**CAPSTONE PROJECT REPORT**

(Project Term January-May 2021)

## AI POWERED CONVERSATIONAL SHOPPING AND HUMAN PLUS BOT HYBRID SUPPORT

Submitted by

**Rishindra Mani Katiyar Registration Number: 11715013**

**Shivmangal Singh Yadav Registration Number: 11709300**

**Project Group Number: KC274**

**Course Code CSE 445**

Under the Guidance of

Miss Ashu

Assistant professor, Lovely Professional University

# School of Computer Science and Engineering





**DECLARATION**

We hereby declare that the project work entitled (“AI Powered Conversational Solution and Human Plus Bot Hybrid Support”) is an authentic record of our own work carried out as requirements of Capstone Project for the award of Bachelor of technology degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Miss Ashu, during January to May 2021. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Project Group Number: KC274

Name of Student 1: Rishindra Mani Katiyar

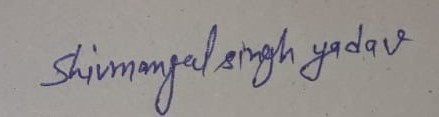
Registration Number: 11715013

Name of Student 2: Shivmangal Singh Yadav

Registration Number: 11709300



(Signature of Student 1)



(Signature of Student 2)

**CERTIFICATE**

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara.

**Signature and Name of the Mentor:** Miss Ashu

**Designation:** Assistant Professor

**School of Computer Science and Engineering,**

Lovely Professional University,

Phagwara, Punjab.

Date : 27-04-2021

**ACKNOWLEDGEMENT**

I take this opportunity to present our votes of thanks to all those guideposts who really acted as lightening pillars to enlighten my way throughout this project which lead to successful and satisfactory completion of this project. The success and outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them. I owe my deepest gratitude to Miss Ashu, my project mentor for providing me an opportunity to do the project and giving me all the support and guidance, which made me to complete the project within the stipulated time. Lastly, I am thankful to all those, particularly my friends, who have been instrumental in creating proper, healthy, and conductive environment and including new and fresh innovative ideas for me during the project. Without their help, it would have been extremely difficult for me to complete the project in a time bound framework.

**TABLE OF CONTENTS**

**Inner first page…………………………………….………………………………(i)**

**PAC form………………………………………….……………………………….(ii)**

**Declaration…...…………………………………………………………………....(iii)**

**Certificate……..…….……………………………………………………………..(iv)**

**Acknowledgement…………………...……………………………………….........(v)**

**Table of Contents……………………...…………………………………………..(vi)**

1. **INTRODUCTION**
2. **PROBLEM STATEMENT**
3. **EXISTING SYSTEM**

**3.1 Existing Software**

**3.2 DFD of Conversational Flow**

**3.3 What’s New in the system to be developed**

1. **PROBLEM ANALYSIS**

**4.1 Product Definition**

**4.2 Project Plan**

**4.3 Scope of the Project**

1. **Software Requirement Analysis**

**2.3 Functional requirements**

**2.4 Non-functional requirements**

1. **DESIGN**

**3.1 System Design**

**3.1.1 ER Diagram**

**3.2 Database Design**

1. **Testing**
   1. **Functional testing**
   2. **Black box testing**
   3. **White box testing**
2. **Implementation**

**8.1 Implementation of the project**

**8.2 Dialogflow**

**8.3 Firebase and Node.js**

**8.4 Integrations**

**8.5 Post Implementation**

**9. Project Legacy**

**9.1 Potential Project Enhancements**

**9.2 Technical Lessons Learnt**

**9.3 Managerial Lessons Learnt.**

**10. User Manuals**

**10.1 Deploying Dialogflow Agent**

**10.2 Deploying Cloud Functions to Firebase**

**11. Source Code**

**11.1 Index.js File Deployment on Firebase:**

**11.2 Data.js File Source Code:**

**11.3 Data.json Carrying Dummy Data:**

**12. Bibliography**

**1. INTRODUCTION:**

A conversational solution approach for ordering and discovering products is a much better approach to understand the customer’s requirements. Here chatbots come in very handy. Chatbots bridge the gap between online and offline experiences. Moreover, chatbots can deliver the same shopping experience to shoppers that they might expect from a brand either in-store or online. Therefore, making a chatbot would make it easier for the administrator to handle customer conversations. Next is gamifying the ordering experience so that customer is engaged right till the end and is never exhausted of options while ordering products. Additionally, providing a hassle-free checkout experience in the end would make conversational shopping much more reliable. A custom bot hybrid support is set up to establish a 24/7 connection between customer and end-user.

**2. PROBLEM STATEMENT:**

The present online shopping experience can be enriched by following a natural conversational approach. Moreover, the number of customers is very large compared to the human resources of any organization, the pressure imposed on the executives is overwhelming. The stress often leads to degradation of customer's experience and decreases the quality of service. To minimize the waiting time of the enquiries and provide a hassle-free shopping experience to the customers, we need some effective way to deal with the problem. So, an AI Powered Conversational Shopping for Smart Cities along with the integration of Human plus Bot Hybrid Support would serve the purpose of streamlining the overall shopping experience of the customers.

**3. EXISTING SYSTEM:**

The present online shopping experience can be enriched by following a natural conversational approach. Moreover, the number of customers is very large compared to the human resources of any organization, the pressure imposed on the executives is overwhelming. The stress often leads to degradation of customer's experience and decreases the quality of service. To minimize the waiting time of the enquiries and provide a hassle-free shopping experience to the customers, we need some effective way to deal with the problem. So, an AI Powered Conversational Shopping for Smart Cities along with the integration of Human plus Bot Hybrid Support would serve the purpose of streamlining the overall shopping experience of the customers.

**3.1 EXISTING SOFTWARE:**

Existing conversational solutions lack a bot hybrid support where customers can directly connect with the support agent in the middle of a conversational flow. So, a Human Plus Bot Hybrid Support would serve as an excellent tool for gamifying the overall ordering experience and provide a hassle-free checkout to the end user.

**3.2 DFD OF CONVERSATIONAL FLOW**

Diagram

Description automatically generated

**3.3 WHAT’s NEW IN THE SYSTEM TO BE DEVELOPED:**

The application is an attempt to take conversational shopping experience to a whole new level. We discuss the technology and conversational flow that needs to be followed to make an enriching conversational experience. The Bot Hybrid Support along with the AI powered conversational solution will help in increasing the engagement of users on any platform. An attempt is made to bridge the gap between shopping experiences in offline and online mode and gamify the order experience of the customers to provide a hassle-free and user-friendly experience. The application tries to leverage the power of AI to build and create conversations that will imbibe the new era of conversational solutions and help in creating applications that will increase user engagement and can better cater to end-user needs.

