AI BASED DISCOURSE FOR BANKING INDUSTRY

TEAM ID : PNT2022TMID35942

NALAIYA THIRAN PROJECT BASED LEARNING

A PROJECT REPORT SUBMITTED BY

YUVAN SHANKAR J (2019504609)
KAVYAPRIYA V K (2019504537)
SHWETHA M K (2019504587)
KARTHICK V (2019504535)

BACHELOR OF ENGINEERING
IN
ELECTRONICS AND COMMUNICATION

MADRAS INSTITUTE OF TECHNOLOGY, ANNA UNIVERSITY
CHENNAI

1. INTRODUCTION

1.1 Project Overview

Artifical intelligence is implemented in various real life scenarios to support 24/7 availability, digital assistance and reduce human error. It is a multi-disclipinary field whose goal is to automate activities that presently require human intelligence. Banking is a predominant sector to provide financial services to the users, which will definitely have a positive impact on the economy of a country. Essential needs like food, shelter requires exchange of money between people. Therefore, money management has become vital in everybody's life. Money management and other financial services can be provided to the people through banks. In order to provide best solution to people at all times, an intelligent system has to introduced to improve efficiency and customer service. An efficient solution for the requirement of this intelligent system in banking can be met by introducing Chatbots based on Artificial Intelligence. These Chatbots use Artificial Intelligence to mimic human conversation. It is built to serve as a virtual assitant that can faciliate customers to ask customers to ask banking-related queries without visiting the bank or calling customer service centers. Chatbots are now becoming popular in business groups as they can minimise the customer support costs.

1.2 Purpose

The main purpose of the project is to serve a bank customer who needs 24/7 service to clear all his doubts and guide him through all the banking processes. An enhanced and smarter way of interaction with the customers has to be built to ensure efficient delivery of service. in order to overcome the user satisfaction issues associated with banking services, chatbot will provide personal and efficient communication between the user and the bank. As traditional forms of financial exchanges change, technology is heralding for financial institutions from human-centered to computer-centered financial services. These banking chatbots allow financial institutions to talk to millions of customers at once. As people continue to avoid branches in favor of digital banking, they expect more banks to establish virtual assitants. Digital banking and Artificial Intelligence has a broad positive impact for its users. The proposed model guides the customers through creation of bank accounts, net banking, loan quieres and general queries.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEMS:

S. No.	REFERENCE	AUTHOR	EXISTING PROBLEM
1	A Study Of Applications Of Artificial Intelligence In Banking And Finance Sector, IJIRMF, 2020	Dr.Lakshkaushik Dattatraya Puri	1. AI has been used in banking for decades, but it remained unnoticed. 2. ii) AI in banking is continuing to transform the industry to provide a greater level of value to their customers, reduce risks, and increase opportunities as the financial engines of our modern economy.
2	Voice recognition bot for internet banking, IEEE, 2022	Gomathy B, Krishna Kumar S, Mukilan R, Naveen Balaji R	1. The bot can be able to train more datasets if wanted. 2. It can be put on every industry for 24/7 and thus get their doubts cleared anytime anywhere.
3	Bank chat bot– An Intelligent Assistant System Using NLP and Machine Learning, IRJET, 2017	Chaitrali S. Kulkarni, Amruta U. Bhavsar, Savita R. Pingale, Prof. Satish S. Kumbhar	1. This intelligent query handling program can be further made efficient by training the model to self-learn and thereby increasing not just the quality of customer service but also reducing human load, increase in productivity and of course increasing number of satisfied customers.
4	Application of Chatbot for consumer perspective using Artificial Intelligence, IEEE, 2021	Abhishek Savanur, Niranjanamurthy M, Amulya M P, Dayananda P	1. The model should not change based on any new replies. 2. It should not just rephrase what people say, but indeed should be taught to answer things what customers require.
5	Analysing and designing conversational banking service architecture for banking company, IEEE 2020	Emil Robert Kaburuan, Adrianus Kelvin, Jery	1. Usage of firewalls could obstruct some organisational activities. 2. ii) Facial biometrics requires huge storage requirements and privacy issues.
6	Conversation to Automation in Banking Through Chatbot Using	Sasha Fathima Suhel, Vinod Kumar Shukla, Sonali Vyas, Ved Prakash	1. The absence of a intelligent question management program capable of not only responding but of self-learning to improve itself in the next stages, thus not only increasing the quality of user service but also reducing human

	Artificial Machine Intelligence Language, IEEE, 2020	Mishra	loads, increasing productivity and, of course, increasing the number of satisfied users can be included. 2. Intelligent answers created by entering not only the current FAQ list, but also various other outlets such as twitter, servers, and other data sources can be included.

2.2 REFERENCES:

1. Dr.Lakshkaushik Dattatraya Puri ,A Study Of Applications Of Artificial Intelligence In Banking And Finance Sector, IJIRMF, 2020
2. Gomathy B, Krishna Kumar S, Mukilan R, Naveen Balaji R ,Voice recognition bot for internet banking, IEEE, 2022
3. Chaitrali S. Kulkarni, Amruta U. Bhavsar, Savita R. Pingale, Prof. Satish S. Kumbhar Bank chat bot– An Intelligent Assistant System Using NLP and Machine Learning, IRJET, 2017
4. Abhishek Savanur, Niranjanamurthy M, Amulya M P, Dayananda P, Application of Chatbot for consumer perspective using Artificial Intelligence, IEEE, 2021
5. Emil Robert Kaburuan, Adrianus Kelvin, Jery ,Analysing and designing conversational banking service architecture for banking company, IEEE 2020
6. Sasha Fathima Suhel, Vinod Kumar Shukla, Sonali Vyas, Ved Prakash MishraConversation to Automation in Banking Through Chatbot Using Artificial Intelligence Language, IEEE, 2020

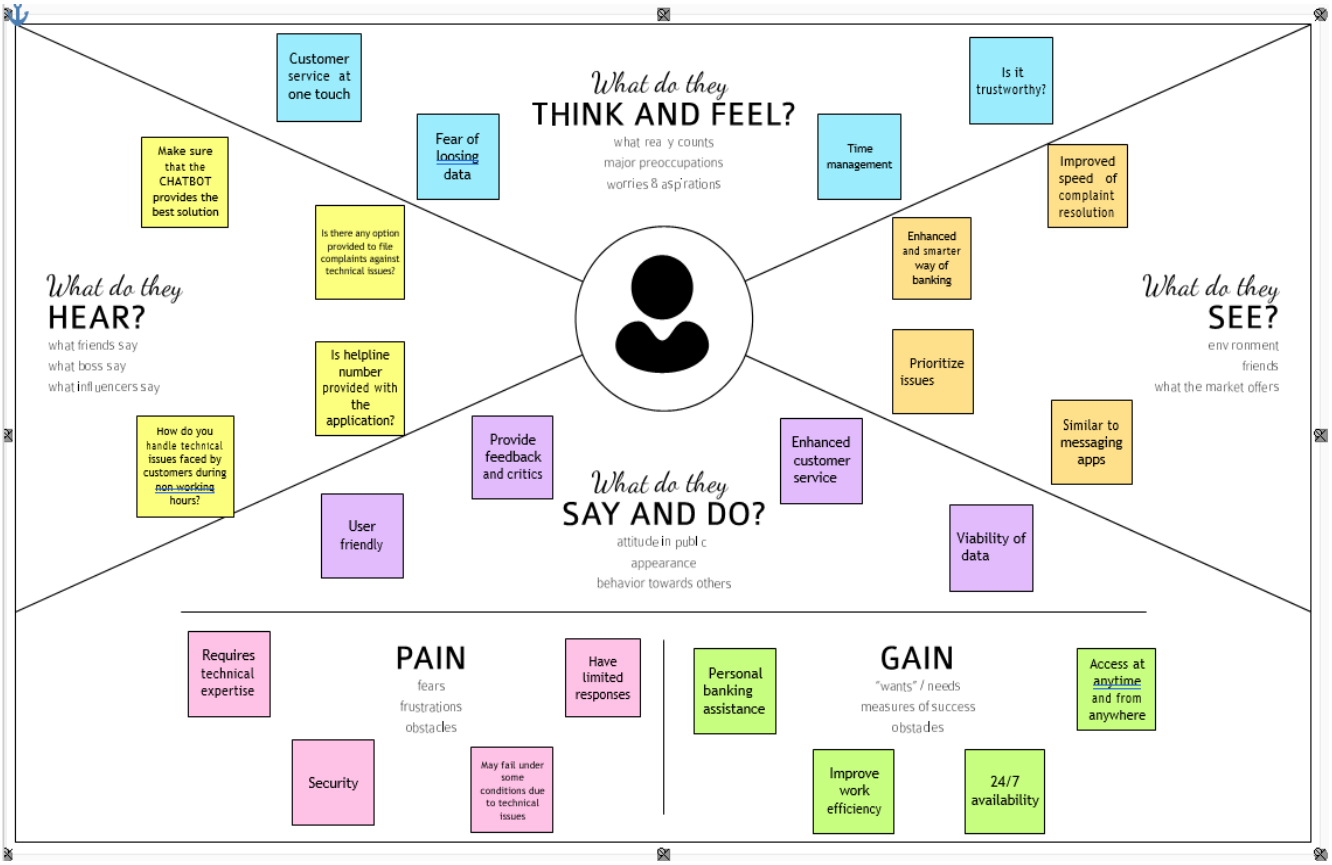
2.3 PROBLEM STATEMENT

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	I am a bank customer	Trying to know my current account balance	But I'm unable to visit a bank	Due to bad weather	Which makes me feel disapppointed
PS-2	I am bank customer	Trying to know about the working hours of the bank and about the loan options available	But unable to visit my home branch	Since they are simple queries	Which makes me feel worried

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

This map is prepared to create a simple and easy to digest visual that captures knowledge about users attitudes and behaviour. This map is useful to understand about the customers of the bank, which could help us to think from user's perceptive. The empathy map is a square divided into four quadrants with the user or client in the middle. Each of the four quadrants comprises a category that helps us delve into the mind of the user. The four empathy map quadrants look at what the user says, thinks, hear and does.



3.2 Ideation & Brainstorming

Brainstorming is part of design thinking and hence used in the ideation phase. Brainstorming is a group activity where each participant shares their ideas as soon as they come to mind. At the conclusion of the session, ideas are categorised and ranked for follow-on action. This expands experience pool and therefore widens the idea space.

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM
How might we improve customer satisfaction on our chatbot and build an enhanced version of it?

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

Yuvan Shankar J

- Provide personalized service to all the customer
- Provide best solution possible at all circumstances
- Avoid misinterpretation of messages
- Perform simple testing on the product created to eliminate errors
- Provide helpline number for emergency situations

Kavyapriya V K

- Resolve complaints
- Send alerts for suspicious activities
- Prioritize issues
- Retrieve bank balance through password
- Trustworthy

Shwetha MK

- Consideration of customer feedback for improvisation of facilities
- Ensured privacy
- Instant response
- Clarification of all possible doubts
- 24/7 availability of services

Karthick V

- Predefined questions
- Avoid blocks of text as reply from chatbot and stick to the point
- Avoid all unwanted messages
- Viability of data
- Improve service over time

Web application

- 24/7 availability of services
- Predefined questions
- Clarification of all possible doubts
- Provide personalized service to all the customer
- Retrieve bank balance through password

Cloud technology

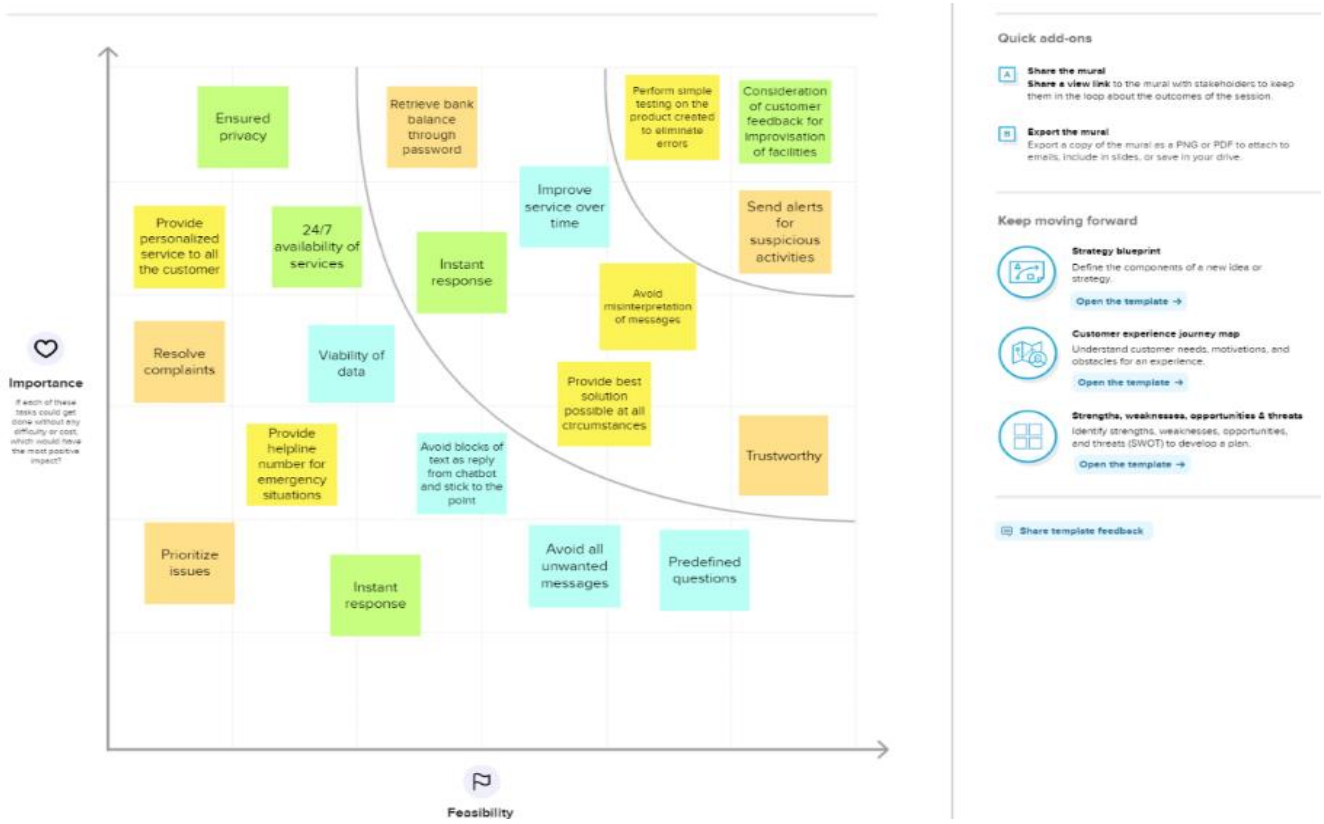
- Trustworthy
- Provide best solution possible at all circumstances
- Consideration of customer feedback for improvisation of facilities
- Making data secure
- Prioritize issues

Customer satisfication

- Provide helpline number for emergency situations
- Improve service over time
- Instant response
- Resolve complaints
- Perform simple testing on the product created to eliminate errors

Quality service

- Avoid misinterpretation of messages
- Avoid all unwanted messages
- Avoid blocks of text as reply from chatbot and stick to the point
- Ensured privacy
- Send alerts for suspicious activities

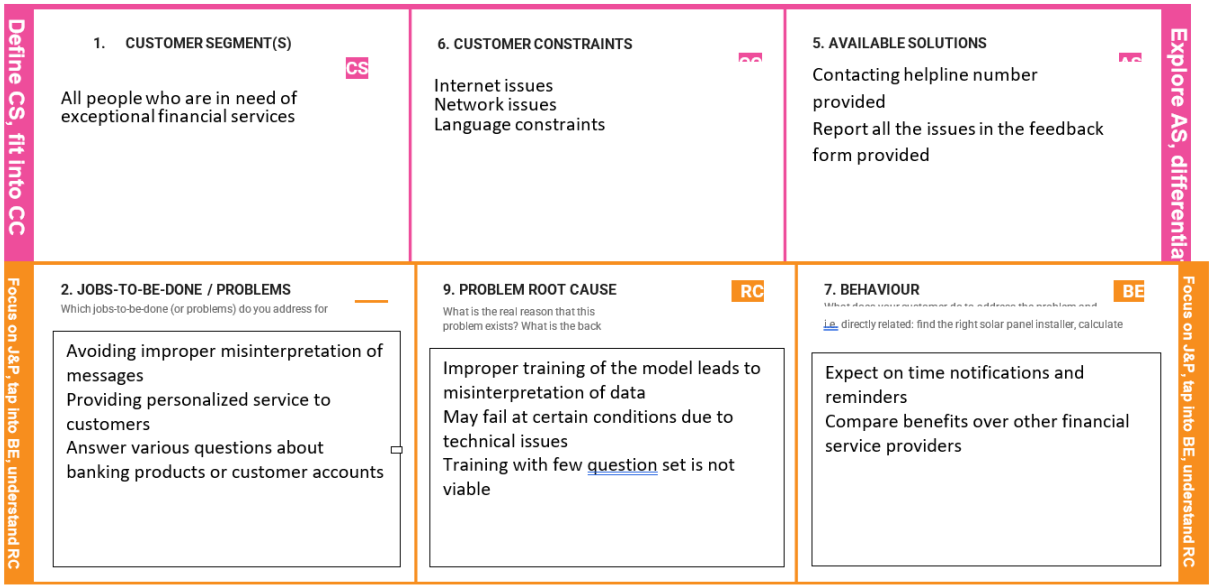


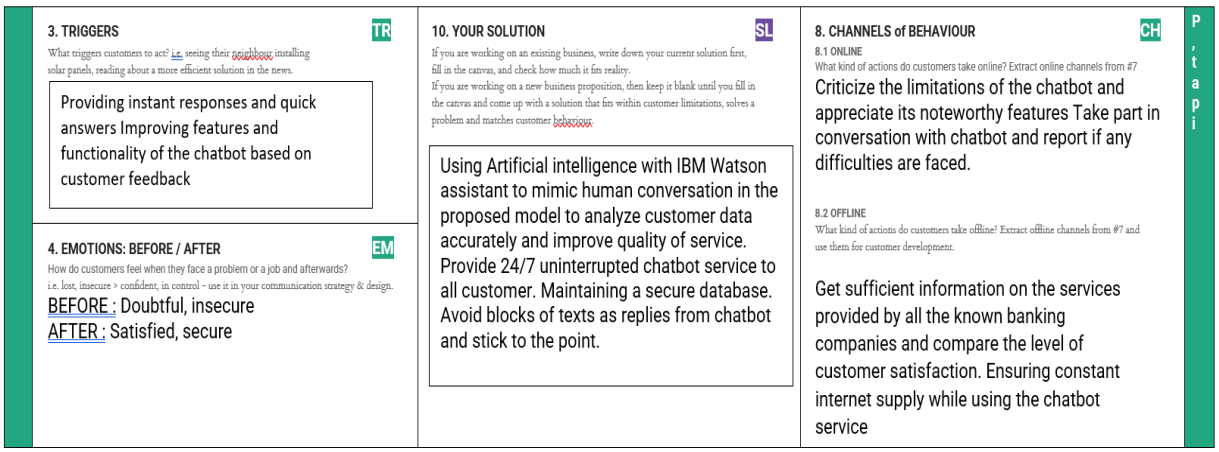
3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To create a chatbot that can answer various questions about banking products or customer accounts and provides an enhanced way of interaction.
2.	Idea / Solution description	Using Artificial Intelligence with IBM Watson assistant to mimic human conversation in the proposed model to analyse customer data accurately and improve quality of service
3.	Novelty / Uniqueness	Users can ask the chatbot to provide the balances for accounts under their name. The bot can even be used to gather feedback from users on how to improve their experience.
4.	Social Impact / Customer Satisfaction	24/7 uninterrupted chatbot service to all the customers with timely improvements of the proposed model based on customer feedback. Handle complex queries by well structured human and computer interaction.

