon Web Services resources.

Get started

Easily deploy your web application in minutes.

[Create Application](#)

Pricing

There's no additional charge for Elastic Beanstalk. You pay for Amazon Web Services resources that we create to store and run your web application, like Amazon S3 buckets and Amazon EC2 instances.

Getting Started

[Launch a web application](#)

More resources

Kommunicate:

Kommunicate is a human+bot hybrid customer support automation platform that provides real-time, proactive, and personalized support for growing businesses. Kommunicate is used to integrate chatbots with the website.

Lex bot can be integrated into your website in a few simple steps:

Step 1: Create a free Kommunicate account

Step 2: Connect your Amazon Lex bot

Step 3: Give your bot an identity

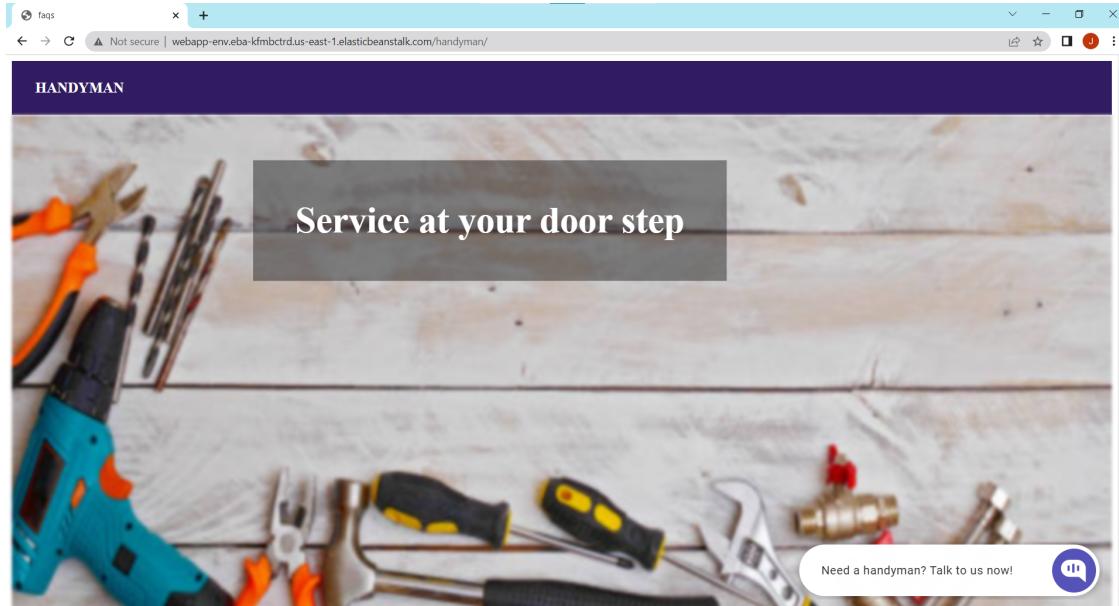
Step 4: Enable/Disable human handoff

Step 5: Assign all the incoming conversations to your Lex bot

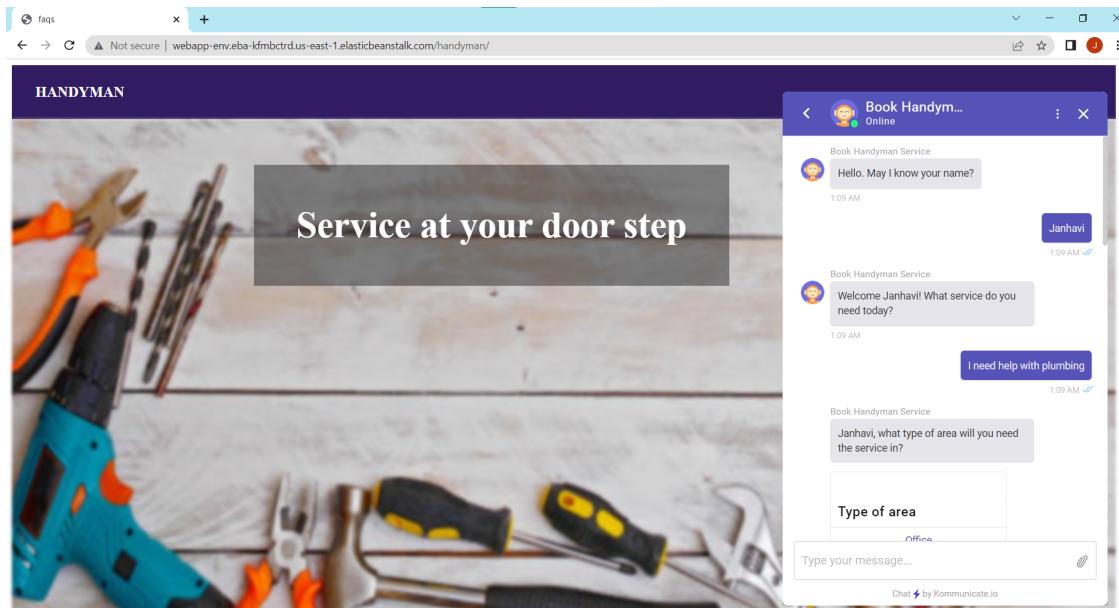
Step 6: Install the Kommunicate on your website by pasting the JavaScript code in the source code