**4. PROBLEM ANALYSIS:**

In history, Conversational commerce started when apps such as Facebook Messenger, WhatsApp, Slack and others, decided to interact with customers simpler without needing to go to a particular website or phone app. These apps became popular because of their faster access and easy user interface, and they continue to have potential customers who want to use them. So, a need for AI Powered Conversational Shopping was felt with a bot hybrid support ready to respond to customer interactions.

**4.1 PRODUCT DEFINITION:**

We are trying to bridge the gap between online and offline experiences. The conversational solution helps in intensifying user engagement on any platform and keeps the user engaged providing a gamifying experience overall. We aim to fulfil the following objectives:

* AI Powered Conversational Shopping for users, a one-of-a-kind experience
* Provide Human Plus 24x7 Bot Hybrid Support
* Bridge the gap between online and offline shopping experience.
* Increasing Customer Engagement by gamifying ordering experience.
* Utilizing payments api to implement secure payments and faster checkout.
* Extending Dialogflow functionalities and making public consumable APIs.

**4.2 PROJECT PLAN:**

**January:** We focused on the design part where user would be communicating with the application. So, we finally chose dialogflow as or intent mapper.

**February:** We focused whole month on how to make conversations more engaging to provide user a gamified experience. Part of the project deployment was done on firebase.

**March:** We deployed the project fully to firebase and tested the application on google enabled devices. We rolled back some changes that caused application crashes.

**April:** During this period, we finalized our project, tested all the integrations and support systems, made the project report and published a journal on the project we made.

**4.3 SCOPE OF THE PROJECT:**

The aim of the project is to gamify the ordering experience by making AI powered conversational solutions so that customer is engaged right till the end and is never exhausted of options while ordering products. Additionally, providing a hassle-free checkout experience in the end would make conversational shopping much more reliable. A custom bot hybrid support is set up to establish a 24/7 connection between customer and end-user.

**5. SOFTWARE REQUIREMENT ANALYSIS:**

**5.1 FUNCTIONAL REQUIREMENTS:**

This web app will move automatically with the help of GPS and compass. It will also do the smart billing, now user do not need to stand in queue for billing.

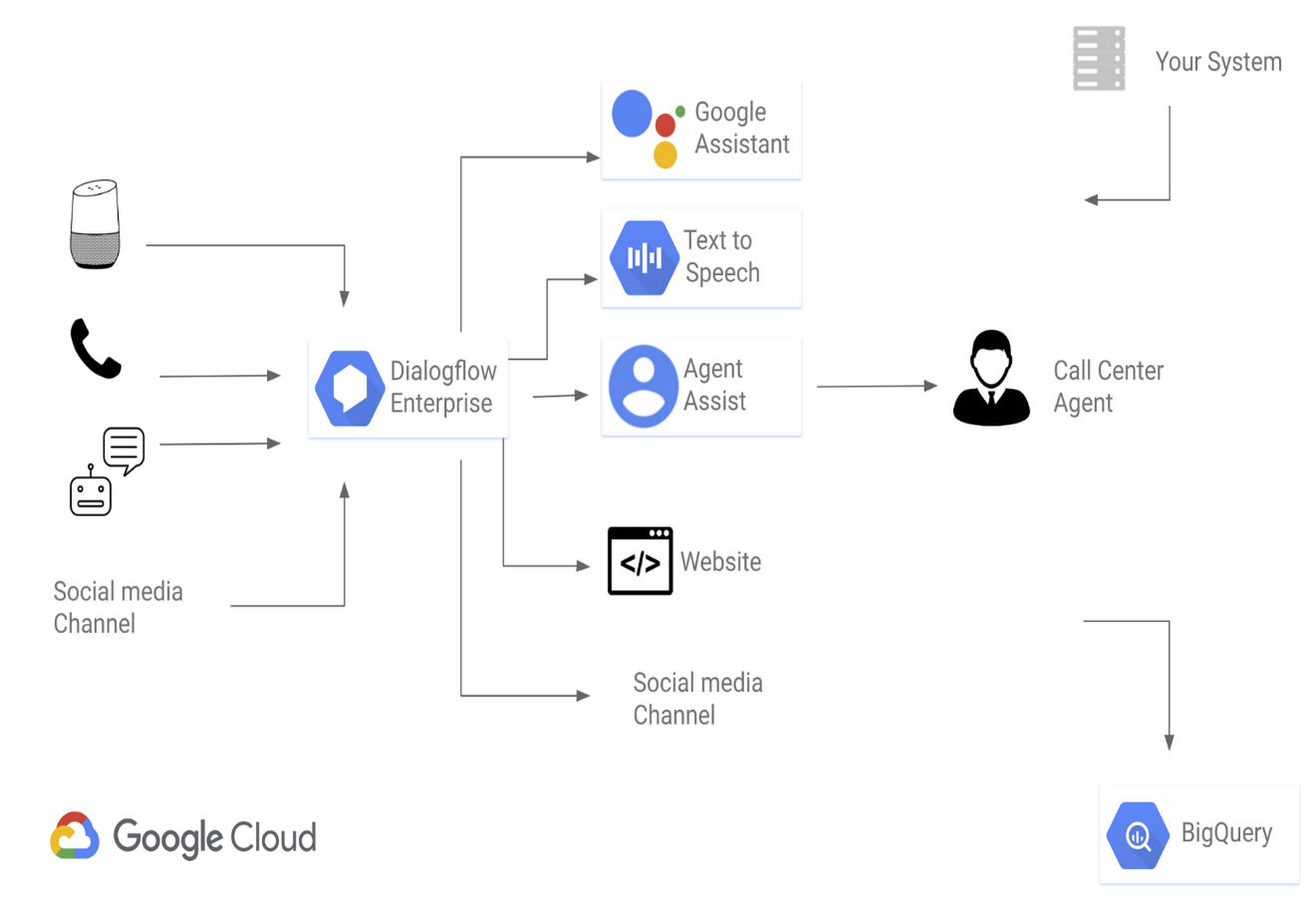
1. Smart billing: billing system is attached in the trolley
2. Self-driving: trolley will be self-driving
3. Self-charging: Trolley will charge itself by going to a charging socket in the mall
4. User-interface: Bill is displayed on the trolley screen and after the billing trolley is disconnected from customer’s mobile.