5.	Business Model (Revenue Model)	Free, efficient, and advanced chatbot service will grab public attention which will surely have a huge positive impact on revenue generated by the company. Also, it helps organisations to automate complex processes.
6.	Scalability of the Solution	Broad positive impact for users and connect them globally. Performance tests can be performed on a regular basis to ensure gradual increase in efficiency of the system without hampering the existing workflow

3.4 Problem Solution fit





4.REQUIREMENT ANALYSIS

4.1 Functional requirements

Following are the functional requirements of the proposed solution

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Bank Account Action	Chatbot should guide the customer to create bank accounts and provide solutions to all related queries
FR-2	General Queries Action	Chatbot must provide best solutions to all general financial services from the customer side.
FR-3	Net Banking Action	Chatbot should be able to guide the customer through net banking registration and explore various options available.
FR-4	Loan Queries Action	Chatbots should ensure that customer who is in need of a loan is aware of all the types of loans and procedures.
FR-5	Speed	Chatbots should be programmed in such a way that they can fetch information and respond quickly thereby allowing users to make hassle-free payments within seconds.
FR-6	User Interface	A webpage has to be developed to provide Chatbot service with elegant user interface
FR-7	Storage	Maintaining database to store data provided by the users in Net Banking registration and Account creation forms.

4.2 Non- functional requirements

Following are the non functional requirements of the proposed solution

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Visually appealing chatbot user interface is easy to use and also provide seamless customer service at one touch
NFR-2	**Security**	Information of customers which are stored in database has to be protected with high level of security.

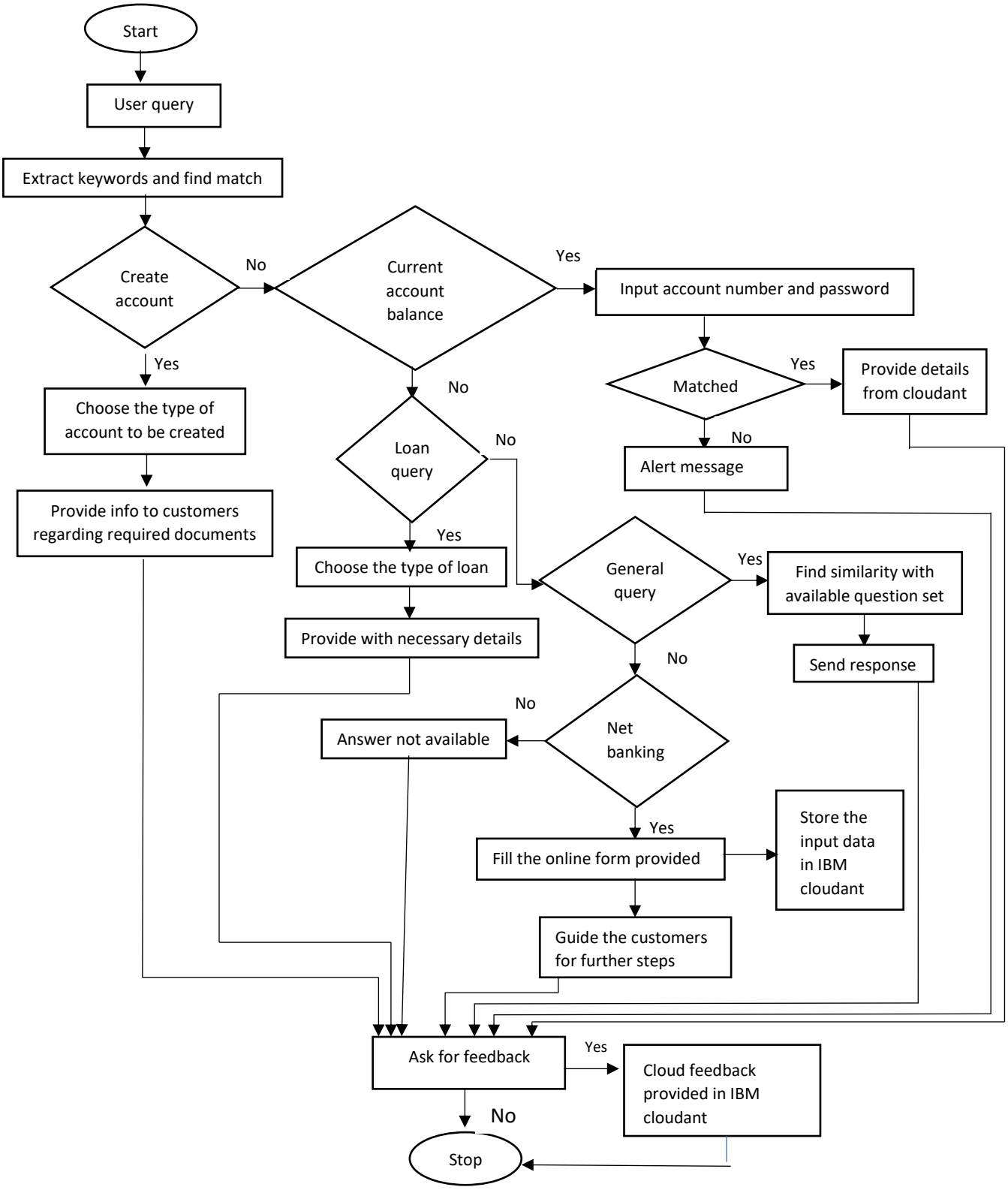
NFR-3	**Reliability**	High level of security ensures reliability and consistent performance attracts more users to the service.
NFR-4	**Performance**	Chatbots reduce human errors and provide customer service to millions of users at an instance of time.
NFR-5	**Availability**	Chatbots provide instant answers throughout a day, subsequently reducing human work load and increasing the number of satisfied customers
NFR-6	**Scalability**	Extending functionality and features of the system on a regular basis based on customer feedback.

5. PROJECT DESIGN

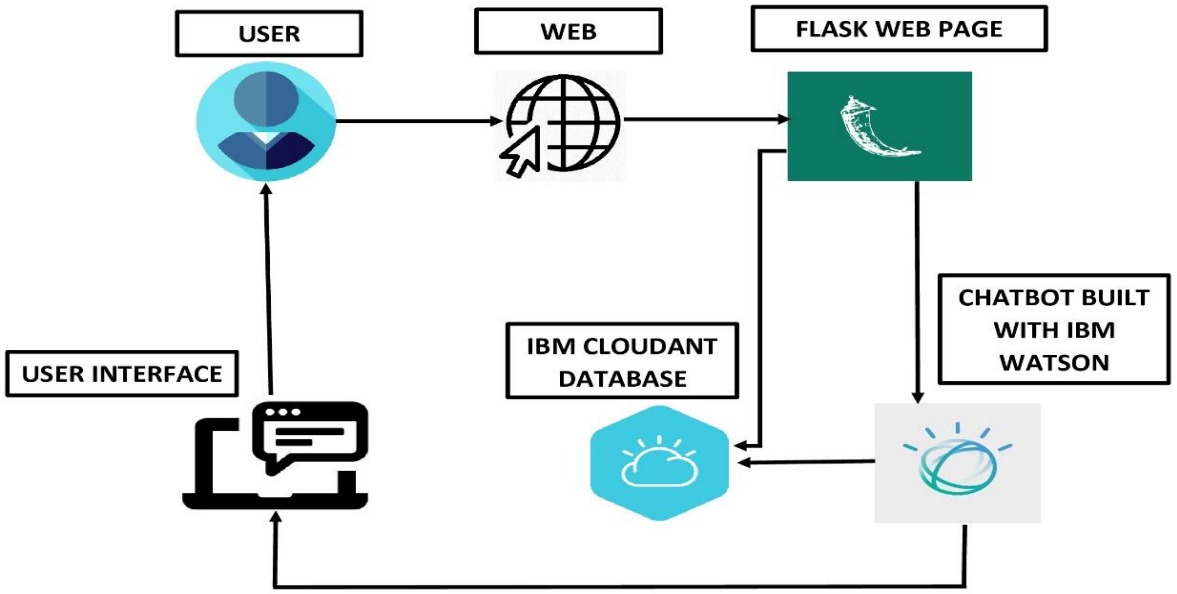
5.1 Data Flow Diagrams

Data flow diagrams are used to represent the flow of data as well as the processes and functions involved to store, manipulate, and distribute data among various components of the system and between the system and the environment of the system by a specific set of graphical representations. It also depicts the logical flow of information in a system and appropriately defines and determines the physical requirements for the construction of the system.

The diagram below consists of the overall application dataflow and processes of the banking process. It consists of the user flow and entities such as Savings account, Current account balance, loan queries, net banking queries and general queries.


Start
User query
Extract keywords and find match
Create account
No
Current account balance
Yes
Input account number and password
Yes
Choose the type of account to be created
Provide info to customers regarding required documents
No
Loan query
No
Yes
Choose the type of loan
Provide with necessary details
Matched
Yes
Provide details from cloudant
No
Alert message
General query
Yes
Find similarity with available question set
Send response
No
Net banking
No
Answer not available
Yes
Fill the online form provided
Store the input data in IBM cloudant
Guide the customers for further steps
Ask for feedback
Yes
Cloud feedback provided in IBM cloudant
No
Stop

5.2 Solution & Technical Architecture



5.3 USER STORIES

Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
IBM Watson	USN-1	As a user, I should be able to access the web page that is integrated with the chatbot	I can access the chatbot integrated with webpage	High	Sprint 1
	USN-2	As a user, I should be able to clear my banking related queries	I can clear my queries related to banking industry	Medium	Sprint 1
Savings account action	USN-3	As a user, I should be able to select my desired type of Savings Account to get details regarding documents required for creation.	I can clear my queries regarding all types of savings account	Medium	Sprint 2

Current account action	USN-4	As a user, I should be able to choose the type of company to get details regarding the documents to be submitted	I can clear all my queries regarding the two types of companies	Medium	Sprint 2
	USN-5	As a user, I can get my current account balance through one step verification process	I can request to maintain database for reference	High	Sprint 2
Loan queries	USN-7	As a user, I can know the different types of loans available	I can gain knowledge about the different types of loans	High	Sprint 2
	USN-8	As a user, I can know the eligibility criteria for all the loan schemes	I can know the criterias for loan schemes	High	Sprint 2
Net banking queries	USN-9	As a user, I can clear all my doubts regarding net banking	I can get all my doubts cleared	High	Sprint 2
	USN-10	As a user, I can access a net banking registration form before heading to a bank	I can complete my registration process	High	Sprint 2
General queries	USN-11	As a user, I can get answers for general banking queries	I can get answers for all the frequently asked questions	Medium	Sprint 2
Customer needs	USN-12	As a user, I can request high quality service from the provider	I can demand for profile verification before giving confidential data	High	Sprint 3
Customer care	USN-13	As a user, I can request developers for help if any if my query is unanswered.	I can ensure smooth user experience and report all the issues raised to the development team.	High	Sprint 4

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint planning is an event in the Scrum framework where the team determines the product backlog items on which they will work on during that sprint and discusses their initial plan for completing those product backlog items. The sprint goal describes the objective of the sprint at a high level, but the backlog items can also be written with an outcome in mind. User stories are one great way of describing the work from a customer point of view. User stories re-focus defects, issues, and improvements on the outcome the customer is seeking rather than the observed problem.

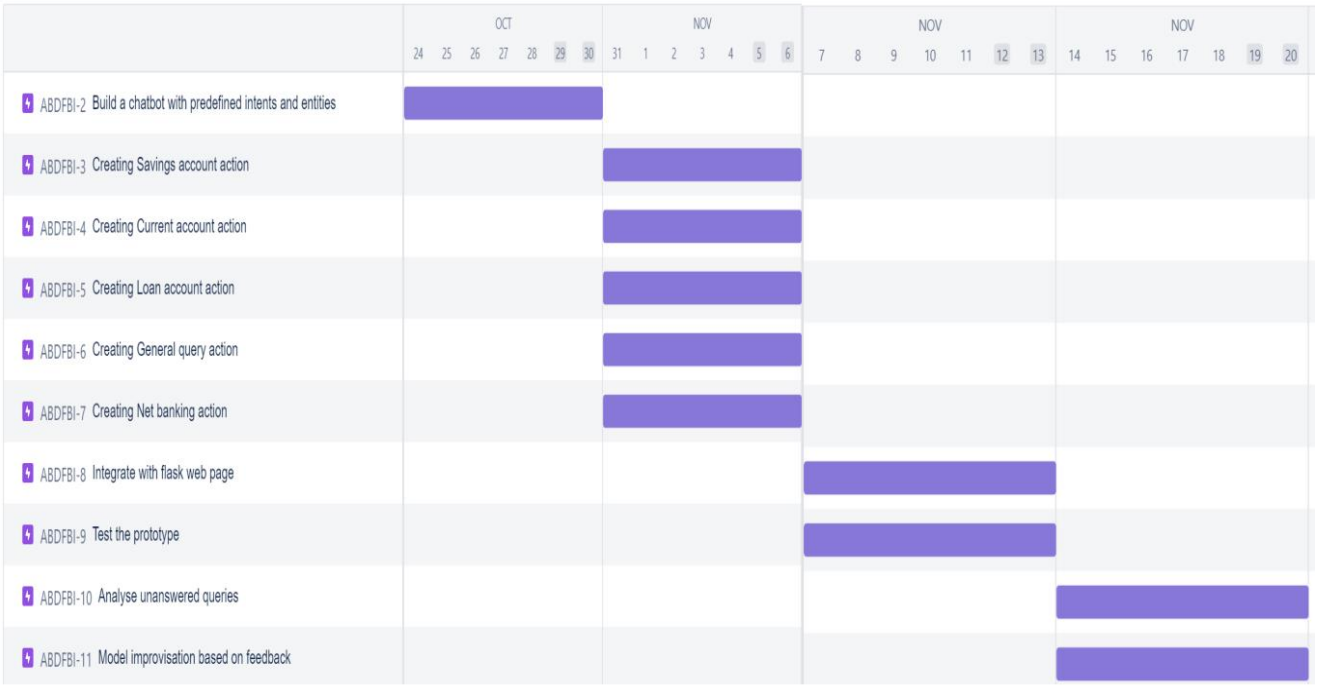
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Building IBM Watson assistant	USN-1	As a user, I can use a chatbot with predefined system intents and entities to accomplish my goals.	12	High	Yuvan Shankar J, Kavyapriya V K
Sprint-1		USN-2	As a user, I can use a chatbot that understands and provides answers to banking related queries of customers.	8	Medium	Shwetha M K, Karthick V
Sprint-2	Modelling	USN-3	As a user, I can converse with a chatbot regarding creation of bank account of desired type.	5	High	Yuvan Shankar J, Karthick V
Sprint-2		USN-4	As a user, I can see a chatbot that assists in loan related queries	5	Medium	Kavyapriya V K, Shwetha M K
Sprint-2		USN-5	As a user, I can request a chatbot to guide in addressing general banking queries	5	High	Yuvan Shankar J, Shwetha M K
Sprint-2		USN-6	As a user, I can converse with a chatbot that facilitates net banking services to do transactions online	5	High	Kavyapriya V K, Karthick V
Sprint-3	User Interface and Testing	USN-7	As a user, I must have a smooth chat experience with good user interface satisfying all my expectations.	10	High	Yuvan Shankar J, Kavyapriya V K
		USN-8	Testing the prototype created to ensure proper interpretation of messages and high-level performance	10		Shwetha M K, Karthick V
Sprint-4	Model improvisation	USN-9	As a user, I can provide feedback on conversation with the chatbot and specify the changes to be made to increase my level of satisfaction.	8	High	Yuvan Shankar J, Kavyapriya V K, Shwetha M K
			Improve service by analyzing the drawbacks of the prototype	7		Kavyapriya V K, Shwetha M K, Karthick V
			Analyzing customer's unanswered queries	5	Medium	Yuvan Shankar J, Karthick V