The screenshot shows the Kommunicate dashboard at dashboard.kommunicate.io/settings/install. The left sidebar has a purple navigation bar with icons for messaging, calls, conversations, chat widget, install (which is selected), billing, developer, and download. The main content area has a header "Install" and a sub-header "Follow the steps below to install Kommunicate Chat in your websites and web apps." It shows two options: "Install on your own" (selected) and "Need help to install?". Below this, there's a section titled "Install Kommunicate on your website:" with instructions to copy the provided JavaScript code and paste it just before the closing body tag. A "Copy" button is available for the code. There are also sections for "Other install options:" with fields for "App ID" (3f400acf62a41ad91fc23392540ecaf8) and "API Key" (SJ3bumGOTq1rlKQ5EnsdN7AcVVbc1sC), each with its own "Copy" button. At the bottom, there are sections for "Web" and "Mobile". A footer note says "View documentation for Content Management System (CMS) integration".

Implementation:

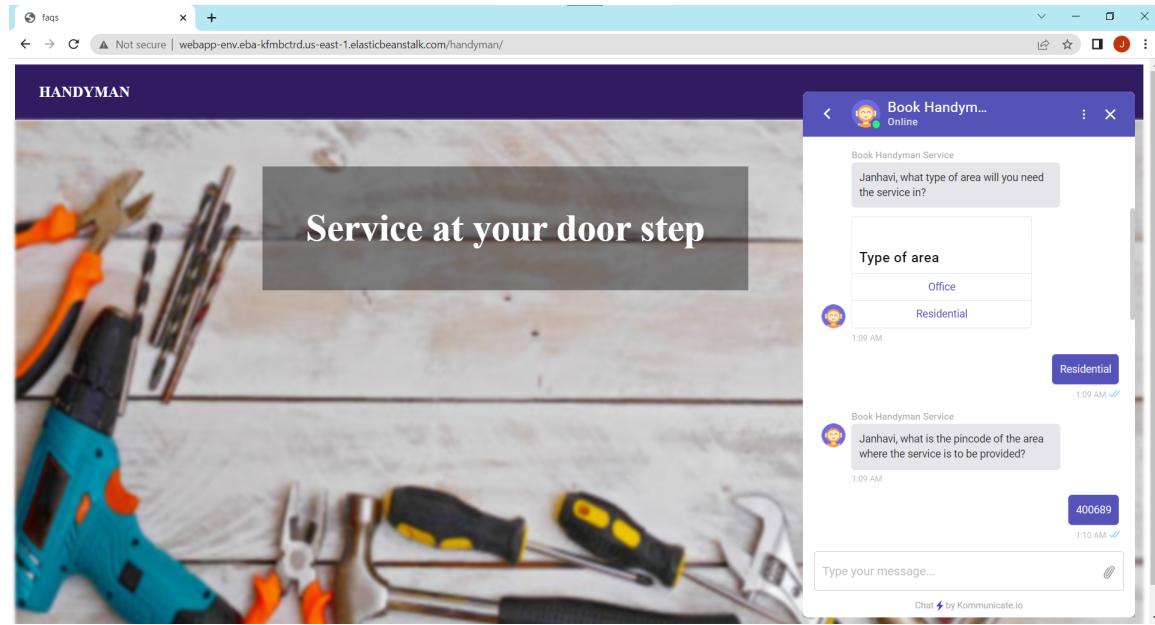


On opening the website link the chatbot pops up text to grab the attention of the user. By clicking on the bot icon the chat window opens.