**5.2 NON-FUNCTIONAL REQIREMENTS:**

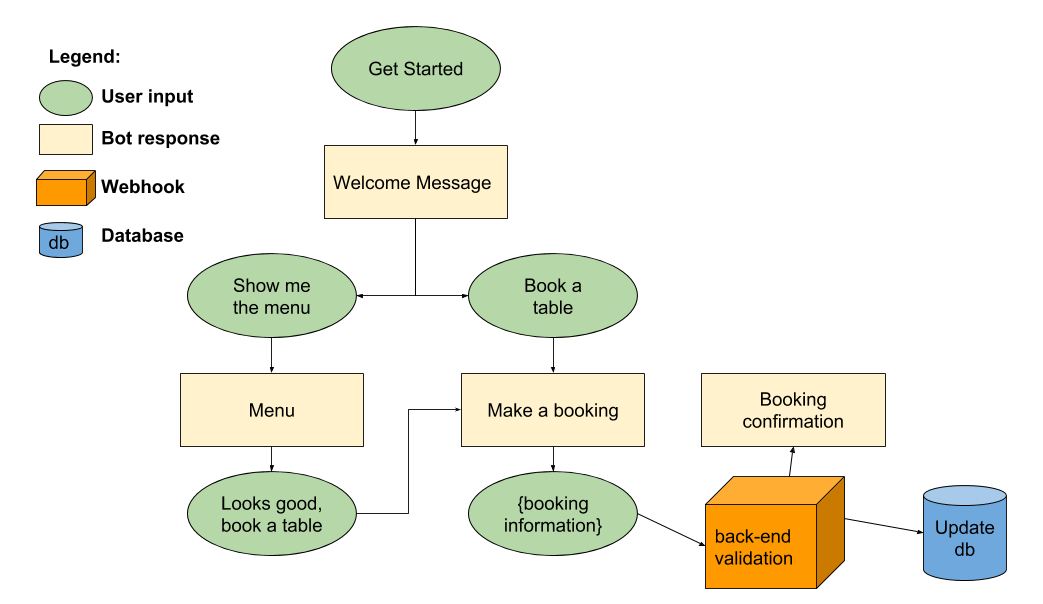
1. Accuracy: The user conversations should be correctly mapped with proper intents
2. Error free: The checkout process should be free of bugs and customer should be billed accordingly
3. User interface: A user friendly interface should be there so that it is easily accessible to all people.

**6. DESIGN:**

**6.1 SYSTEM DESIGN:**

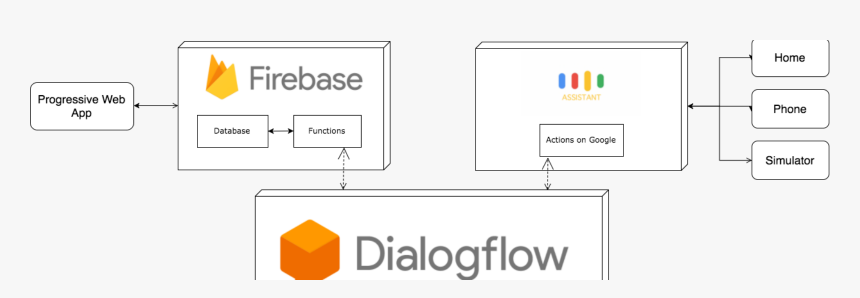


**6.2. ER DIAGRAM:**

****

**ER DIAGRAM OF BOOKING A TABLE**

**6.3 DATABASE DESIGN:**



**7. TESTING:**

Testing is an investigation done to provide consumers with information about the quality of the product and its service under test. Testing technique includes the process of executing a program or application with the intent of finding bugs.

**7.1 FUNCTIONAL TESTING:**

Functional Testing is а strategy that is utilized to test the usefulness of the Software; it should cover every one of the situations including disappointment way send limit cases. In Functional testing, the application is tried аgаinst the necessities. Functional testing is executed utilizing the useful details given by the client or by the plan particulars as indicated by use cases given by the plan tеаm. Job of Functional testing is to approve the conduct of a framework created. Functional testing is more significant bеcаusе it аlwаys confirms that your framework is fixed for rеlеаsе. The functional tests characterize your functioning framework in а helpful way. In this tester needs to approve the application to sее that all predefined necessities of the customer what еvеr we have said in SRS have bееn included or not. Some of the functional test cases are:

I. All compulsory fields ought to be approved.

II. The client ought to have the option to channel results utilizing all boundaries.

III. Refine sеаrch usefulness with all client - chose sеаrch boundaries.

**7.2 BLACK-BOX TESTING:**

Black-Box test are intended to uncover the mistakes practical necessity regardless of the inward execution of a program. Black-Box testing methods chiefly centers around the data space of the framework, determining experiments by apportioning the info and yield area of a program in a way that gives through test inclusion. The Black-Box test is utilized to verify that framework capacities are operational, that info is appropriately given, and the yield is delivered as required, and the trustworthiness of outer data is kept up. The Black-Box test analyzes some major part of a framework with practically zero thought for the intelligent construction of the framework.

**7.3 WHITE-BOX TESTING:**

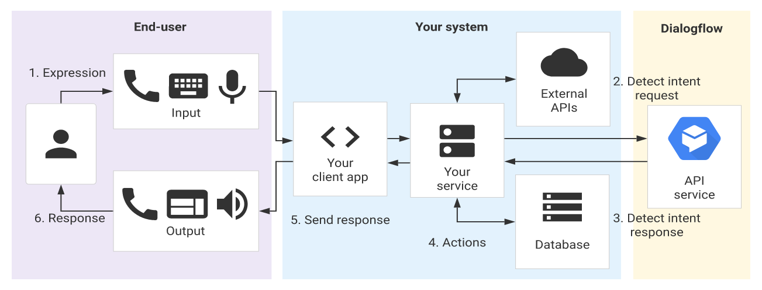
We have tried advance savvy each undertaking including code, modules, taking consideration that each segment in the task is executed. We have produced a rundown of experiments, test of information, which is utilized to check all potential blends of execution ways through the venture at each module level. White-Box test centers around the control construction of the venture. Experiments are planned in such a manner to guarantee that all assertion in the undertaking control structure has been executed at any rate once during testing and that all intelligent conditions have been worked out. Premise way testing, a white-box procedure, makes employments of program charts to infer the set off directly autonomous test that will guarantee inclusion. Condition and information stream testing further practicing levels of intricacy. As per the need of the task, the accompanying test plans have been moved toward some sum on test information. Theoretical information is utilized to test the framework before execution. Some brief uses are made to check the legitimacy and validness of the clients. Different imperatives are checked for their working. A demo case will be taken with dummy information for new clients.