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA

ROAD MAP

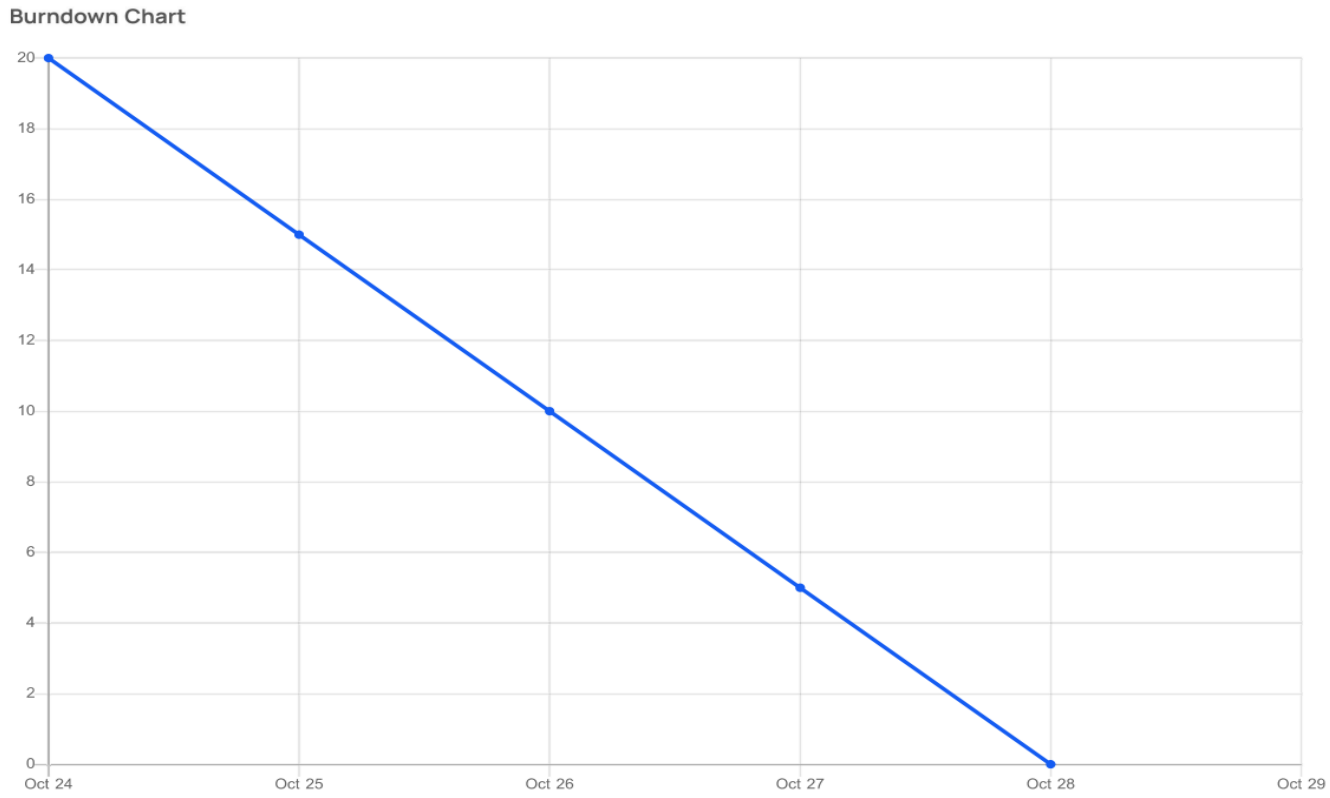


BURNDOWN CHART

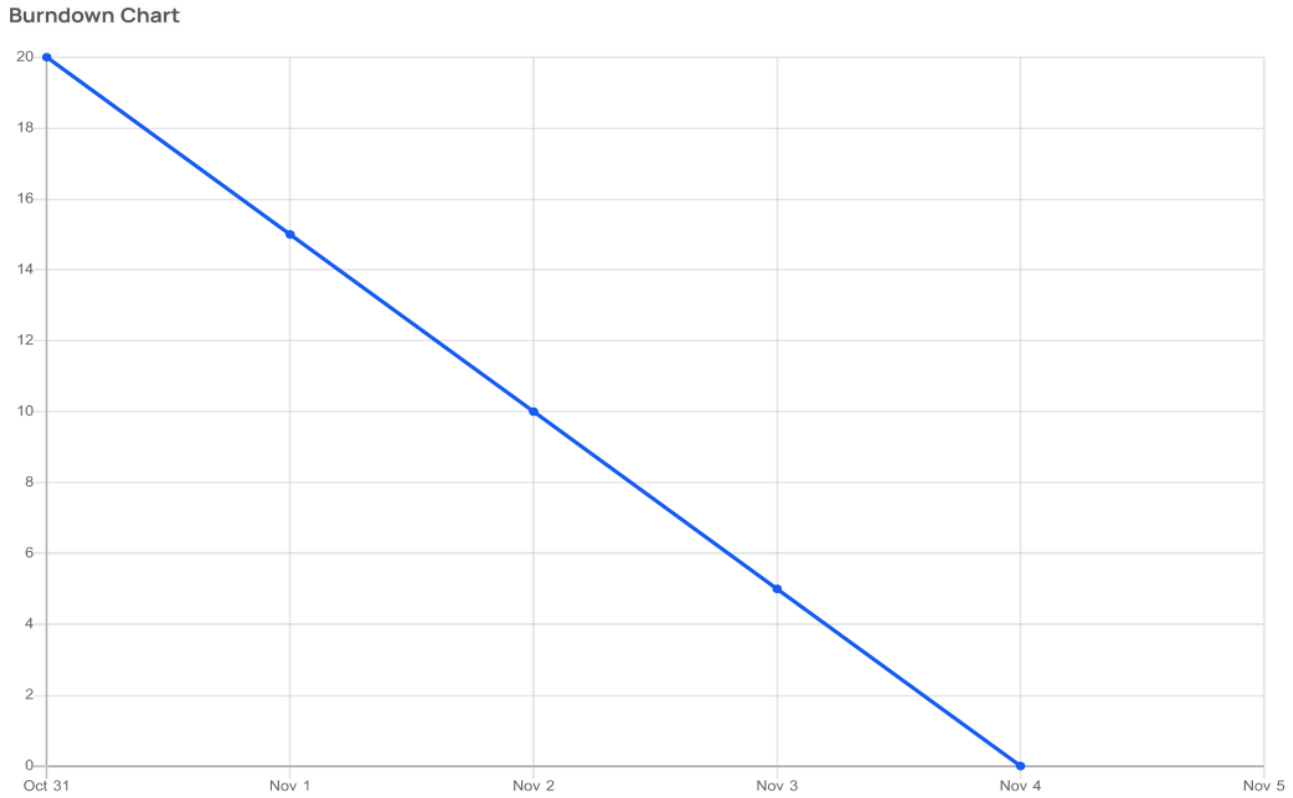
The Burndown Chart in Jira is a powerful tool that helps visualize and calculate tasks within a sprint. Using it allows you to make the best decisions and draw the most valuable conclusions in all sprint phases, from planning to processing and completion. It helps to plan the sprint as accurately as possible. A burndown chart is a graphical representation of work left to do versus time. It shows how quickly you and your team are burning through your customer's user stories.

We divided this into four sprints by laying out the work to be performed for each sprint. The objective of sprint planning is to work out the key details regarding the team's planned work during each sprint. The burndown chart for each sprint are as follows.

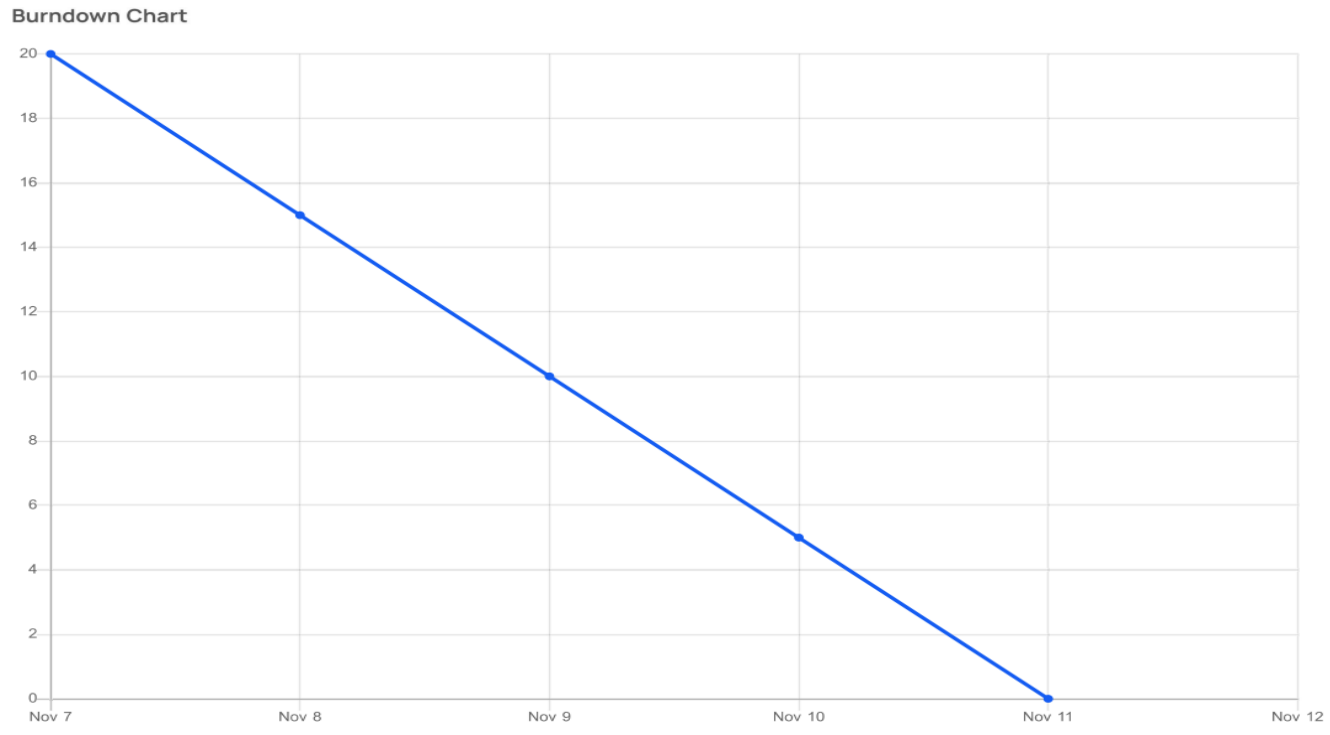
SPRINT 1



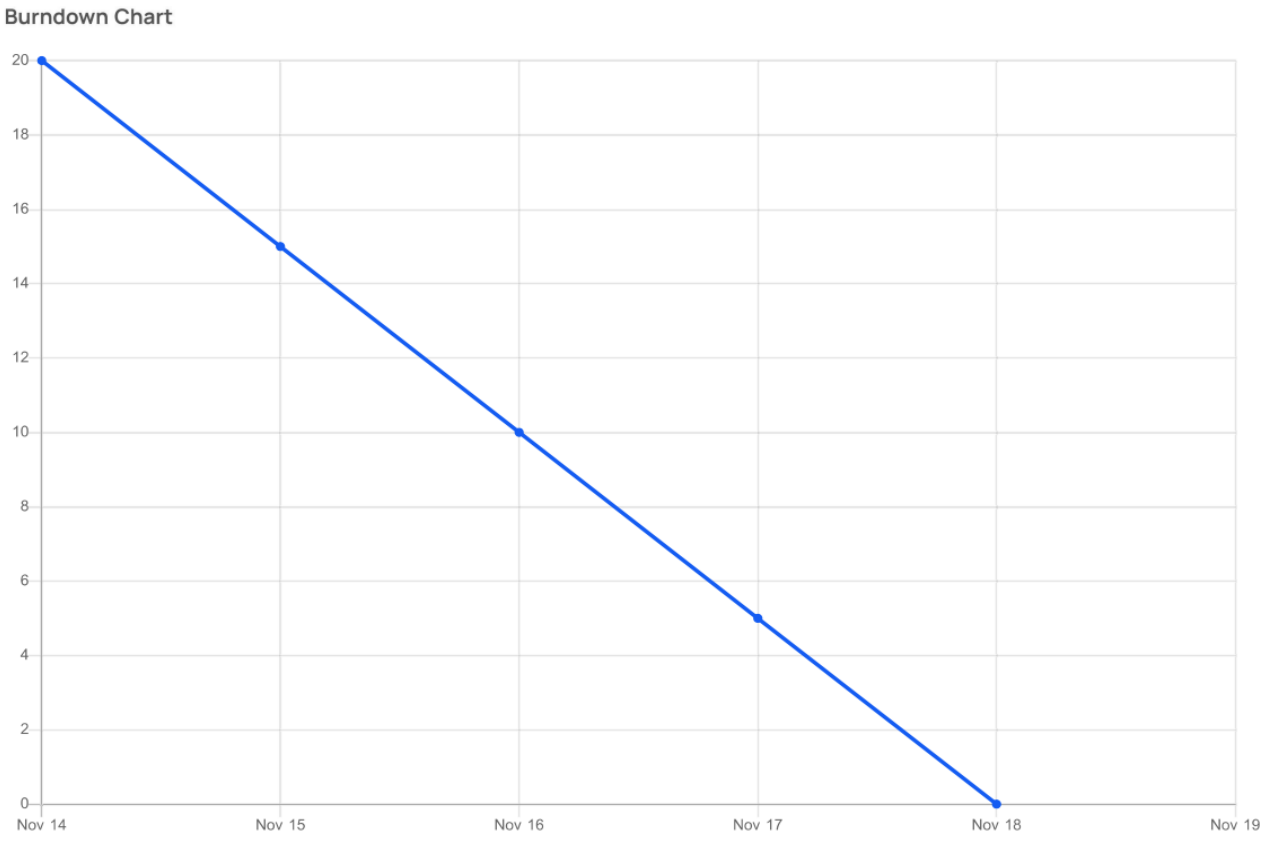
SPRINT 2



SPRINT 3



SPRINT 4

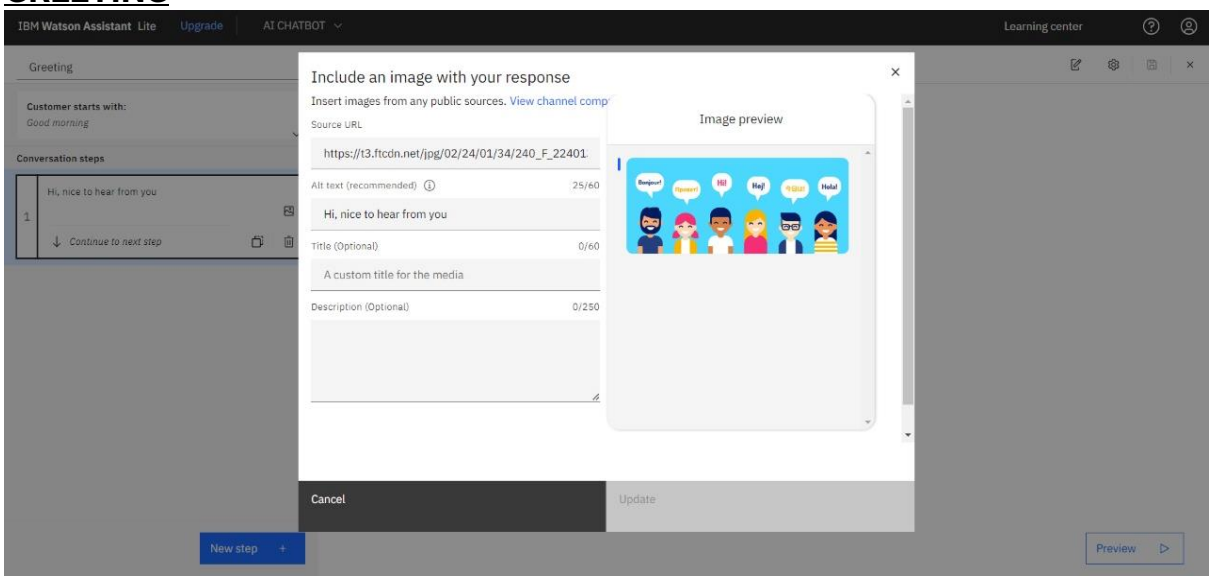

Burndown Chart
20
18
16
14
12
10
8
6
4
2
0
Nov 14
Nov 15
Nov 16
Nov 17
Nov 18
Nov 19

7.CODING & SOLUTIONING

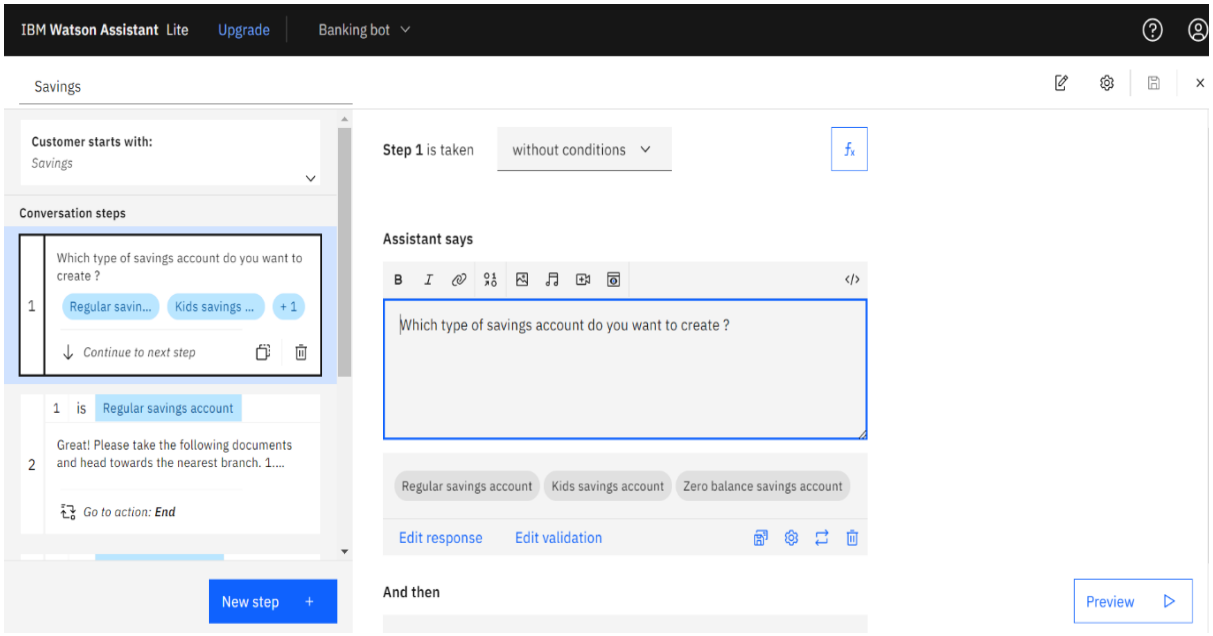
7.1 Feature 1

Firstly, we have created the skills and actions provided in the guided projects under project workspace.