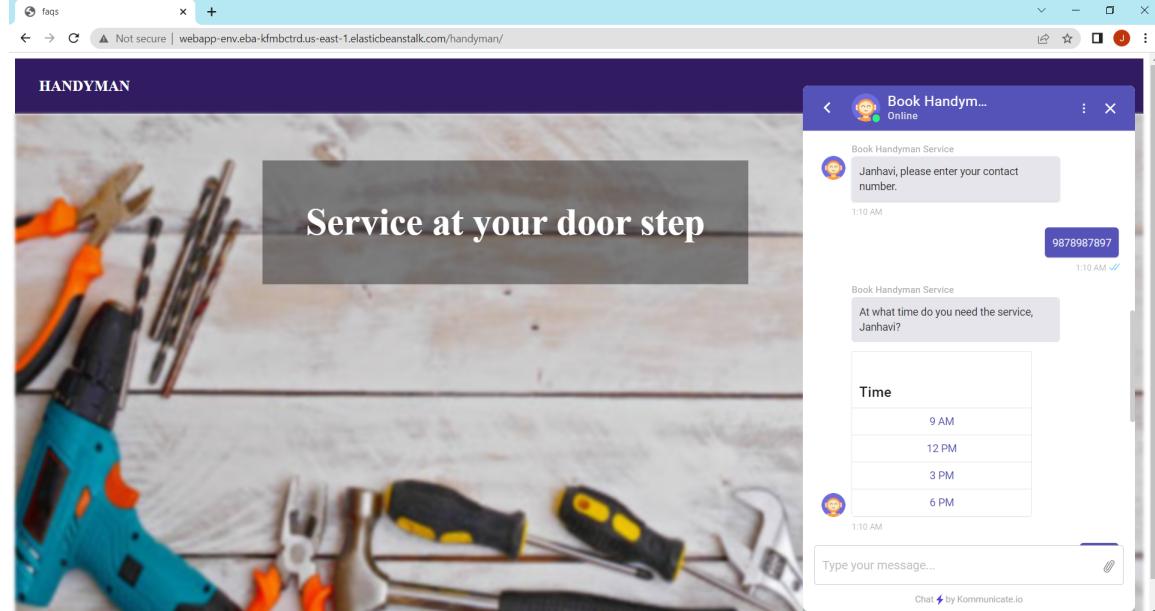


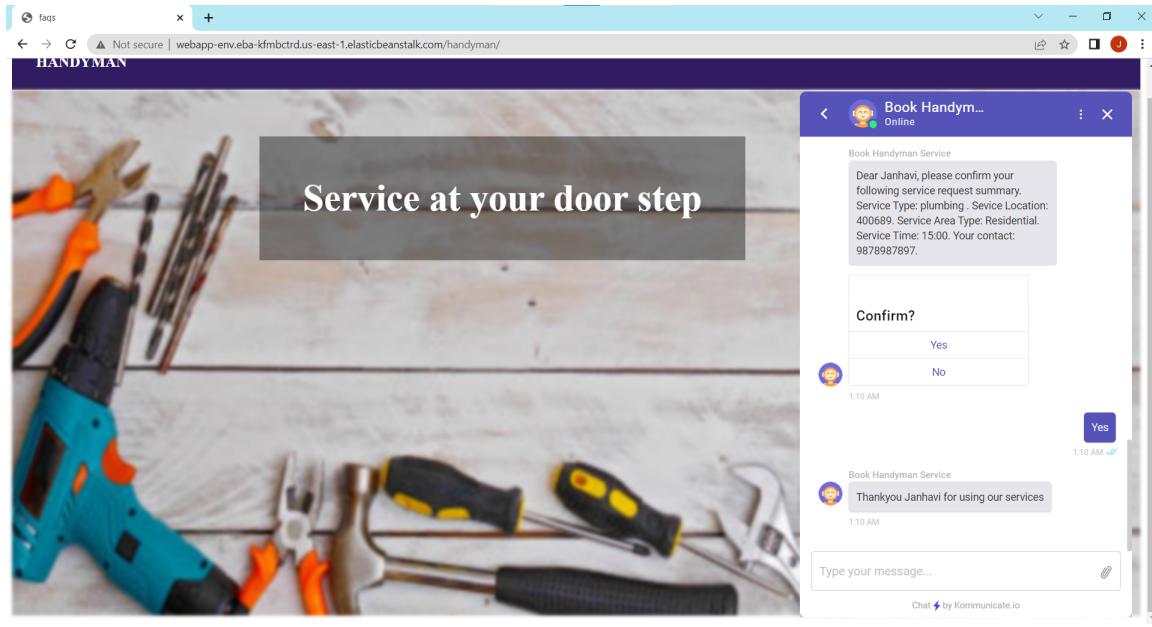
The bot initiates the conversation by first asking the name of the user. It stores that name and addresses all the questions using that name.

It then asks the user what service is needed to which user should reply. The bot detects the keywords and stores the type of service needed.



The bot then asks for the pincode in which the user wants the service in.





Learning outcomes:

We have learnt to use various cloud service like:

- Learnt to use Amazon Lex ,for creating a chatbot
- Used AWS Elastic Beanstalk ,for deploying the web application on cloud

We also learnt to use Kommunicate to integrate the Lex bot with the frontend