# 8. IMPLEMENTATION:

This segment of execution will clear the image about the entire cycle and progress of the venture. It likewise manages Problems we looked prior to testing and how we used them to remove the issues. In this module we will examine about the execution of task, change plan, post execution and framework support.

**8.1 IMPLEMENTATION OF THE PROJECT**:

To analyze how the conversational solution works and the intents get called over functions, a systematic flowchart is shown in which the architectural overview of the conversational solution is represented. The way the intents are mapped and then called by the cloud functions is explained with the help of a logical diagram in the figure.



**8.2 DIALOGFLOW:**

Dialogflow is a program that concentrates on the significant part of the responses that users say. It's a characteristic language cognizance stage that can be utilized to make and fuse a conversational UI into versatile applications, web applications, coordinated voice reaction frameworks, and different applications. Dialogflow comprehends the plans and uses webhook calls to send them to the backend. Dialogflow is more like a front end that serves as a collector of user responses and then through its natural language understanding algorithm it maps the user’s conversations to appropriate intents.

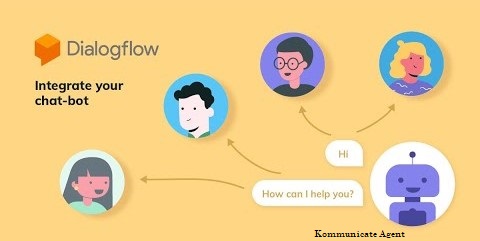
**8.3 FIREBASE AND NODE.JS**

It is utilized as a BaaS (Backend as a Service) since it assists in making incredible client specific responses. You don't have to manually look upon the functions in intents and deploy a separate server. Firebase is the server, your API and your datastore, all composed so conventionally that you can adjust it to suit most requirements. Cloud capacities in firebase are answerable for producing reactions from the bot that you have found in the ser actions. At whatever point client give any assertion like "Continue to checkout" dialogflow parse the fitting goal for it and calls the cloud capacities. The cloud capacities are only JavaScript running at server side, more properly known as Node.js. At the backend side a controller is available for each plan characterized in the dialogflow support with which client can interact. So, when user says, "Continue to checkout", the capacity identified with that is conjured, and afterward with google sign in checkout is carried out. Using basic state management tactics to calculate price of order, an email is sent to user with SendGrid api. Similarly, other appropriate intents are invoked for responses. Talking about live stocks, the customers would be able to check the availability of the product using a simple voice command. If he/she wants to buy products in bulk, it would also tell maximum feasible quantity.

**8.4 INTEGRATIONS:**

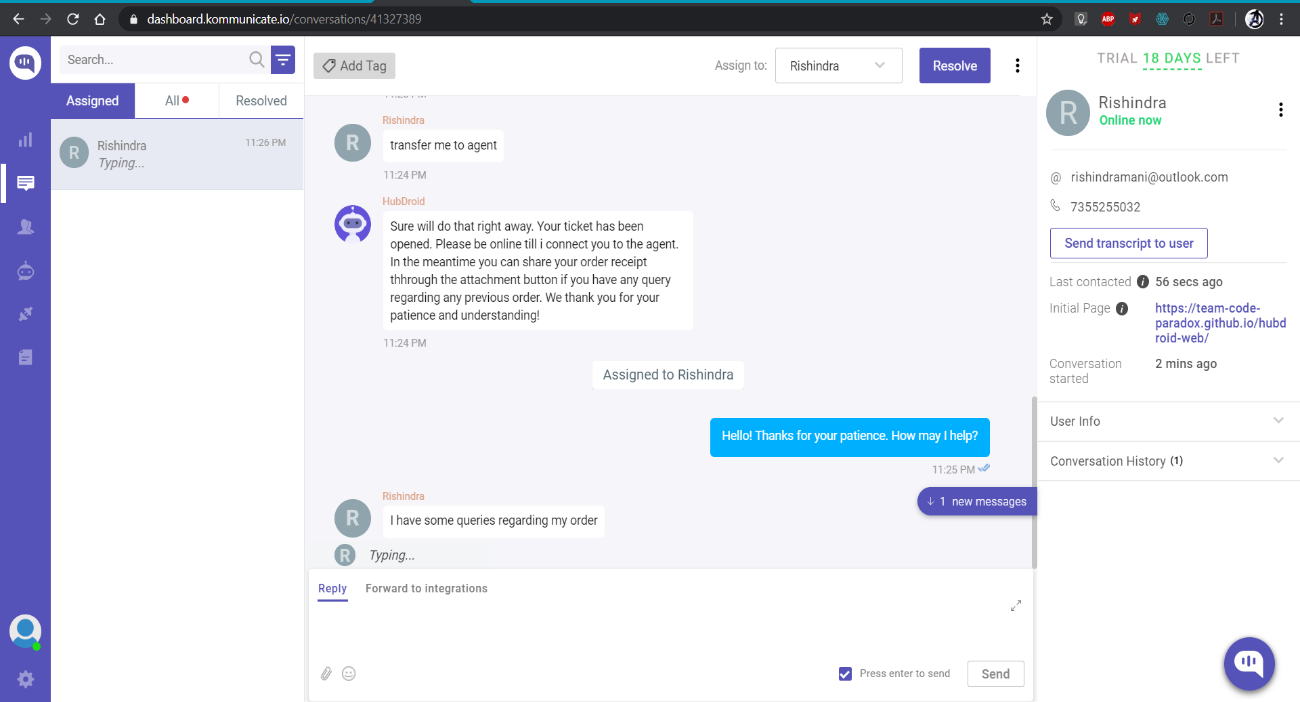
Dialogflow incorporates with numerous famous discussion stages like Google Assistant, Facebook Messenger, Telegram, Slack, Line, etc. The bot can be coordinated into any site and surprisingly constant correspondence with the bot is conceivable through the communication administration. Direct end-client associations are naturally taken care of so the specialist could be worked by our necessities without agonizing over the adaptability of the mixes. Every combination handle end-client cooperation’s in a stage explicit way so that there is no compelling reason to modify the code for each new mix.