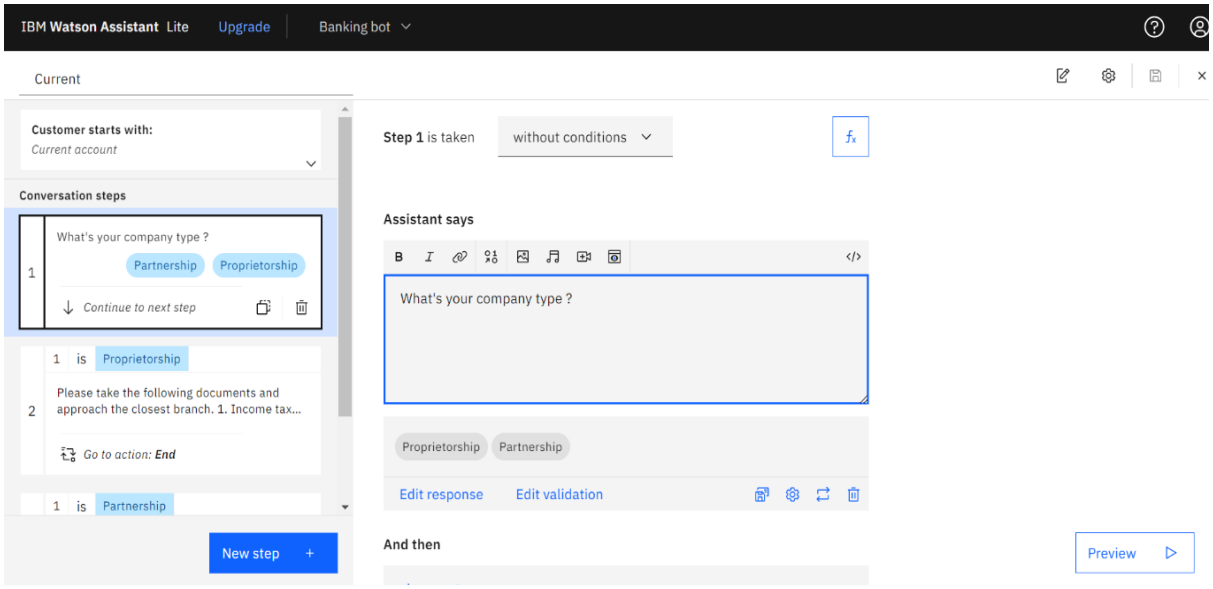
GREETING



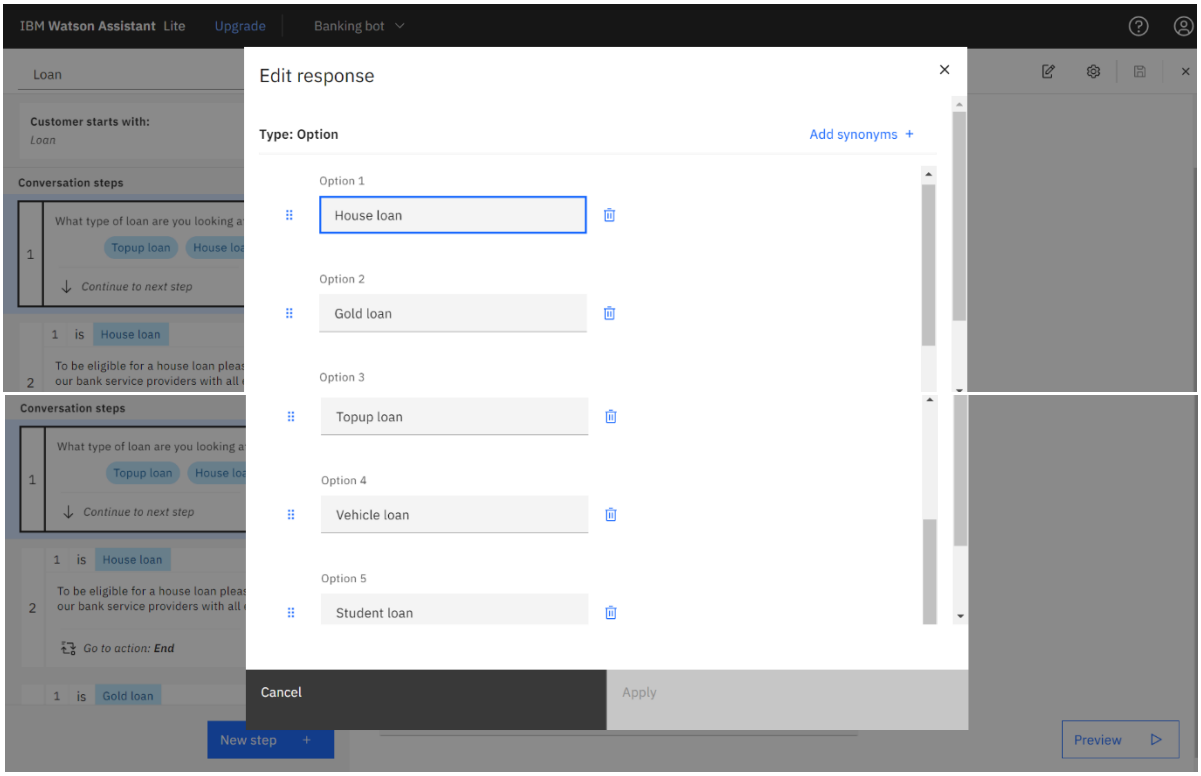
CREATING SAVINGS ACCOUNT ACTION



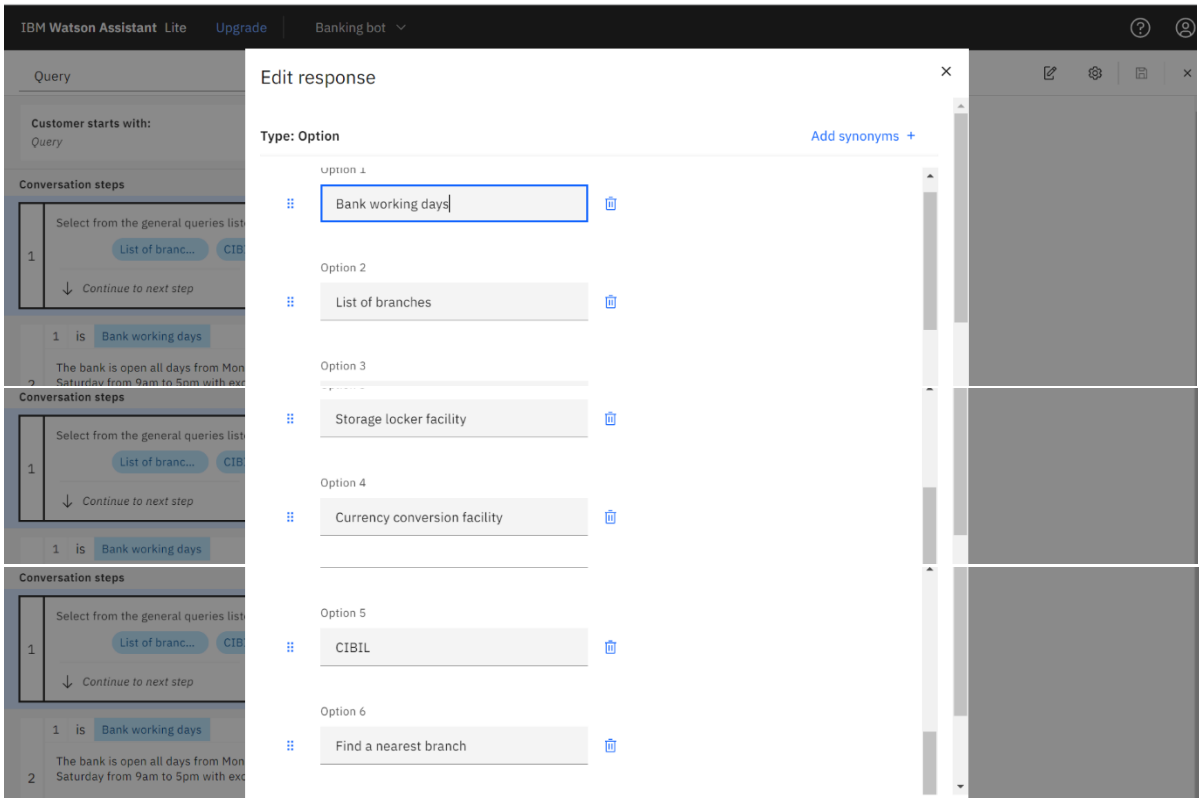
CREATING CURRENT ACCOUNT ACTION



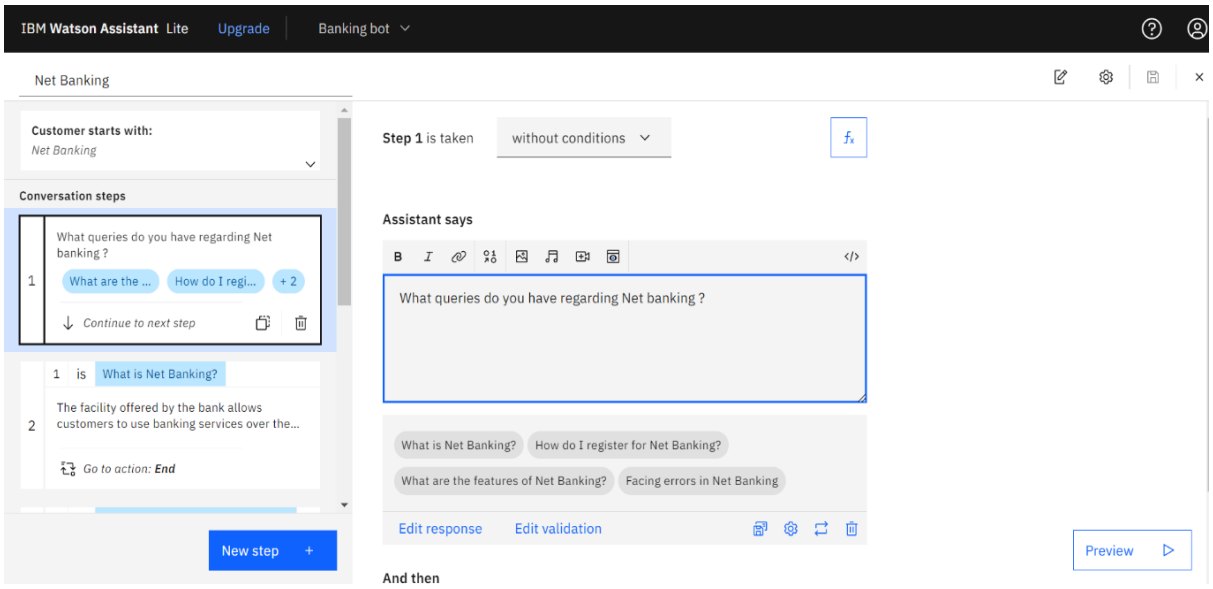
CREATING LOAN ACCOUNT ACTION



CREATING GENERAL QUERY ACTION



CREATING NET BANKING ACTION



7.2 FEATURE 2

In addition to the existing actions, we have added some unique features such as

- Checking account balance after authentication.
- Storing user feedback in IBM cloudant database.
- Storing input data from net banking registration in IBM cloudant database.

Mainly, we have focused on developing a database to store user data and improving the features of our chatbot by adding new actions to it.

We have used **IBM cloudant** to create the required databases for our model.

Offers, current account balance, feedback data storage and net banking registration are the new features added in this sprint. Links for current account balance, feedback form and net banking registration form is provided by the chatbot asresponse and they are also provided in the webpage created.

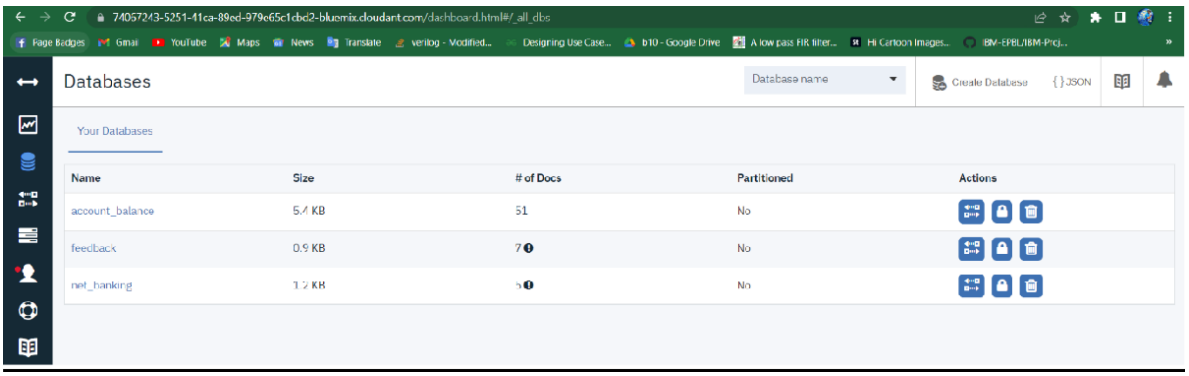
IMPORTING REQUIRED LIBRARIES:

```
from flask import Flask,render_template
from flask import *
from cloudant.client import Cloudant
from cloudant.result import Result
```

INTEGRATING IBM CLOUDANT DATABASE:

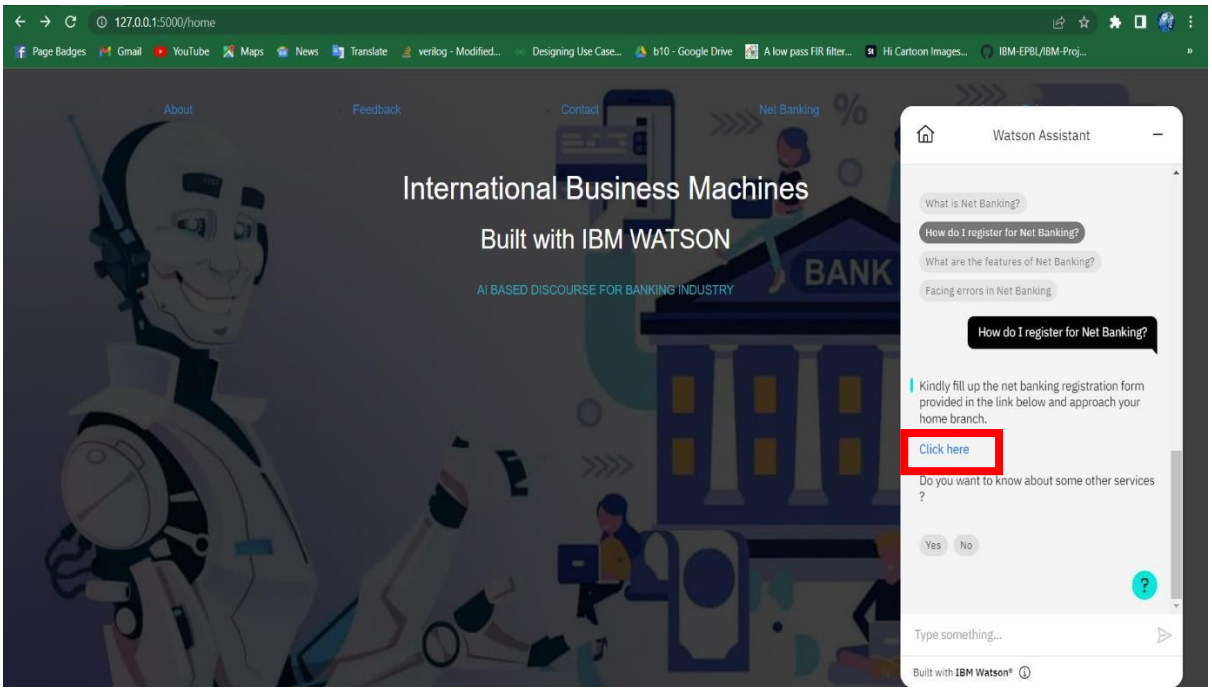
```
ACCOUNT_NAME = "74067243-5251-41ca-89ed-979e65c1cbd2-bluemix"
API_KEY = "IqWgY3pe1n9DGN4pN_9uSXzmlCs27ML4DkcTAePwkFbv"

client = Cloudant.iam(ACCOUNT_NAME,API_KEY,connect=True)
```

DIFFERENT DATABASES CREATED:



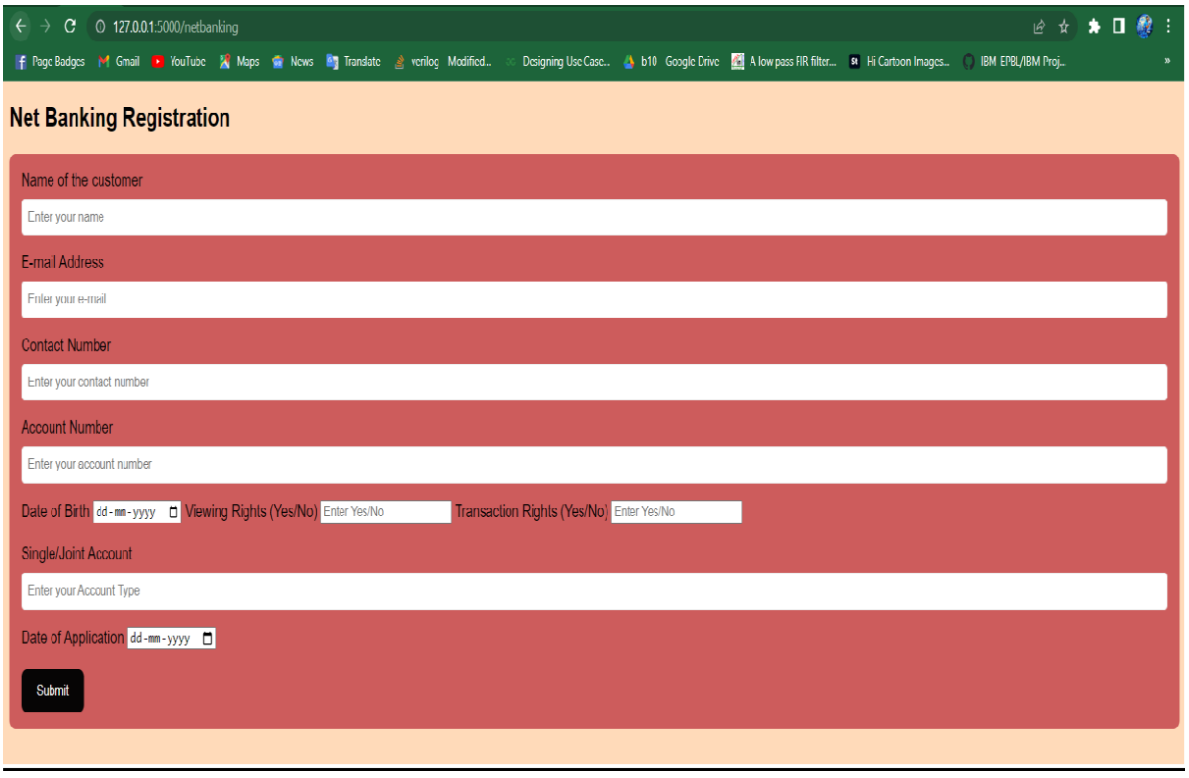
NET BANKING:

We have added a link in the chatbot which is given as a response when the user asks about net banking registration.



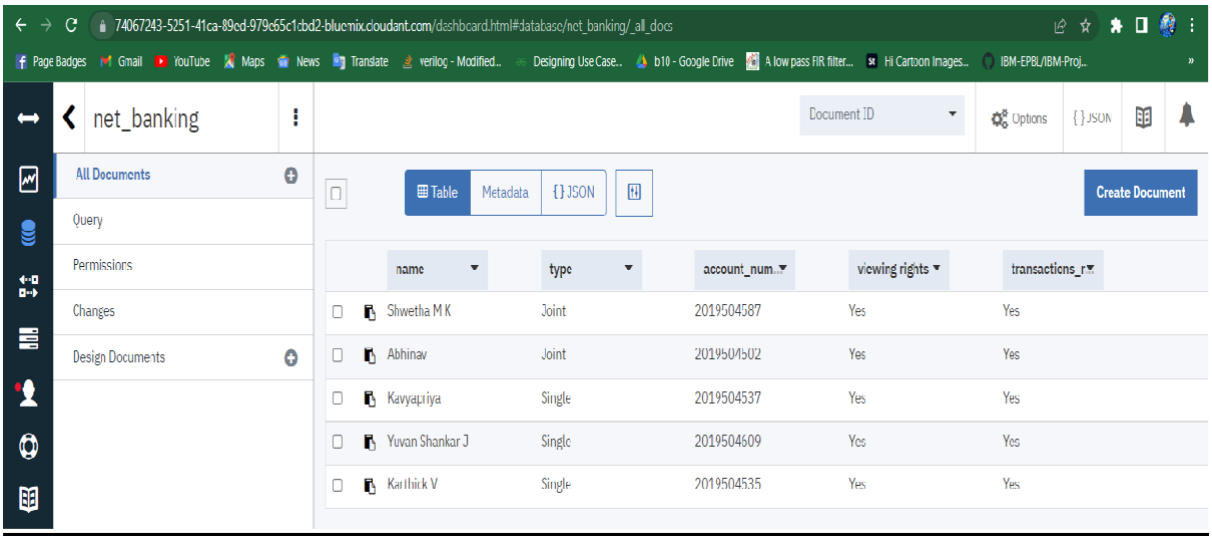
The link “click here” will direct the user to net banking registration form.