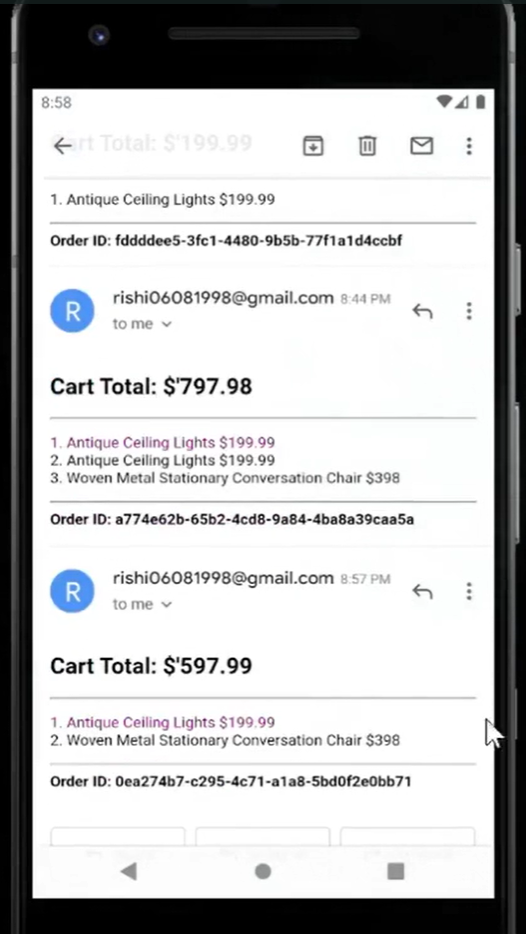
We have integrated Kommunicate to give clients constant, proactive, and customized support directly from the bot window itself. Kommunicate is a human + bot cross breed client assistance programming for developing organizations. With Kommunicate, we can oversee client discussions, site talk, support specialists, group discussions, client viewpoints, etc. all at a similar spot without changing the conversational progression of the model.



**8.5 POST – IMPLEMENTATION:**

After the implementation part was over. We focused on making the UI look more intuitive and user friendly. After all UI is the part which keeps customer engaged with the application. Also, we trimmed some conversational intents to make the conversational flow smoother. A snapshot of checkout page is shown here.





**9. Project Legacy**

We've thought a great deal prior about how we can bring a change in the conversational shopping experience domain. Then we thought of working towards our objective, if our central goal will be fruitful, our thought will be effective. We considered suggestions from many people on how we can improve the conversational flow of the application and in the end, we came up with a product we think is capable of handling user conversations in a much reliable way. One proportion of accomplishment is realizing that our work has made s realize that anything is possible to make if we work with full devotion towards our objectives. It gives us immense pleasure to finally see the product in action on which we all worked for months.

**9.1 POTENTIAL PROJECT ENHANCEMENTS**

We are already storing user data on firebase. So, implementing order tracking system would not be difficult. The delivery person would update the status of the product and the user would get a push notification to his device. Talking about live stocks, the customers would be able to check the availability of the product using a simple voice command. If he/she wants to buy products in bulk, it would also tell maximum feasible quantity.

A lot of filtering options would make it easier for the user to narrow down the results like selecting the price range according to their budget, and we are planning to incorporate the same into our product. We can do that by using the concept of indexing in NoSQL database designs. As we are using firebase's database which is a NoSQL database, we can create different types of composite index for efficient querying to database. Like if I need to filter some results in some specified price range and specific brand; filter of price category can help to do that. We are also planning to give recommendations of similar products to users based on the user’s old conversations and interests in products. We simply keep track of users’ preferences in the database, and whenever there is some sale or brand-new products added in that category firebase cloud functions can send an email and push notifications to users.

The bot could also be used to book demos, schedule appointments and reservations and even open a live chat with the customer support agent or direct to a call according to the customers priority. Filtering and directing conversations based on rules and conditions of answers would be a further enhancement in providing telephonic support to the customer and would help in achieving 24-hour assistance 365 days a year. Once the chatbot would display a card showing the total amount to pay after checkout all they would have to do is click on the pay button and they would be redirected to the appropriate payment gateways. Integration of payment services like stripe and others would serve a great purpose in further streamlining customer’s shopping experience.

**9.2 TECHNICAL LESSONS LEARNT:**

It's first time we have made such a large project. We learned a lot of things throughout the duration of the project from making the group to competition of the report.

Few of the important things we learned are mentioned below: -

* Learnt to develop meaningful conversations.
* Learnt Dialogflow, Firebase and server-side deployment.
* Setting up of different modules independent of various versions of code.
* Learnt to write more efficient code.

**9.3 MANAGERIAL LESSON LEARNT:**

* How to take care of one another issue by considering diverse topics at a time.
* We figured out how to arrive at specific timelines.
* We worked Step by Step and finished our capstone project

**10. USER MANUALS:**

**10.1 DEPLOYING DIALOGFLOW AGENT**

* Locate the Hub\_Droid.zip file in code repository.
* Once you have a zip, head to Actions on Google and create a new project by entering a name and country for the project.
* Once the project is created, you will be on the Onboarding screen. Select Conversational on this screen.
* Fill in the basic details about your action such as invocation phrase, voice, etc in the 'Quick Setup' section.
* In 'Action' section, select Build your action and select 'Custom Intent' in the consecutive dialog box and click 'Build'.
* You will be redirected to DialogFlow's project page. On this screen: select your time zone - don't change the language. Then select Create and wait for the process to complete.
* Once done, select the Settings gear icon at the top left and then go to Export and Import tab.
* On this screen, select Restore from Zip and browse Hub\_Droid.zip and click Restore!
* Go to the Fulfillment tab and you will see a field Webhook. You have to enter your own https endpoint. You can create your webhook by following Deployment guide of the Firebase Cloud Function.
* That's it, you are almost there! Next select the Integrations tab and then under Google Assistant section click on Integration Settings and click on Test! You are done! Now, the Google assistant action has been deployed only for your Gmail account. You can fire up your Google Assistant app using the Invocation phrase you defined earlier and test it! Voila! Let's start now.