NET BANKING REGISTRATION FORM:



STORAGE OF USER DATA PROVIDED IN IBM CLOUDANT DATABASE:

The data provided by the users in the registration form is stored in the net_banking database as shown below.



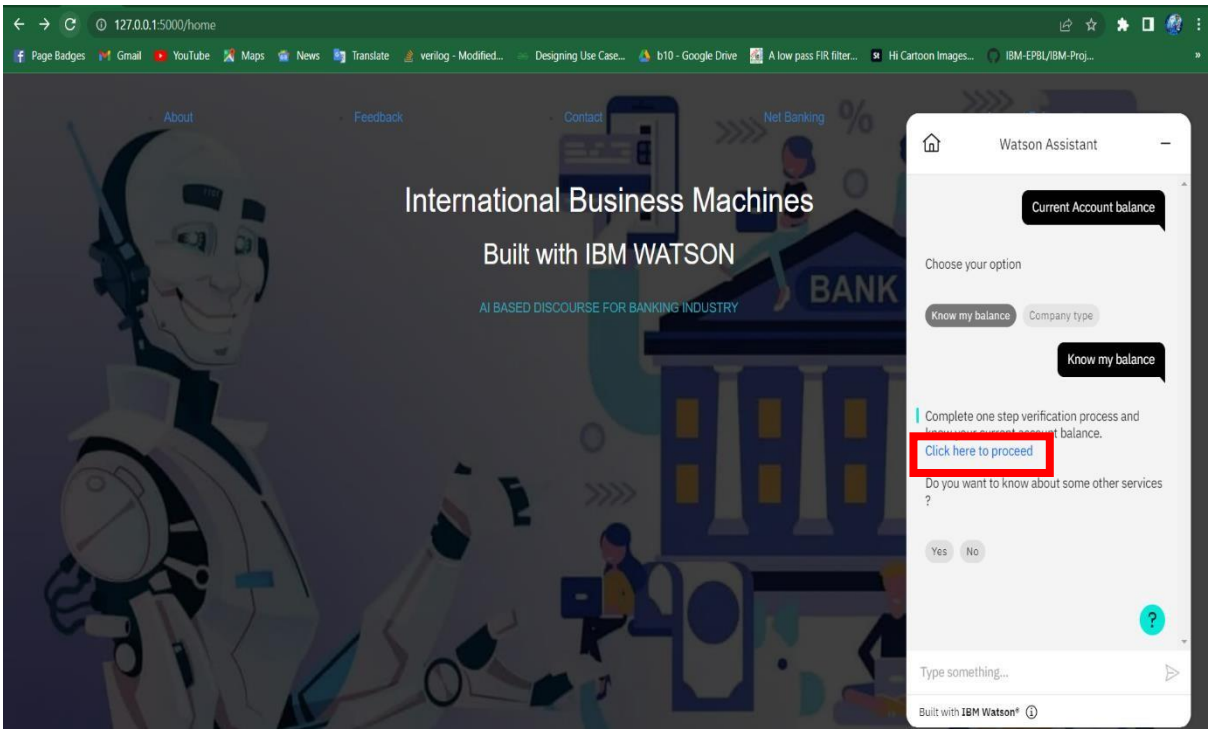
PYTHON CODE FOR USER DATA STORAGE:

```
@app.route('/netbanking',methods = ['GET','POST'])
def netbanking():
    print("******")
    if request.method == "POST":
        email = request.form['mail']
        name = request.form['name']
        acc_num = request.form['num']
        contact_number=request.form['number']
        date_of_birth=request.form['dob']
        date_of_application=request.form['doa']
        viewing_rights=request.form['viewing']
        transaction_rights=request.form['transaction']
        type_of_account=request.form['type']
        jsonDocument= {
            'email':email,
            'name':name,
            'account_number':acc_num,
            'contact_numer':contact_number,
            'viewing rights':viewing_rights,
            'transactions_rights':transaction_rights,
            'date_of_birth':date_of_birth,
            'date_of_application':date_of_application,
            'type':type_of_account
            }
        newDocument = mydatabase1.create_document(jsonDocument)
        result = Result(mydatabase1.all_docs,include_docs=True)
        print(result[0])
        return redirect(url_for('home'))
        print('#####')
    return render_template('netbanking.html')
```

ACCOUNT BALANCE:

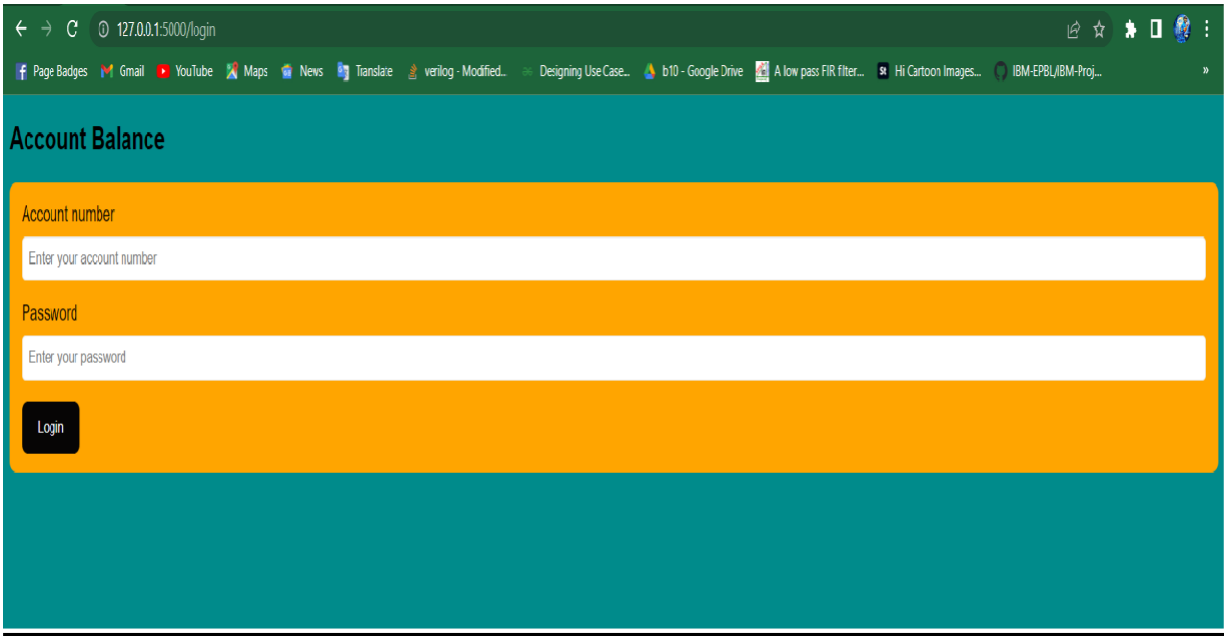
We have added a new feature for the users to find their account balance with one step authentication.

When the user asks for his account balance, a link appears as a response which will direct the user to the login page.



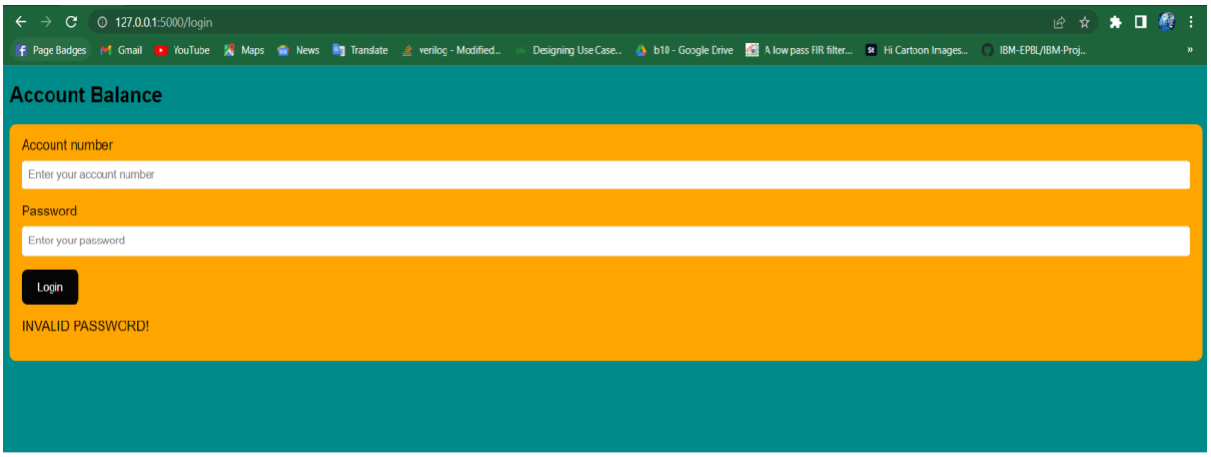
ACCOUNT BALANCE WEBPAGE:

The 'click here' link shown in the above image directs the user to the login page.

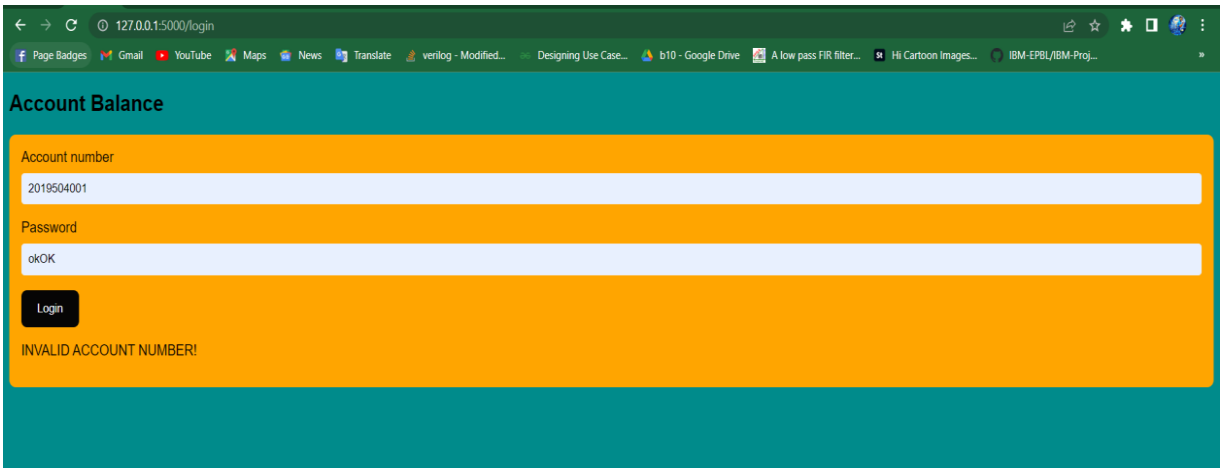


The information given by the user is validated and the following messages are shown based on the result of validation.

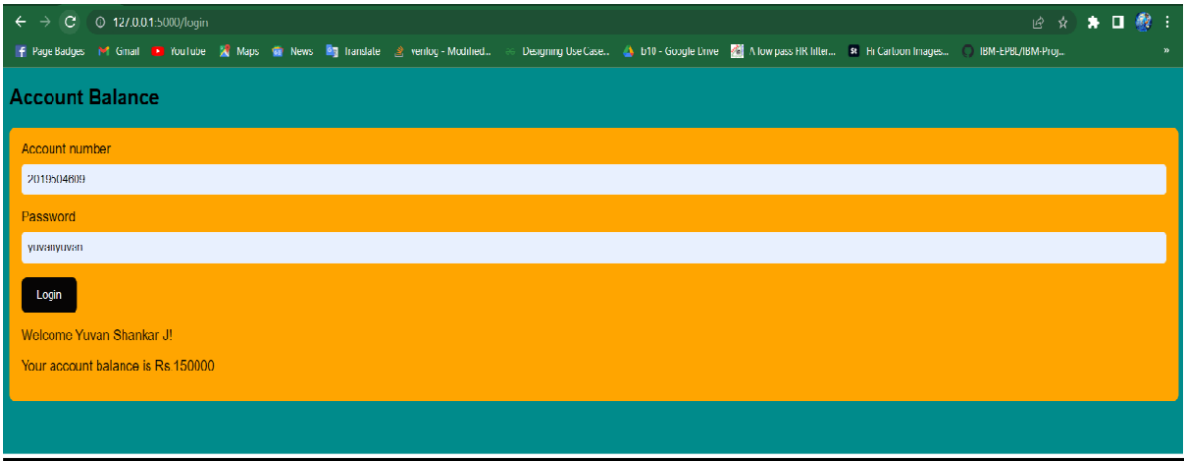
CASE 1: ACCOUNT NUMBER AND PASSWORD MISMATCH:



CASE 2: ACCOUNT NUMBER IS INVALID



CASE 3: ACCOUNT NUMBER AND PASSWORD MATCH

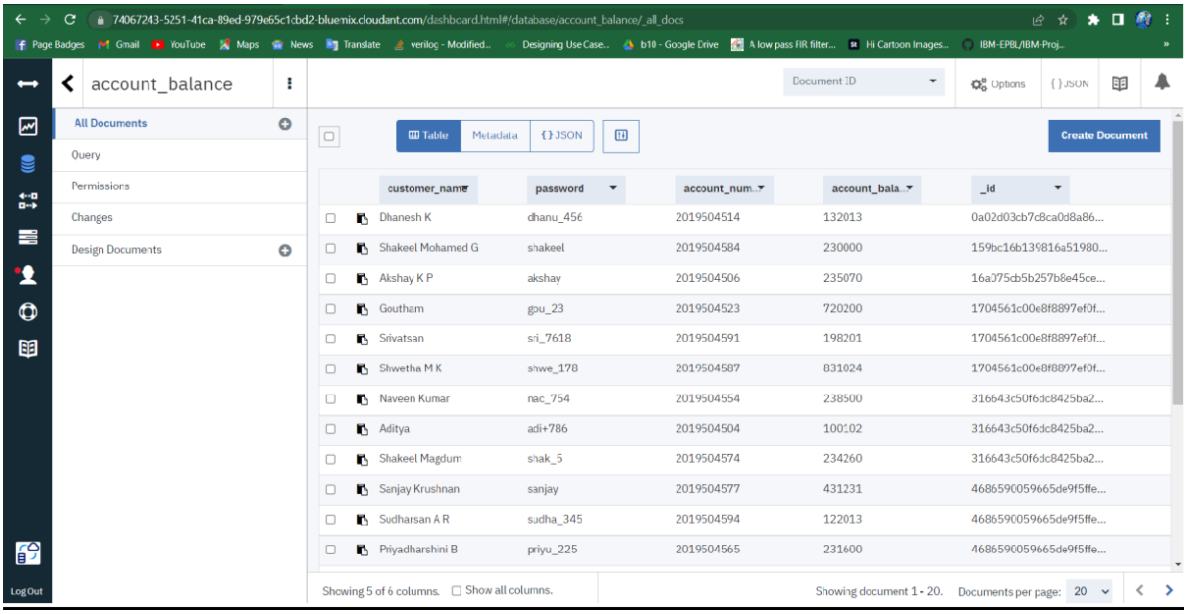


PYTHON CODE FOR USER DATA VALIDATION:

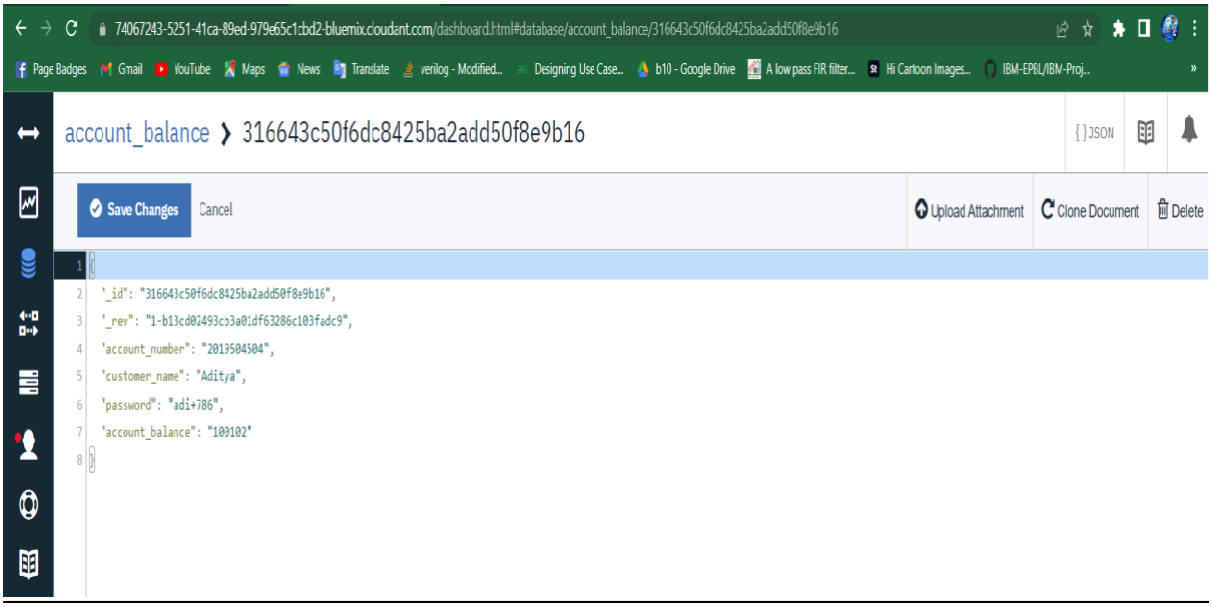
```
mydatabase2=client.create_database('account_balance')
@app.route('/login',methods=['GET','POST'])
def login():
    if request.method=='POST':
        account_number=request.form['accnum']
        password=request.form['pass']
        account_found=False
        for documents in mydatabase2:
            if documents['account_number']==account_number:
                if documents['password']==password:
                    flash_name='Welcome'+' '+documents['customer_name']+'!'
                    flash(flash_name)
                    flash_text='Your account balance is'+' '+"Rs."+documents['account_balance']
                    flash(flash_text)
                else:
                    flash('INVALID PASSWORD!')
                account_found=True
                break
        if not account_found:
            flash('INVALID ACCOUNT NUMBER!')
    return render_template('login.html')
```

ACCOUNT BALANCE DATABASE:

The input data provided by the user is validated using the python code above. Validation is done by comparing the inputs given and the data available in the database. If the account number and password provided matches with account number and password of any document in the database, the corresponding account balance is given as output.

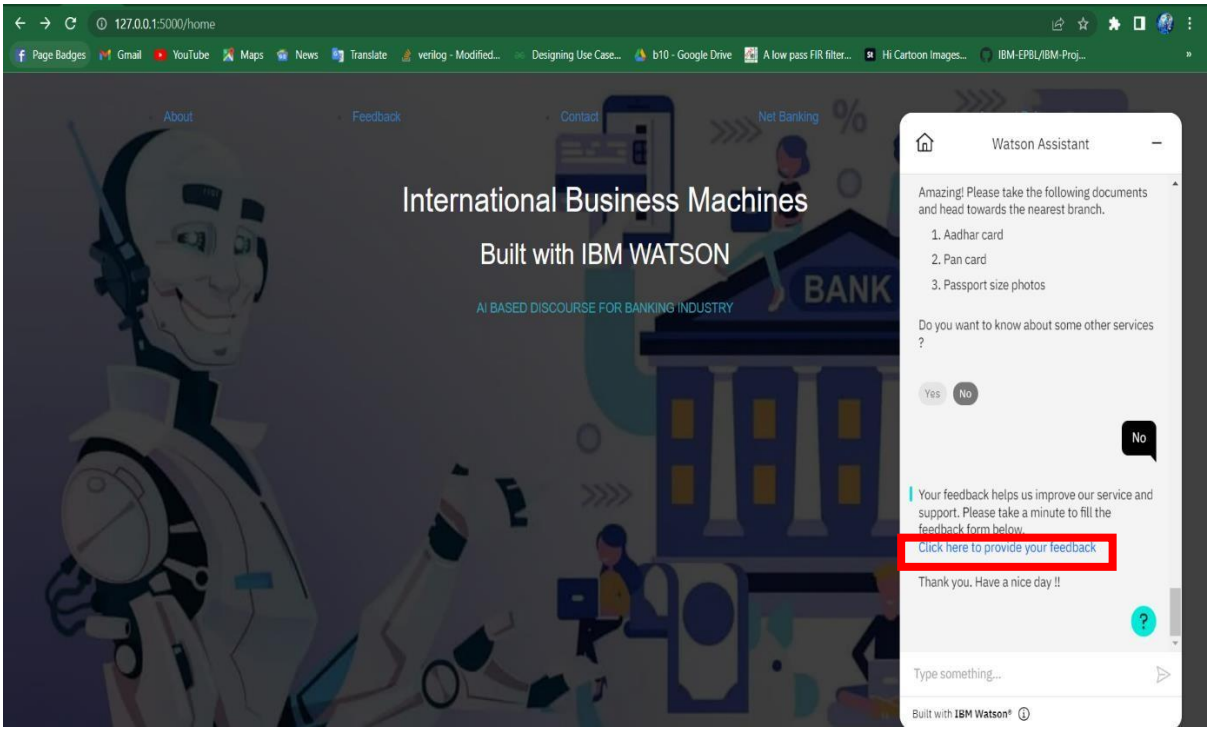


A SAMPLE DOCUMENT STORED IN THE ACCOUNT BALANCE DATABASE:



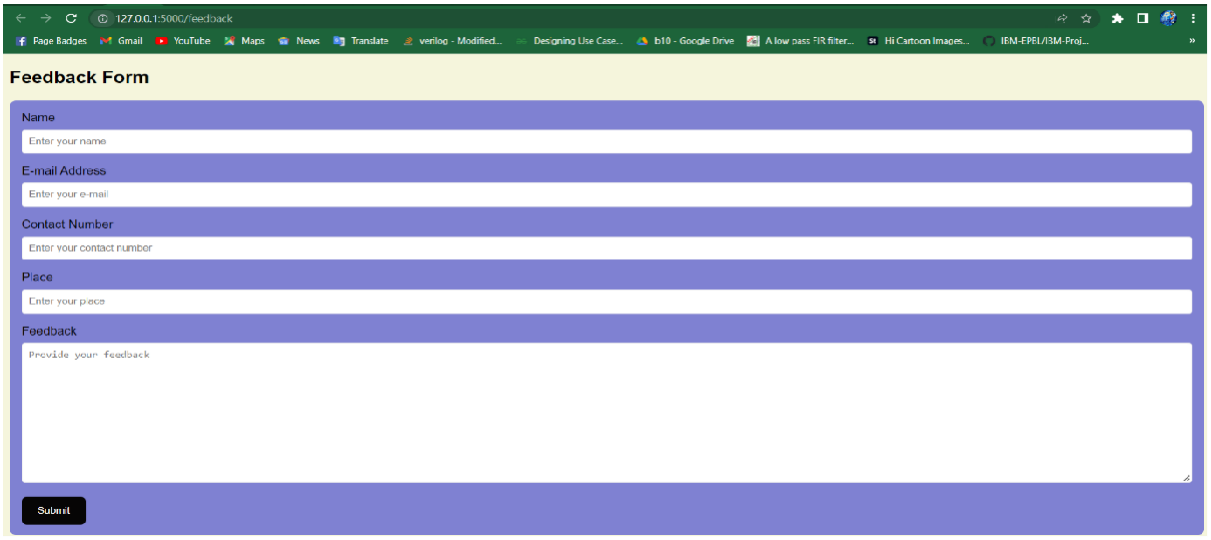
FEEDBACK:

A feedback form has been provided for the users at the end of a conversation to share their experience and mention any difficulties faced by them while they use the chatbot.



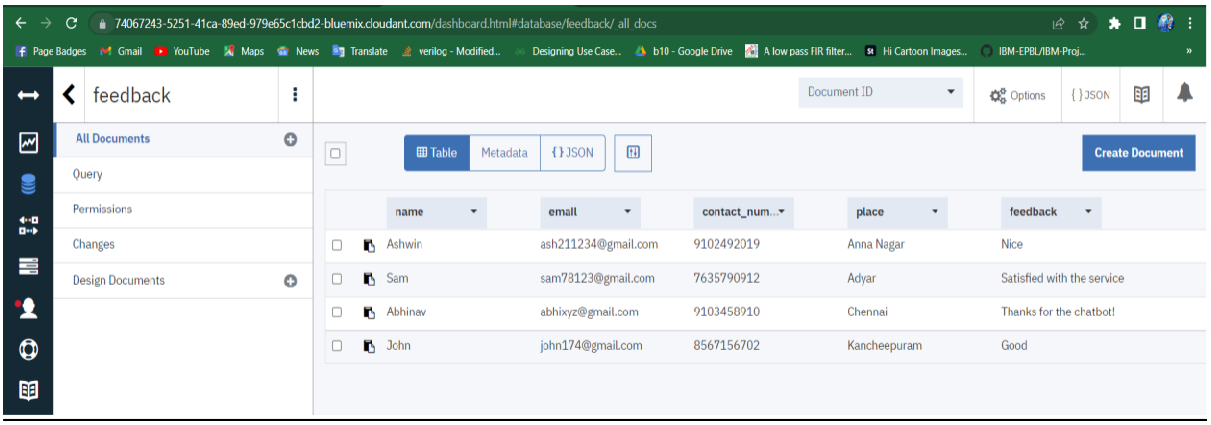
The link 'Click here to provide your feedback' will direct the user to feedback form webpage.

FEEDBACK FORM WEBPAGE:



STORAGE OF USER FEEDBACKS IN IBM CLOUDANT DATABASE:

Data provided by the users is stored in the feedback database as shown below.



PYTHON CODE FOR FEEDBACK DATA COLLECTION:

```
@app.route('/feedback',methods = ['GET','POST'])
def feedback():
    print("******")
    if request.method == "POST":
        email = request.form['mail']
        name = request.form['name']
        num = request.form['num']
        place= request.form['place']
        msg = request.form['message']
        #print(email,name,num,place,msg)
        jsonDocument= {
            'email':email,
            'name':name,
            'contact_number':num,
            'place':place,
            'feedback':msg
            }
        newDocument = mydatabase.create_document(jsonDocument)
        result = Result(mydatabase.all_docs,include_docs=True)
        print(result[0])
        return redirect(url_for('home'))
        print('#####')
    return render_template('feedback.html')
```

OFFERS:


International Business Machines
Built with IBM WATSON
AI BASED DISCOURSE FOR BANKING INDUSTRY
Watson Assistant
Do you want to know about some other services ?
Yes
How can I help you ?
Offers
Offers
Please select a category
Dining
Shopping
BookMyShow
Travel
Type something...


International Business Machines
Built with IBM WATSON
AI BASED DISCOURSE FOR BANKING INDUSTRY
Watson Assistant
Please select a category
Travel
Upto 15% Instant Discount on Flights and Hotels
Offer duration : 1st Nov 2022 to 31st Dec 2022
Coupon code : IBM_YKSK
Do you want to know about some other services ?
Type something...


International Business Machines
Built with IBM WATSON
AI BASED DISCOURSE FOR BANKING INDUSTRY
Watson Assistant
Offers
Please select a category
Shopping
Flat 15% Instant Discount upto 100 INR on all Debit Card and Credit Card
Offer duration : 1st Nov 2022 to 31st Jan 2023
Coupon code : IBM_YKSK
Do you want to know about some other services ?
Type something...


International Business Machines
Built with IBM WATSON
AI BASED DISCOURSE FOR BANKING INDUSTRY
Watson Assistant
Yes
BookMyShow
20% discount upto INR 100/- using debit and credit cards on BookMyShow
Offer duration : 19th Nov to 19th Dec 2022
Coupon code : IBM_YKSK
Do you want to know about some other services ?
Type something...

8.TESTING

8.1 TEST CASES

	Date: 19-Nov-22 **Team ID: PNT2022TMID35942** **Project Name: Project - AI Based Discourse For Banking Industry** **Maximum Marks: 4 marks**
1	Verify user is able to clarify all his/her dounts regarding savings account account
2	Verify user is able to clarify all his/her doubts regarding net banking account
3	Verify user is able to clarify all his/her doubts regarding loans
4	Verify user is able to clarify all his/her doubts regarding current account
5	Verify user is able to clarify all his/her doubts general queries
6	Verify user is able to Login using valid account and password to check his account balance
7	Verify whether data provided by user in net banking registration form is stored in IBM Cloudant database
8	Verify whether data provided by user in feedback form is stored in IBM Cloudant database
9	Verify whether the system generates an alert message when the user provides invalid password or account number in Login page
10	Verify whether all the webpages provided are accessible to the users
11	Verify whether the chatbot provides accurate responses for keywords from the user side

				Date	19-Nov-22							
				Team ID	PNT2022TMID35942							
				Project Name	Project - AI Based Discourse For Banking Industry							
				Maximum Marks	4 marks							
Test case ID	**Feature Type**	**Component**	**Test Scenario**	**Pre-Requisite**	**Steps To Execute**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Comments**	**TC for Automation(Y/N)**	**BUG ID**	**Executed By**
LoginPage_TC_OO 1	Functional	Home Page	Verify user is able to see the about, contact, account balance, net banking and feedback form webpage links and access those webpages	Computer with Internet connection	1.Enter Home page URL and click enter 2.Click on Contact, about, net banking ,account balance, feedback links provided. 3.Verify whether those webpages are available	http://127.0.0.1:5000/ (local server url link)	User should be able to access all the webpages provided	Working as expected	Pass	Webpage functions as per requirements	Yes	Nil	Shwetha M K
LoginPage_TC_OO 2	Functional	Chatbot	Verify user is able to access the chatbot integrated with the webpage and check whether the chatbot is provided with the required actions and features	Computer with Internet connection	1.Enter Home page URL and press Enter 2.Click on chatbot icon provided at the left bottom of the webpage 3.Verify whether the chatbot has the following features: 1.Savings account action 2.Current account action 3. Net banking action 4. General queries action 5. Provides feedback link at the end of conversation 6. Provides link for login page to check current account balance 7. Provides link for net banking registration form	http://127.0.0.1:5000/ (local server url link)	Chatbot should have all the mentioned features	Working as expected	Pass	The application worked as per needs	Yes	Nil	Shwetha M K and Karthick V
LoginPage_TC_OO 3	UI	Net banking webpage	Verify the UI elements in net banking webpage	Computer with Internet connection	1.Enter net banking page URL and press enter 2.Click on net banking webpage link 2.Verify the following UI elements a.email text box b.customer name text box c.account number textbox d.contact number textbox e.account type textbox f.data of birth and data of application	http://127.0.0.1:5000/netbanking (local server url link)	All the UI elements should work properly and after submission of form, we should be directed back to the home page	Working as expected	Pass	UI elements worked as per needs	Yes	Nil	Karthick V
LoginPage_TC_OO 4	UI	Feedback form webpage	Verify the UI elements in feedback form webpage	Computer with Internet connection	1.Enter feedback form webpage URL and press enter 2.Verify the following UI elements a.email text box b.customer name text box c.place textbox d.contact number textbox e.feedback textbox	http://127.0.0.1:5000/feedback (local server url link)	All the UI elements should function as expected and after submission of form, we should be directed back to the home page	Working as expected	Pass	UI elements worked as per needs	Yes	Nil	Kavyapriya V
LoginPage_TC_OO 5	UI	Account balance webpage	Verify the UI elements in account balance webpage	Computer with Internet connection	1.Enter account balance webpage URL and press enter 2.Verify the following UI elements a. Account number textbox b. Password textbox c. Login button	http://127.0.0.1:5000/login (local server url link)	All the UI elements should should function as expected	Working as expected	Pass	UI elements worked as per needs	Yes	Nil	Kavyapriya V
LoginPage_TC_OO 6	Functional	IBM cloudant database	Verify whether user feedback data is stored in IBM cloudant database after submission	Computer with Internet connection	1.Go to Feedback form webpage 2.Enter the details required 3.Check whether the data entered is stored in database after submission	For feedback form Name: Kavyapriya V K E-mail Address: kavya123@gmail.com Contact Number: 9019201298 Place: Chennai Feedback: Good	Data entered by the users in feedback form and net banking registration form should be stored in the IBM cloudant database	Working as expected	Pass	Data storage system worked as per needs	Yes	Nil	Kavyapriya V

				Date	19-Nov-22								
				Team ID	PNT2022TMID35942								
				Project Name	Project - AI Based Discourse For Banking Industry								
				Maximum Marks	4 marks								
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO 7	Functional	IBM cloudant database	Verify whether user data in net banking registration form is stored in IBM cloudant database after submission	Computer with Internet connection	1.Go to Net banking webpage 2.Enter the details required 3.Check whether the data enteted is stored in database after submission	For net banking registration form Name of the customer: Yuvan Shankar J Email-address: yuvan123@gmail.com Contact number: 9013201212 Account number: 2019504609 Date of birth: 24-11-2001 Viewing rights: Yes Transaction rights: Yes Date of application: 20-11-2022	Data entered by the users in feedback form and net banking registration form should be stored in the IBM cloudant database	Working as expected	Pass	Data storage system worked as per needs	Yes	Nil	Yuvan Shankar J
LoginPage_TC_OO 8	Functional	Account balance webpage	Verify user is able to login using invalid account number and check his account balance	Computer with Internet connection	1.Enter account balance webpage URL and press enter 2.Enter invalid account number 3.Enter password 4. Click on Login button	Account number:2391019 password: Testing123	Webpage should show 'Invalid account number' alert message	Working as expected	Pass	Authentication system worked as per needs	Yes	Nil	Yuvan Shankar J
LoginPage_TC_OO 9	Functional	Account balance webpage	Verify user is able to login using invalid password and check his account balance	Computer with Internet connection	1.Enter account balance webpage URL and press enter 2.Enter invalid password 3.Enter password 4. Click on Login button	Account number: 2019504609 password: Testing123	Webpage should show 'Invalid password' alert message	Working as expected	Pass	Authentication system worked as per needs	Yes	Nil	Yuvan Shankar J
LoginPage_TC_OO 10	Functional	Account balance webpage	Verify user is able to login using valid account number and password	Computer with Internet connection	1.Enter account balance webpage URL and press enter 2.Enter valid account number and password 3.Enter password 4. Click on Login button	Account number: 2019504609 password: yuvanyuvan	webpage should display a greeting message with customer name and his account balance	Working as expected	Pass	Authentication system worked as per needs	Yes	Nil	Yuvan Shankar J

8.2 USER ACCEPTANCE TESTING

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	1	1	0	0	2
Duplicate	0	0	0	0	0
External	0	0	0	0	0
Fixed	1	1	0	0	0
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	1	1	0	0	2

This report shows the number of test cases that have passed, failed, and untested

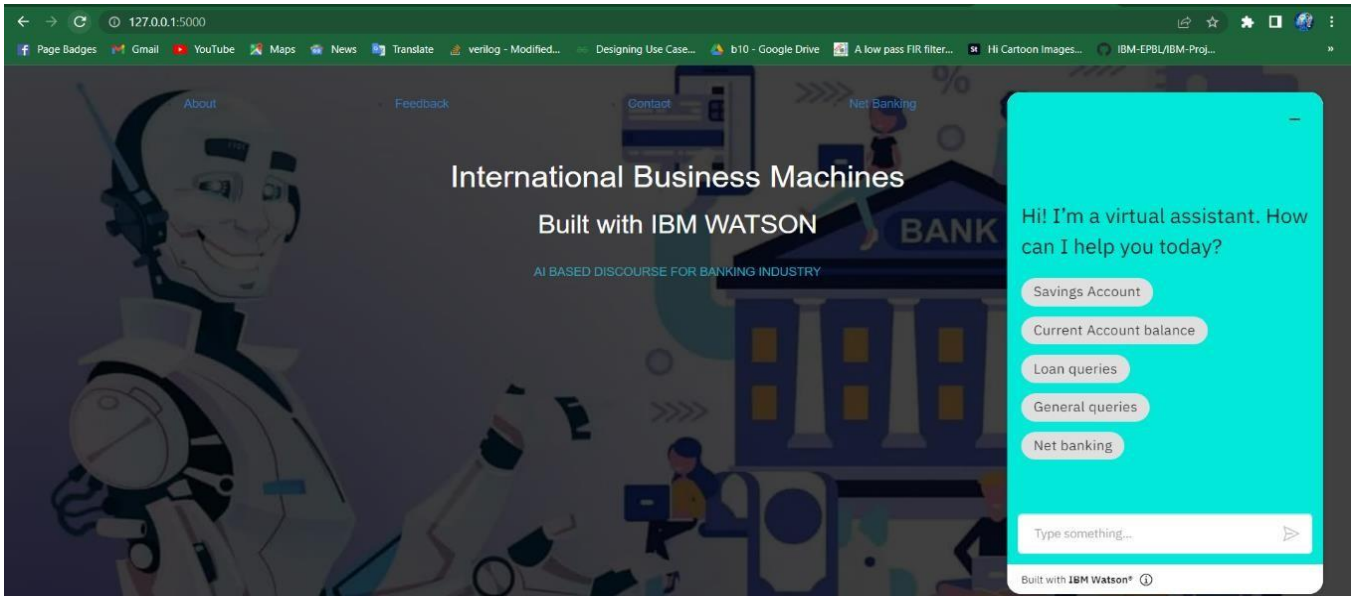
Section	Total Cases	Not Tested	Fail	Pass
User Interface	7	0	0	7
Chatbot response for different queries on banking	121	0	0	121
Security and Authentication	8	0	0	8
Data storage in IBM Cloudant Database	10	0	0	10
Alert Message Generation while Authentication	11	0	0	11
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. RESULTS