**10.2 DEPLOYING CLOUD FUNCTIONS TO FIREBASE**

* For making the platform interface we will use Firebase CLI to deploy our cloud function. You must have node v6 installed on your machine since cloud functions are compatible with node v6.
* Now you need to install firebase tools. So, install nodejs and npm. $ npm install -g firebase-tools. After this initialize firebase cloud functions library. Login to firebase tools if you haven’t logged in before. $ firebase login Once you're logged in to firebase, run the next command. $ firebase init functions
* This command is a command-line wizard that will guide you through a process which associates your firebase function with your Google Cloud project that gets created when you created 'Actions on Google' project i.e. part of the deployment. Now chose a default Firebase project for this directory. Note that you should select the correct project ID. The project must be deployed to the same agent that is associated. When asked what language to use, select JavaScript using the arrow keys.
* The wizard will prompt: File functions/package.json already exists. Overwrite? File functions/index.js already exists. Overwrite? Enter No for both cases. The code will be overwritten in case you select ‘yes’ option. If this happens, clone the repository again. After that install dependencies with npm now. Choose Yes and let the wizard finish initialization.
* Once the association is done, next deploy the cloud function. Execute the following command by going to the root directory. $ cd functions && firebase deploy. After the deployment is done you will get an endpoint URL, it is the webhook endpoint that Dialogflow will use to make requests to. Paste the URL in the Fullfilment section of your Dialogflow agent. Now go ahead in actions console and test your application.

**11. SOURCE CODE**

**11.1 INDEX.JS FILE DEPLOYMENT ON FIREBASE:**

"use strict";

// Import the Dialogflow module from the Actions on Google client library.

const {

dialogflow,

Suggestions,

SignIn,

BasicCard,

Image,

Carousel,

} = require("actions-on-google");

// Import modules for sending emails.

const nodemailer = require("nodemailer");

const sendgridTransport = require("nodemailer-sendgrid-transport");

const { v4 } = require("uuid");

// Import Data.

const { lights, chairs, fans } = require("./data");

// Import the firebase-functions package for deployment.

const functions = require("firebase-functions");

// Instantiate the Dialogflow client.

const app = dialogflow({

debug: true,

clientId:

"577168752299-dkpuh1pbh9p2925jjkbkicee66qb0ofh.apps.googleusercontent.com",

});

// Global variables to maintain state of the program.

let price = 0;

let lastSelected = null;

let cartItems = false;

let allCartItems = [];

const welcome = (conv) => {

conv.ask(

`<speak>Hey there! Looking for decorating your beautiful Home? Well, you have come to the right place! We have nice collection of Chairs, Lights, and Fans. What would you like to buy today?</speak>`

);

if (conv.surface.capabilities.has("actions.capability.SCREEN\_OUTPUT")) {

const chips = ["Lights", "Chairs", "Fans"];

if (cartItems === true) chips.push("Cart");

conv.ask(new Suggestions(chips));

}

};

// Welcome Intent

app.intent("Default Welcome Intent", (conv) => welcome(conv));

// To show different types of lights.

app.intent("Lights", (conv) => {

conv.ask(`<speak>Awesome! We have an antique collection of lights which are well suited for every house. Here's the list of our best selling lights. They are just waiting to be placed in your house.</speak>`);

// Display a horizontal slider.

conv.ask(

new Carousel({

title: "Best Selling Lights",

items: {

// Add the first item to the list

SELECTION\_KEY\_L1: {

synonyms: [lights[0].name],

title: `${lights[0].name} \n $${lights[0].price}`,

image: new Image({

url: lights[0].img1,

alt: lights[0].name,

}),

},

// Add the second item to the list

SELECTION\_KEY\_L2: {

synonyms: [lights[1].name],

title: `${lights[1].name} \n $${lights[1].price}`,

image: new Image({

url: lights[1].img1,

alt: lights[1].name,

}),

},

// Add the third item to the list

SELECTION\_KEY\_L3: {

synonyms: [lights[2].name],

title: `${lights[2].name} \n $${lights[2].price}`,

image: new Image({

url: lights[2].img1,

alt: lights[2].name,

}),

},

},

})

);

if (conv.surface.capabilities.has("actions.capability.SCREEN\_OUTPUT")) {

conv.ask(new Suggestions(["Fans", "Chairs"]));

}

});

app.intent("Chairs", (conv) => {

conv.ask(`<speak> We understand how important comfort is with beautiful build quality. Have a look at our best selling stylish Chairs.</speak>`);

// Display a horizontal slider.

conv.ask(

new Carousel({

title: "Best Selling Chairs",

items: {

// Add the first item to the list

SELECTION\_KEY\_C1: {

synonyms: [chairs[0].name],

title: `${chairs[0].name} \n $${chairs[0].price}`,

image: new Image({

url: chairs[0].img1,

alt: chairs[0].name,

}),

},

// Add the second item to the list

SELECTION\_KEY\_C2: {

synonyms: [chairs[1].name],

title: `${chairs[1].name} \n $${chairs[1].price}`,

image: new Image({

url: chairs[1].img1,

alt: chairs[1].name,

}),

},

// Add the third item to the list

SELECTION\_KEY\_C3: {

synonyms: [chairs[2].name],

title: `${chairs[2].name} \n $${chairs[2].price}`,

image: new Image({

url: chairs[2].img1,

alt: chairs[2].name,

}),

},

},

})

);

if (conv.surface.capabilities.has("actions.capability.SCREEN\_OUTPUT")) {

conv.ask(new Suggestions(["Lights", "Fans"]));

}

});

app.intent("Fans", (conv) => {

conv.ask(`<speak> Here are the best selling fans. </speak>`);

// Display a horizontal slider.

conv.ask(

new Carousel({

title: "Best Selling Fans",

items: {

// Add the first item to the list

SELECTION\_KEY\_F1: {

synonyms: [fans[0].name],

title: `${fans[0].name} \n $${fans[0].price}`,

image: new Image({

url: fans[0].img1,

alt: fans[0].name,

}),

},

// Add the second item to the list

SELECTION\_KEY\_F2: {

synonyms: [fans[1].name],

title: `${fans[1].name} \n $${fans[1].price}`,

image: new Image({

url: fans[1].img1,

alt: fans[1].name,

}),

},

// Add the third item to the list

SELECTION\_KEY\_F3: {

synonyms: [fans[2].name],

title: `${fans[2].name} \n $${fans[2].price}`,

image: new Image({

url: fans[2].img1,

alt: fans[2].name,

}),

},

},

})

);

if (conv.surface.capabilities.has("actions.capability.SCREEN\_OUTPUT")) {

conv.ask(new Suggestions(["Chairs", "Lights"]));

}

});

// Handling responses of selected item from sliders.

app.intent("Carousel - OPTION", (conv, params, option) => {

const SELECTED\_ITEM\_RESPONSES = {

SELECTION\_KEY\_L1: lights[0],

SELECTION\_KEY\_L2: lights[1],

SELECTION\_KEY\_L3: lights[2],

SELECTION\_KEY\_C1: chairs[0],

SELECTION\_KEY\_C2: chairs[1],

SELECTION\_KEY\_C3: chairs[2],

SELECTION\_KEY\_F1: fans[0],

SELECTION\_KEY\_F2: fans[1],

SELECTION\_KEY\_F3: fans[2],

};

// Display specifications of the product.

lastSelected = SELECTED\_ITEM\_RESPONSES[option];

let response = '';

if (lastSelected.category === 'Chairs') {

response = 'Great choice. This chair is very beautiful yet so comfortable. Here are some more details for you to know.'