9.1 PERFORMANCE TESTING

Model Summary

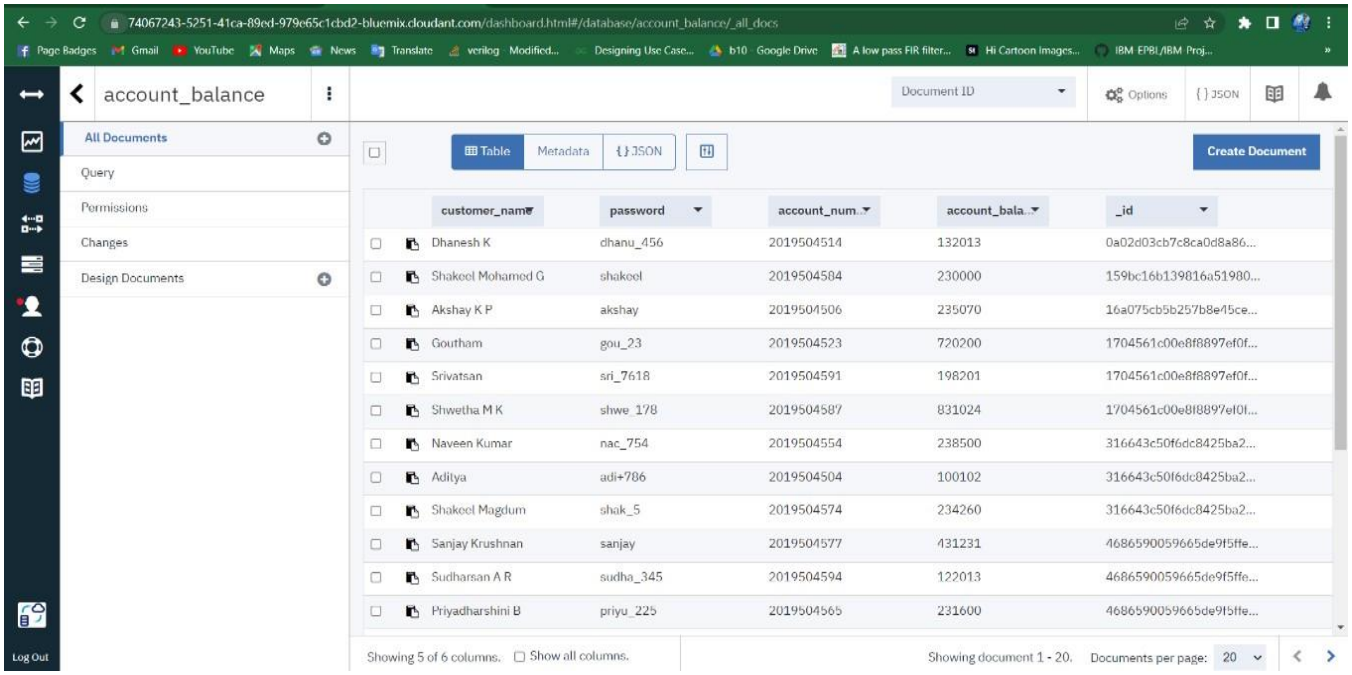
- **Actions provided in chatbot:**
- Savings account
- Current account
- Net banking
- General queries
- Loan queries
- Offers



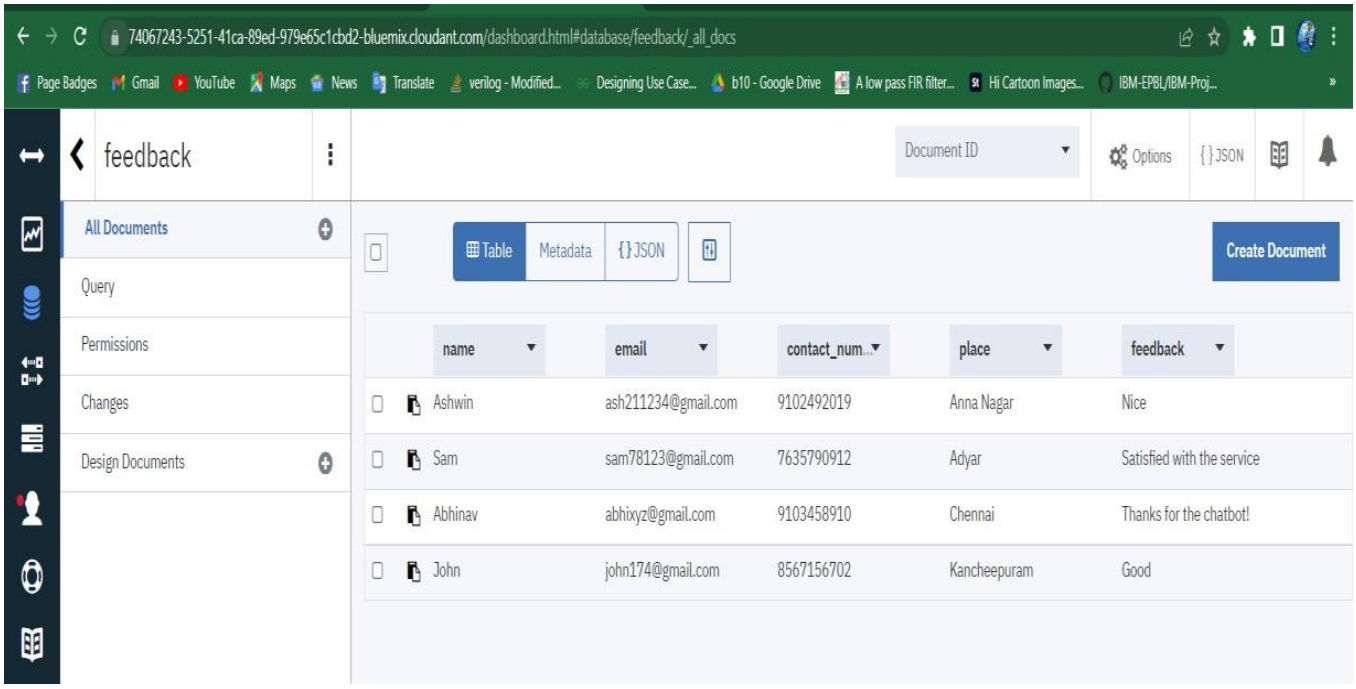
Unique features:

- Checking account balance after authentication.
- Storing user feedback in IBM cloudant database.
- Storing input data from net banking registration in IBM cloudant database

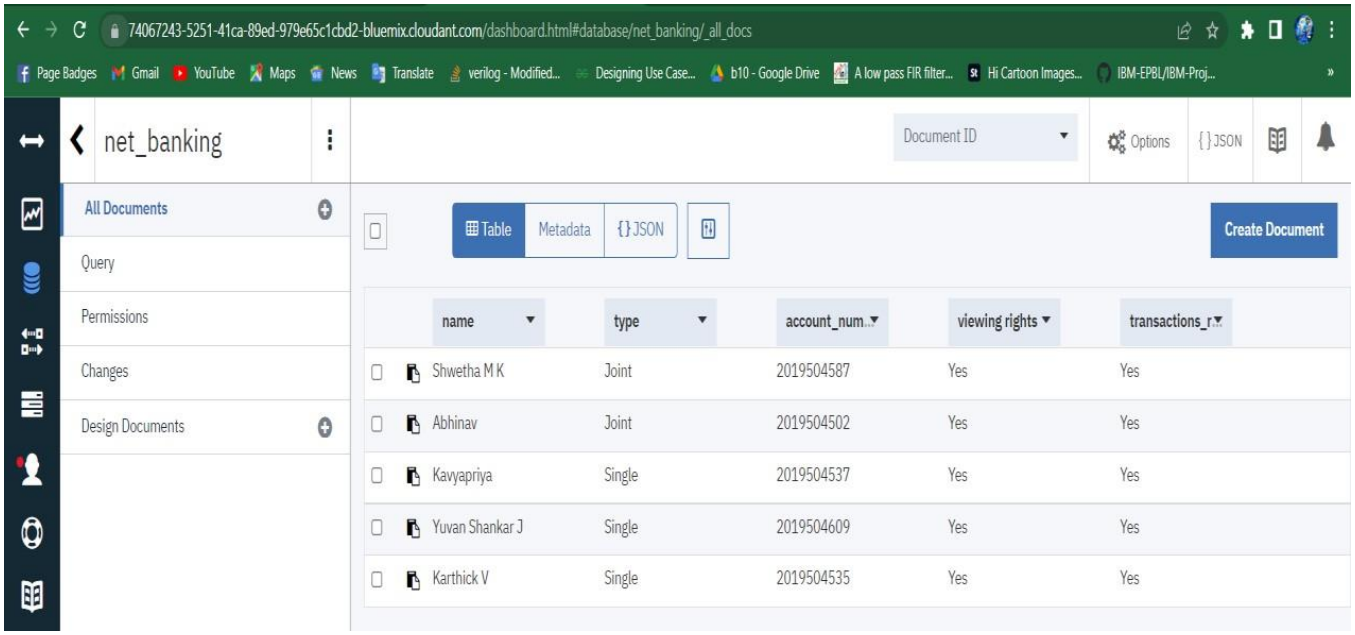
DATABASE TO STORE ACCOUNT DETAILS TO VALIDATE WHEN THE USER CHECKS HIS ACCOUNT BALANCE:



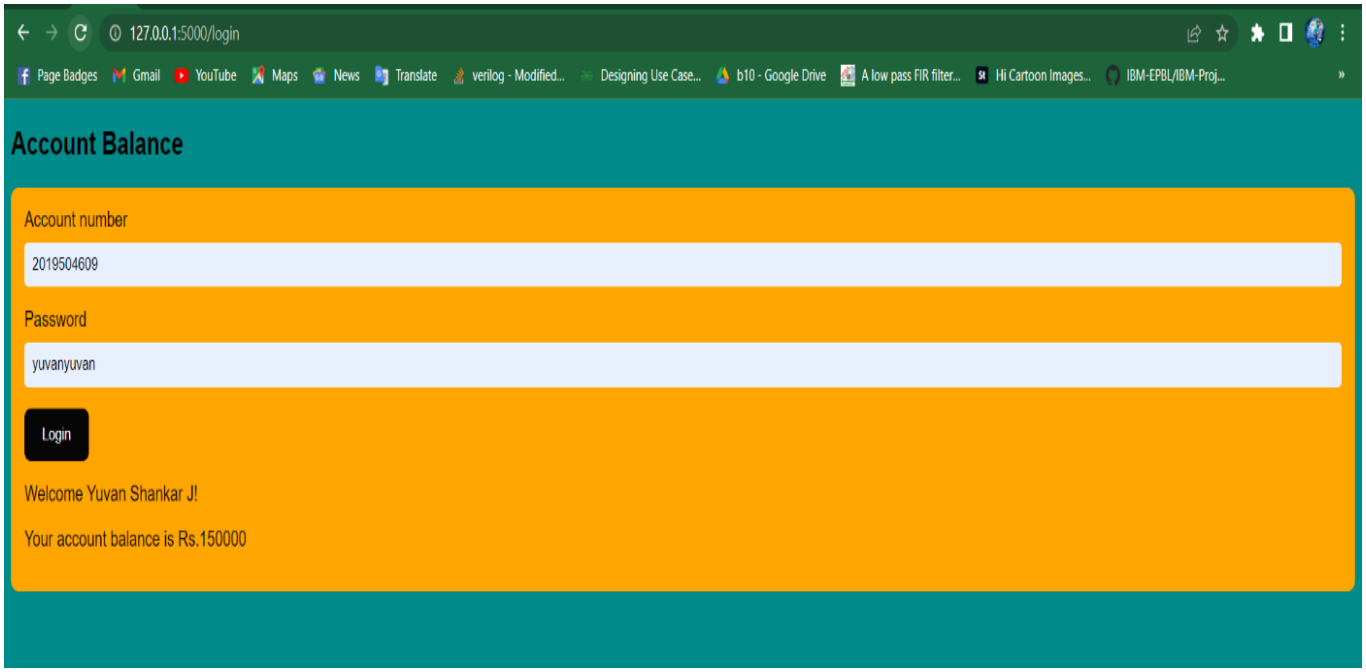
DATABASE TO STORE USER FEEDBACK:



DATABASE TO STORE USER DATA FROM NET BANKING REGISTRATION FORM:



CHECKING ACCOUNT BALANCE AFTER AUTHENTICATION:

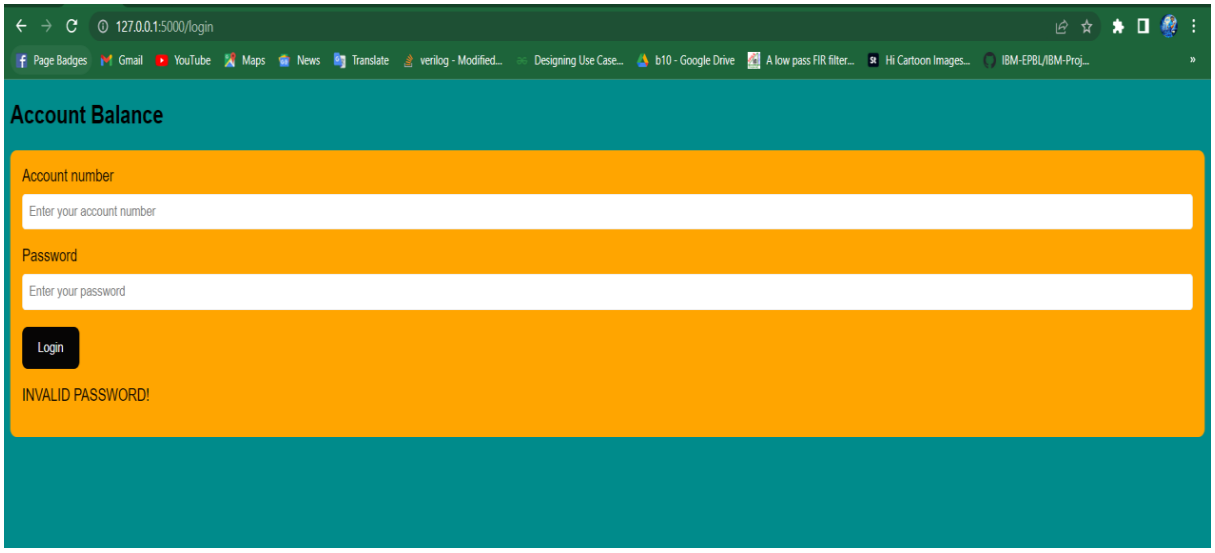


Accuracy

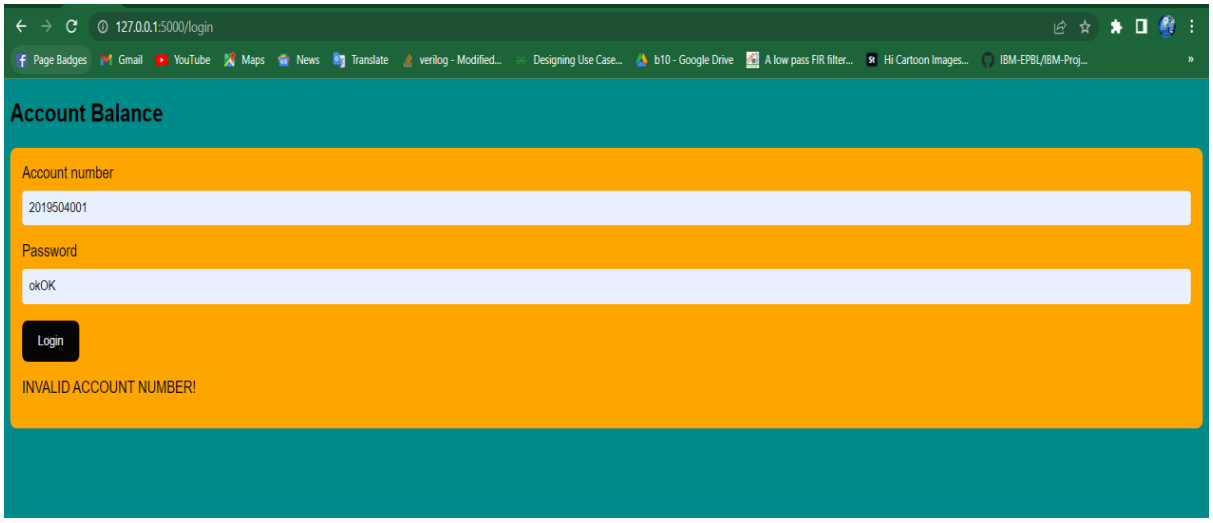
- Validation of user credentials using login isdone accurately by usingaccount details in the database.
- Instant and accurate response is provided by thechatbot for all queries.