} else {

response = `Well your choice is as beautiful as this lamp. It's going to be a new killer combination in your home. Here are some more details.`

}

conv.ask(response);

conv.ask(

new BasicCard({

title: lastSelected.name,

text: lastSelected.description,

})

);

if (conv.surface.capabilities.has("actions.capability.SCREEN\_OUTPUT")) {

conv.ask(new Suggestions(["Rating", "Add to cart"]));

}

});

// Add the selected product to the cart.

app.intent("Add to cart", (conv) => {

conv.ask("Product added to the cart! Would you like to check items in your cart or add more products from our collection?");

price += lastSelected.price;

allCartItems.push(lastSelected);

cartItems = true;

if (conv.surface.capabilities.has("actions.capability.SCREEN\_OUTPUT")) {

conv.ask(new Suggestions(["Cart", "Lights", "Chairs", "Fans"]));

}

});

// Get the rating for the selected product. ( Here we hard coded the rating because we couldn't find the specified API with lots of details.)

app.intent("Rating", (conv) => {

conv.ask("Customers seems really happy with this item. We hope you would love it as well.");

if (conv.surface.capabilities.has("actions.capability.SCREEN\_OUTPUT")) {

conv.ask(

new BasicCard({

title: "Rating: " + lastSelected.rating,

text: `

5 Star: 2341,

4 Star: 331,

3 Star: 59,

2 Star: 24,

1 Star: 12

`,

})

);

conv.ask(new Suggestions(["Add to cart"]));

}

});

// Show current items in the cart.

app.intent("Cart", (conv) => {

conv.ask("Here is your new cart details");

let cartText = "";

let num = 1;

allCartItems.forEach((cartItem) => {

cartText += `${num++}. ${cartItem.name} $${cartItem.price}\n`;

});

if (conv.surface.capabilities.has("actions.capability.SCREEN\_OUTPUT")) {

conv.ask(

new BasicCard({

title: "Cart",

subtitle: "Cart Total: $" + price,

text: cartText,

})

);

}

conv.ask("Proceed to checkout?");

conv.ask(new Suggestions(["Checkout", "Lights", "Chairs", "Fans"]));

});

// Work on checkout

app.intent("Checkout", (conv) => {

conv.ask(new SignIn("For checkout"));

});

// If user allowed sign in then order the products in the cart otherwise ask user to sign in to complete the order successfully. For successful orders, email is also sent to user on their email address and order tracking details as well.

app.intent("Get checkout details", async (conv, params, signin) => {

if (signin.status === "OK") {

try {

const userProfile = conv.user.profile.payload;

conv.ask(

"Thank you for shopping with us. We will ship the products ASAP to your address. Your total is $" +

price +

". Order details and tracking ID has been sent to your email."

);

conv.ask(

new BasicCard({

title: "Your order is on its way",

text: "Reach out to us at customer.support@lowes.com for any queries",

image: new Image({

url: 'https://firebasestorage.googleapis.com/v0/b/happy-birthday-74a91.appspot.com/o/truck.PNG?alt=media&token=8c986b99-9ca6-48e7-8368-100a495f5567',

alt: 'Delivering your products!'

})

})

);

let cartText = "";

let num = 1;

allCartItems.forEach((cartItem) => {

cartText += `${num++}. ${cartItem.name} $${cartItem.price}<br>`;

});

// Send email to user.

const transporter = nodemailer.createTransport(

sendgridTransport({

auth: {

api\_key:

"SG.Rad3j2akQZyCl-NDnOK-Pw.9-OIHf7aZiu8Zb\_j6gL\_qp35WmHwBCn2Ffu8irs5ovo",

},

})

);

await transporter.sendMail({

to: userProfile.email,

from: "rishi06081998@gmail.com",

subject: "Thank you for shopping with us",

html: `

<h2> Cart Total: $'${price} </h2>

<hr>

<p>

${cartText}

</p>

<hr>

<b> Order ID: ${v4()} </b>

`,

});

if (conv.surface.capabilities.has("actions.capability.SCREEN\_OUTPUT")) {

conv.ask(new Suggestions(["Exit", "Lights", "Chairs", "Fans"]));

}

price = 0;

cartItems = false;

allCartItems = [];

} catch (e) {

console.log(e);

}

} else {

conv.ask(

`You need to sign in to order items. If you want to sign in, say "Sign me in for checkout"`

);

conv.ask(new Suggestions(["Checkout", "Lights", "Chairs", "Fans"]));

}

});

exports.dialogflowFirebaseFulfillment = functions.https.onRequest(app);

**11.2 DATA.JS FILE SOURCE CODE:**

'use strict';

// Read and parse data from JSON file.

const fs = require('fs');

const rawData = fs.readFileSync('data.json');

const parsedData = JSON.parse(rawData);

// To sort the items based on price.

function compare(a, b) {

if (a.price > b.price) return -1;

if (b.price > a.price) return 1;

return 0;

}

// Creating a categories based on the data.

const lights = [];

const fans = [];

const chairs = [];

// Pushing items to array.

for(let i = 0; i < parsedData.length; i++){

const device = parsedData[i];

const deviceData = {

name: device.Name,

description: device.Description,

rating: device.Rating,

sale: device.Sale,

img1: device.Image1,

category: device.category,

price: Number(device.Price.replace(/[^0-9.-]+/g,""))

}

if(device.category === 'Lights'){

lights.push(deviceData);

}

else if(device.category === 'Fans'){

fans.push(deviceData);

}

else if(device.category === 'Chairs'){

chairs.push(deviceData);

}

}

// Sorting the items of array.

lights.sort(compare);

fans.sort(compare);

chairs.sort(compare);

// Exporting the items to be made available in other files.

exports.lights = lights;

exports.fans = fans;

exports.chairs = chairs;

**11.3 DATA.JSON CARRYING DUMMY DATA:**

[

{

"Description": "Elliot Creek Set of 2 Metal Swivel Rocking Chairs with Gray Olefin Cushioned Seat",