VALIDATION OF USER CREDENTIALS:

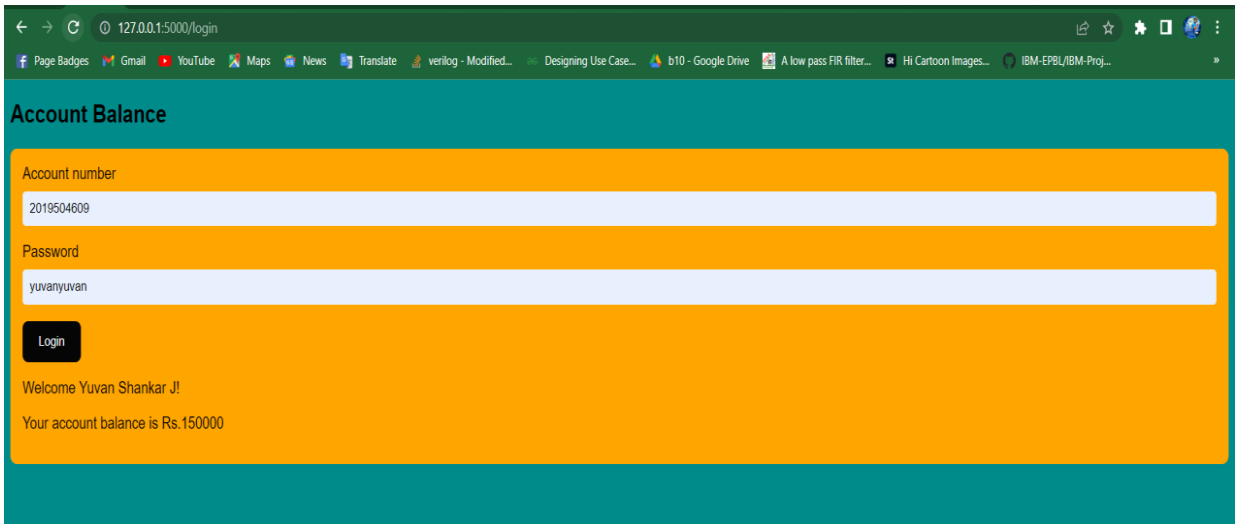
WHEN USER PROVIDES INVALID PASSWORD:



WHEN USER PROVIDES INVALID ACCOUNT NUMBER:



WHEN USER PROVIDES CORRECT ACCOUNT NUMBER AND PASSWORD:



10. ADVANTAGES AND DISADVANTAGES

10.1 ADVANTAGES

The need to stay available all the time is at the center of the ever-growing popularity of chatbots across industries. And if your business wants to engage customers round the clock and improve their experience, it must use a bot at some point in the time. This will help manage customer requests with instant responses and boost satisfaction levels. Chatbots improve customer engagement and reduce customer service cost. Chatbots ensure scalabilty of support. Chatbots enhance consistency in service and neglects manual errors. Manual intervention is reduced, therefore workload is decreased. Chatbots provide responses in an instantaneous manner. Therefore, there is no need for customers to waste their precious time waiting for response from the service providers. Bots can ensure a touch of personalization by engaging customers with one-on-one conversations, maintaining a natural-sounding tone, and by being good at interactive communication. Storing customer details provided during net banking registration and account creation forms in cloud could be useful for future reference. Feedback system implemented could help us to address technical issues faced by customers and rectify them.

10.2 DISADVANTAGES

Chatbots have limited responses, so they're not often able to answer multi-part questions or questions that require decisions. This often means your customers are left without a solution, and have to go through more steps to contact your support team. They sound too mechanical and can only give answers to problems that they have been programmed with. Chatbots cannot maintain a natural-sounding conversation in-depth with customers and that is why they are only useful in solving basic queries. But this can create a disconnect with customers who prefer the human approach when solving their problems. Chatbots require timely updates and more advanced Natural Language Processing capabilities are also developed with time. The proposed model cannot work without proper internet connection.

11. CONCLUSION

This project proposes an Artificial Intelligence based Chatbot with elegant user interface to support customer service in a computer-centred manner. This model is very helpful in cases where we require our customers require 24/7 connectivity. Banking is an important sector in the society for its economical well being. Concerning the proposed development, it can be concluded that the use of digital banking and artificial intelligence has a broad positive impact not only for the company but also for its users. chatbots and other types of AI assistants are of great use in any industry that has to provide high-quality customer support.Therefore, AI based Chatbots is a major requirement in financial industry to improve customer service. This model doesnot work without internet connecitivity. The Artificial Intelligence based Chatbot created in this project guides the customer through account creation, net banking queries, loan queries and general financial queries. A feedback system has been implemented using IBM Cloudant as the database to collect feedback from users and store it for future reference to improve service. Basic net banking registration is also provided with this model and the user information is used in IBM Cloudant database. This is a drawback as people without internet connection cannot access the service provided.

12. FUTURE SCOPE

The proposed model can be integrated with speech to text and text to speech services to improve ease of conversation. Natural Language Process techniques can be extended to support users conversing through different languages. This model can be provided to the users in future as an offline application to overcome the drawback of requirement of continous internet connection. More features and banking related actions can be added to the chatbot to improve efficienct and customer satisfaction. More and more banks tend to integrate chatbots into their mobile apps. This is a convenient way to stay in touch with their clients and, at the same time, reduce the involvement of human personnel. Banking bots are already evolving and the way they work will change fast too. Instead of getting a text message saying customers are overdrawn, a chatbot could pop-up and offer a series of immediate resolutions to the problem, helping the customer understand the ramifications of each. As a further development, chatbots will predict human behavior more accurately and use this information for self-learning.

13. APPENDIX

SOURCE CODE

Chatbot.html

```
<html>
  <head>
  <title>
    Banking Chatbot
  </title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <link rel="stylesheet" href="{{url_for('static',filename='style.css')}}">

</head>
<body>
  <script>
 window.watsonAssistantChatOptions = {
  integrationID: "6137f6ce-9292-4889-875b-977c474b9366", // The ID of this
integration.
  region: "us-south", // The region your integration is hosted in.
  serviceInstanceID: "5a1aaab3-a10a-4add-a628-e34f5a029880", // The ID of your
service instance.
  onLoad: function(instance) { instance.render(); }
 };
 setTimeout(function(){
  const t=document.createElement('script');
  t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
  document.head.appendChild(t);
 });
</script>



  <div class="banner">
    <div class="navbar">
        <li><a href="{{url_for('about')}}">About</a></li>
        <li><a href="{{url_for('feedback')}}">Feedback</a></li>
```

```
            <li><a href="{{url_for('contact')}}">Contact</a></li>
            <li><a href="{{url_for('netbanking')}}">Net Banking</a></li>
            <li><a href="{{url_for('login')}}">Account Balance</a></li>
          </ul>
        </div>

    <div>
      <h1 class="para">International Business Machines</h1>
      <h2 class="para">Built with IBM WATSON</h2>
      <P class="pages"> AI BASED DISCOURSE FOR BANKING INDUSTRY</P>
    </div>
    </div>

</body>
</html>
```

Feedback.html

```
<!DOCTYPE html>
<html>
    <title>Feedback Form</title>
    <style>
      body {
        font-family: Arial, Helvetica, sans-serif;
        background:Beige;
      }
      * {
        box-sizing: border-box;
      }
      input[type=text],
      select,
      textarea {
        width: 100%;
        padding: 8px;
        border: 1px solid #eeeeee;
        border-radius: 4px;
        box-sizing: border-box;
        margin-top: 8px;
        margin-bottom: 16px;
        resize: vertical;
      }
      input[type=submit] {
        background-color: #060505;
        color: #ffffff;
        padding: 12px 20px;
        border: none;
        border-radius: 7px;
        cursor: pointer;
```

```
    }
    input[type=submit]:hover {
      background-color: #7f81d2;
    }
    .container {
      border-radius: 8px;
      background-color: #7f81d2;
      padding: 15px;
    }
   </style>
  </head>
  <body >
   <h2>Feedback Form</h2>
   <div class="container">
    <form action="feedback" method="POST">
      <label for="fname">Name</label>
      <input type="text" id="name" name="name" placeholder="Enter your name">
      <label for="mail">E-mail Address</label>
      <input type="text" id="mail" name="mail" placeholder="Enter your e-mail">
      <label for="num">Contact Number</label>
      <input type="text" id="num" name="num" placeholder="Enter your contact
number">
      <label for="place">Place</label>
      <input type="text" id="place" name="place" placeholder="Enter your place">
      <label for="message">Feedback</label>
      <textarea id="message" name="message" placeholder="Provide your feedback"
style="height:200px"></textarea>
      <input type="submit" value="Submit" name="submit">
    </form>
   </div>
   <!--<body>
       <form class="needs-validation" novalidate>
    <div class="form-row">
      <div class="col-md-4 mb-3">
        <label for="validationCustom01">First name</label>
        <input type="text" class="form-control" id="validationCustom01"
placeholder="First name"  required>
        <div class="valid-feedback">
          Looks good!
        </div>
      </div>
    </div>
    <div class="form-group">
      <div class="form-check">
        <input class="form-check-input" type="checkbox" value="" id="invalidCheck"
required>
        <label class="form-check-label" for="invalidCheck">
          Agree to terms and conditions
```

```
        </label>
        <div class="invalid-feedback">
          You must agree before submitting.
        </div>
      </div>
    </div>
    <button class="btn btn-primary" type="submit">Submit form</button>
  </form>

  <script>
  // Example starter JavaScript for disabling form submissions if there are invalid fields
  (function() {
    'use strict';
    window.addEventListener('load', function() {
      // Fetch all the forms we want to apply custom Bootstrap validation styles to
      var forms = document.getElementsByClassName('needs-validation');
      // Loop over them and prevent submission
      var validation = Array.prototype.filter.call(forms, function(form) {
        form.addEventListener('submit', function(event) {
          if (form.checkValidity() === false) {
            event.preventDefault();
            event.stopPropagation();
          }
          form.classList.add('was-validated');
        }, false);
      });
    }, false);
  })();
  </script>-->
 </body>
</html>
```

Contact.html

```
<htmL>
  <style>

      .image{
         width: 100%;
         height: 70%;
      }
      p{
         padding: 10px;
      }
  </style>
  <body>
```

```
      <IMG SRC="https://www.silverbazel.com/wp-content/uploads/2021/04/Contact-us-banner-1.png" alt="bank image" class="image">
      <div class="para">
      <p> Contact us

<br>
1. Yuvan Shankar J   8925380524<br>
2. Kavyapriya V K    9566088902<br>
3. Shwetha M K       8903199113<br>
4. Karthick V        9150447509<br>
</div>
   </body>
</htmL>
```

Login.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Account Balance</title>
    <style>
      body {
        font-family: Arial, Helvetica, sans-serif;
        background-color:darkcyan;
      }
      .header {
      padding: 30px;
      text-align: center;
      background: #bc1a6b;
      color: white;
      font-size: 30px;
      }
      * {
        box-sizing: border-box;

      }
      input[type=text],
      select,
      textarea {
        width: 100%;
        padding: 8px;
        border: 1px solid #eeeeee;
        border-radius: 4px;
        box-sizing: border-box;
        margin-top: 8px;
        margin-bottom: 16px;
        resize: vertical;
```

```
    }
    input[type=text],
    select,
    textarea {
      width: 100%;
      padding: 8px;
      border: 1px solid #eeeeee;
      border-radius: 4px;
      box-sizing: border-box;
      margin-top: 8px;
      margin-bottom: 16px;
      resize: vertical;
    }
    input[type=submit] {
      background-color: #060505;
      color: #ffffff;
      padding: 12px 20px;
      border: none;
      border-radius: 7px;
      cursor: pointer;
    }
    input[type=submit]:hover {
      background-color: #7f81d2;
    }
    input[type=submit]:hover {
      background-color: #7f81d2;
    }
    .container {
      border-radius: 8px;
      background-color: #FFA500;
      padding: 15px;
    }
   </style>
  </head>
  <body>
   <h2>Account Balance</h2>
   <div class="container">
    <form action="login" method="POST">
     <label for="accnum">Account number</label>
     <input type="text" id="accnum" name="accnum" placeholder="Enter your account
number">
     <label for="pass">Password</label>
     <input type="text" id="pass" name="pass" placeholder="Enter your password">
     <br>
     <input type="submit" value="Login" name="submit">
     {% with messages = get_flashed_messages() %}
       {% if messages %}
          {% for message in messages %}
```

```
                <p>{{ message }}</p>
            {% endfor %}
          {% endif %}
        {% endwith %}
      </form>
    </div>
  </body>
</html>
```

Netbanking.html

```
<!DOCTYPE html>
<html>

  <head>
    <title>Title of the document</title>
    <style>
      body {
        font-family: Arial, Helvetica, sans-serif;
        background:peachpuff;
      }
      * {
        box-sizing: border-box;
      }
      input[type=text],
      select,
      textarea {
        width: 100%;
        padding: 8px;
        border: 1px solid #eeeeee;
        border-radius: 4px;
        box-sizing: border-box;
        margin-top: 8px;
        margin-bottom: 16px;
        resize: vertical;
      }
      input[type=text],
      select,
      textarea {
        width: 100%;
        padding: 8px;
        border: 1px solid #eeeeee;
        border-radius: 4px;
        box-sizing: border-box;
        margin-top: 8px;
        margin-bottom: 16px;
        resize: vertical;
```

```
    }
    input[type=submit] {
      background-color: #060505;
      color: #ffffff;
      padding: 12px 20px;
      border: none;
      border-radius: 7px;
      cursor: pointer;
    }
    input[type=submit]:hover {
      background-color: #7f81d2;
    }
    input[type=submit]:hover {
      background-color: #7f81d2;
    }
    .container {
      border-radius: 8px;
      background-color: #CD5C5C;
      padding: 15px;
    }
   </style>
  </head>
  <body>

   <h2>Net Banking Registration</h2>
   <div class="container">
    <form action="netbanking" method="POST">
      <label for="fname">Name of the customer</label>
      <input type="text" id="name" name="name" placeholder="Enter your name">
      <label for="mail">E-mail Address</label>
      <input type="text" id="mail" name="mail" placeholder="Enter your e-mail">
      <label for="number">Contact Number</label>
      <input type="text" id="number" name="number" placeholder="Enter your contact
number">
      <label for="num">Account Number</label>
      <input type="text" id="num" name="num" placeholder="Enter your account
number">
      <label for="dob">Date of Birth</label>
      <input type="date" id="dob" name="dob" placeholder="Date of birth">
      <label for="viewing">Viewing Rights (Yes/No)</label>
      <input type="viewing" id="viewing" name="viewing" placeholder="Enter Yes/No">
      <label for="transaction">Transaction Rights (Yes/No)</label>
      <input type="transaction" id="transaction" name="transaction"
placeholder="Enter Yes/No">
      <br>
      <br>
      <label for="type">Single/Joint Account</label>
      <input type="text" id="type" name="type" placeholder="Enter your Account Type">
```

```
        <br>
        <label for="doa">Date of Application</label>
        <input type="date" id="doa" name="doa" placeholder="Date of birth">
        <br>
        <br>
        <input type="submit" value="Submit" name="submit">
      </form>
    </div>
  </body>
</html>
```

Style.css

```
*{
    margin: 0;
    padding: 0;
    font-family: sans-serif;
}
.banner{
    width: 100%;
    height: 100vh;
    background-image: linear-gradient(rgba(0,0,0,0.75),rgba(0,0,0,0.75)),url('https://www.techrounder.com/wp-content/uploads/2021/07/ai-in-banking.jpg');
    background-size: cover;
    background-position: center;
}
.navbar{
    width: 100%;
    margin: auto;
    padding: 35px 0;
    display: flex;
    align-items: center;
    justify-content: space-between;
}
.logo{
    width: 400px;
    cursor: pointer;

}
.navbar ul li{
    list-style: none;
    display: inline-block;
    margin: 0 20px;
    position: relative;


}
```

```
.navbar ul li a{
    text-decoration: none;
    color: #fff;
    text-transform: uppercase;


}
.navbar ul li::after{
    content: '';
    height: 30px;
    width: 0;
    background: #009688;
    position: absolute;
    left: 0;
    bottom: -10px;
    transition: 0.5s;
}
.navbar ul li:hover::after{
    width: 100%;
}
.para{
    color: white;
    text-align: center;
}
.pages{
    text-align: center;
    color: rgb(50, 160, 179);
    padding: 20px;
}
button{
    width:200px;
    padding: 15px;
    text-align: center;
    margin: 20px 10px;
    border-radius: 25px;
    font-weight: bold;
    border: 2px solid #009688;
    background: transparent;
    color: #fff;
    cursor: pointer;
    position: relative;
    overflow: hidden;

}

.content{
    width: 100%;
    position: absolute;
```

```
    top: 50%;
    transform: translate(-50%);
    text-align: center;
    color: #fff;
}
.content h1{
    font-size: 120px;
    margin-top: 100px;
}
.content p{
    margin: 20px auto;
    font-weight: 25px;
```

app_route.py

```
"""
"""

from flask import Flask,render_template
from flask import *
from cloudant.client import Cloudant
from cloudant.result import Result
app=Flask(__name__)
ACCOUNT_NAME = "74067243-5251-41ca-89ed-979e65c1cbd2-bluemix"
API_KEY = "IqWgY3pe1n9DGN4pN_9uSXzmlCs27ML4DkcTAePwkFbv"

client = Cloudant.iam(ACCOUNT_NAME,API_KEY,connect=True)
mydatabase = client.create_database('feedback')
if mydatabase.exists():
    print(" successfully created.\n")

@app.route('/')
def Chatbot():
    return render_template('Chatbot.html')

@app.route('/home')
def home():
    return render_template('Chatbot.html')
@app.route('/about')
def about():
    return render_template('about.html')
@app.route('/feedback',methods = ['GET','POST'])
def feedback():
    print("******")
    if request.method == "POST":
        email = request.form['mail']
        name = request.form['name']
        num = request.form['num']
```

```
        place= request.form['place']
        msg = request.form['message']
        #print(email,name,num,place,msg)
        jsonDocument= {
            'email':email,
            'name':name,
            'contact_number':num,
            'place':place,
            'feedback':msg
            }
        newDocument = mydatabase.create_document(jsonDocument)
        result = Result(mydatabase.all_docs,include_docs=True)
        print(result[0])
        return redirect(url_for('home'))
        print('#####')
    return render_template('feedback.html')
@app.route('/contact')
def contact():
    return render_template('contact.html')
######
mydatabase1 = client.create_database('net_banking')
if mydatabase1.exists():
    print(" successfully created.\n")
@app.route('/netbanking',methods = ['GET','POST'])
def netbanking():
    print("******")
    if request.method == "POST":
        email = request.form['mail']
        name = request.form['name']
        acc_num = request.form['num']
        contact_number=request.form['number']
        date_of_birth=request.form['dob']
        date_of_application=request.form['doa']
        viewing_rights=request.form['viewing']
        transaction_rights=request.form['transaction']
        type_of_account=request.form['type']
        jsonDocument= {
            'email':email,
            'name':name,
            'account_number':acc_num,
            'contact_numer':contact_number,
            'viewing rights':viewing_rights,
            'transactions_rights':transaction_rights,
            'date_of_birth':date_of_birth,
            'date_of_application':date_of_application,
            'type':type_of_account
            }
        newDocument = mydatabase1.create_document(jsonDocument)
```

```
        result = Result(mydatabase1.all_docs,include_docs=True)
        print(result[0])
        return redirect(url_for('home'))
        print('#####')
    return render_template('netbanking.html')
mydatabase2=client.create_database('account_balance')
@app.route('/login',methods=['GET','POST'])
def login():
    if request.method=='POST':
        account_number=request.form['accnum']
        password=request.form['pass']
        account_found=False
        for documents in mydatabase2:
            if documents['account_number']==account_number:
                if documents['password']==password:
                    flash_name='Welcome'+' '+documents['customer_name']+'!'
                    flash(flash_name)
                    flash_text='Your account balance is'+'
'+"Rs."+documents['account_balance']
                    flash(flash_text)
                else:
                    flash('INVALID PASSWORD!')
                account_found=True
                break
        if not account_found:
            flash('INVALID ACCOUNT NUMBER!')
    return render_template('login.html')
if __name__ =='__main__':
    app.secret_key = 'super secret key'
    app.run(debug=True)
```

GITHUB LINK

https://github.com/IBM-EPBL/IBM-Project-497-1658304212

PROJECT DEMO LINK

https://youtu.be/DQycpImFjJk