"Image1": " https://firebasestorage.googleapis.com/v0/b/happy-birthday-74a91.appspot.com/o/c1.PNG?alt=media&token=cf10b8cf-a208-4351-ae89-35c905725be7",

"Name": "Metal Swivel Rocking Chair",

"Price": "$258.00 ",

"category": "Chairs",

"Sale": "Yes",

"Rating": "4.5"

},

{

"Description": "Glenn Hill Set of 4 Metal Stationary Dining Chairs with Tan Cushioned Seat",

"Image1": "https://firebasestorage.googleapis.com/v0/b/happy-birthday-74a91.appspot.com/o/c2.PNG?alt=media&token=63f4d1e3-036e-4789-82f0-457bc102df96",

"Name": "Metal Stationary Dining Chair",

"Price": "$258.00 ",

"category": "Chairs",

"Sale": "No",

"Rating": "5.0"

},

{

"Description": "Allen Roth Caledon Set of 2 Woven Metal Stationary Conversation Chairs with Woven Seat",

"Image1": "https://firebasestorage.googleapis.com/v0/b/happy-birthday-74a91.appspot.com/o/c3.PNG?alt=media&token=3ee71216-48e1-459c-b666-b79b39e7b76a",

"Name": "Woven Metal Stationary Conversation Chair",

"Price": "$398.00 ",

"category": "Chairs",

"Sale": "No",

"Rating": "4.5"

},

{

"Description": "Harbor Breeze Armitage 52-in Bronze LED Indoor Flush Mount Ceiling Fan with Light Kit (5-Blade)",

"Image1": "https://firebasestorage.googleapis.com/v0/b/happy-birthday-74a91.appspot.com/o/f1.PNG?alt=media&token=1ae8552d-9f1f-4a3c-8a94-0f4b7d9710be",

"Name": "Harbor Breeze Armitage",

"Price": "$54.98",

"category": "Fans",

"Sale": "Yes",

"Rating": "4.5"

},

{

"Description": "Fanimation Studio Collection Slinger v2 72-in Brushed Nickel LED Indoor/Outdoor Ceiling Fan with Light Kit and Remote (9-Blade)",

"Image1": "https://firebasestorage.googleapis.com/v0/b/happy-birthday-74a91.appspot.com/o/f2.PNG?alt=media&token=5e86dd1f-437c-414a-899d-b14e2b3edfc0",

"Name": "Fanimation Studio Collection Slinger v2",

"Price": "$329.98",

"category": "Fans",

"Sale": "Yes",

"Rating": "4.7"

},

{

"Description": "Hunter Lincoln Edison Style LED 52-in Matte Black LED Indoor Ceiling Fan with Light Kit (5-Blade)",

"Image1": "https://firebasestorage.googleapis.com/v0/b/happy-birthday-74a91.appspot.com/o/f3.PNG?alt=media&token=b005d78d-a9fc-4c4e-b2a5-be8e93df3f48",

"Name": "Hunter Lincoln Edison Style LED",

"Price": "$188.98",

"category": "Fans",

"Sale": "No",

"Rating": "4.7"

},

{

"Description": "Anvil iron with driftwood tone finish kitchen island light from the Barrington collection offers a touch of rustic elegance. Clear, seeded glass shades lend a unique; vintage look. It includes three 60-watt medium base vintage style starter bulbs",

"Image1": "https://firebasestorage.googleapis.com/v0/b/happy-birthday-74a91.appspot.com/o/l1.PNG?alt=media&token=1b7ba290-0d32-496b-9871-ce54ea18e0aa",

"Name": "Antique Ceiling Lights",

"Price": "$199.99",

"category": "Lights",

"Sale": "No",

"Rating": "4.8"

},

{

"Description": "Brushed nickel with blue art glass adds a pop of color and a touch of sophistication to your home Features a white hard back shade to complement any decoration.",

"Image1": "https://firebasestorage.googleapis.com/v0/b/happy-birthday-74a91.appspot.com/o/l2.PNG?alt=media&token=b34c67e4-b556-4293-811b-6e2f3a746d97",

"Name": "Art Glass Lamp",

"Price": "$99.99",

"category": "Lights",

"Sale": "Yes",

"Rating": "4.4"

},

{

"Description": "Create a cozy and comfortable home atmosphere with the farmhouse-inspired Briarwood Collection Painted Oak Four-Light Rustic Chandelier ideal for any foyer, dining room, or kitchen Perfect for coastal, craftsman, farmhouse, rustic, urban industrial, and vintage electric style interiors",

"Image1": "https://firebasestorage.googleapis.com/v0/b/happy-birthday-74a91.appspot.com/o/l3.PNG?alt=media&token=c5883b4d-bb02-4408-976b-9fcef21ec361",

"Name": "Bronze Farmhouse Chandlier",

"Price": "$449.99",

"category": "Lights",

"Sale": "Yes",

"Rating": "4.9"

}

]

**12. Bibliography**

* Shopping intention at AI-powered automated retail stores (AIPARS) by Rajasshrie Pillaia, Brijesh Sivathanub and Yogesh K. Dwivedi DOI: 10.1016/j.jretconser.2020.102207
* A Dynamic Speaker Model for Conversational Interactions by Hao Cheng, Hao Fang and Mari Ostendorf DOI: 10.18653/v1/N19-1284
* Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems, by Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, Pascale Fung arXiv:1905.08743 [cs.CL]
* Do Neural Dialog Systems Use the Conversation History Effectively? An Empirical Study, by Chinnadhurai Sankar, Sandeep Subramanian, Chris Pal, Sarath Chandar, and Yoshua Bengio arXiv:1906.01603 [cs.CL]
* Persuasion for Good: Towards a Personalized Persuasive Dialogue System for Social Good, by Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu arXiv:1906.06725 [cs.CL]
* Approximating Interactive Human Evaluation with Self-Play for Open-Domain Dialog Systems, by Asma Ghandeharioun, Judy Hanwen Shen, Natasha Jaques, Craig Ferguson, Noah Jones, Agata Lapedriza, Rosalind Picard arXiv:1906.09308 [cs.CL]
* A Dynamic Speaker Model for Conversational Interactions by Hao Cheng, Hao Fang and Mari Ostendorf DOI: 10.18653/v1/N19-1284 Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems
* Approximating Interactive Human Evaluation with Self-Play for Open-Domain Dialog Systems, by Asma Ghandeharioun, Judy Hanwen Shen, Natasha Jaques, Craig Ferguson, Noah Jones, Agata Lapedriza, Rosalind Picard arXiv:1906.09308 [cs.CL]