A

PROJECT REPORT

ON

QuickConnect

IN PARTIAL FULFILMENT OF

MASTER'S DEGREE IN COMPUTER APPLICATIONS (M.C.A.)

GOVERNMENT MCA COLLEGE

MANINAGAR, AHMEDABAD

(Affiliated to the Gujarat Technological University)

SUBMITTED BY: -

ENROLLMENT NO	STUDENT NAME
[215690694057]	**Viraj Sanghani**
[215690694028]	**Rahul Panchal**
[215690694008]	**Masira Mansuri**

GOVERNMENT MCA COLLEGE

MANINAGAR, AHMEDABAD

(Affiliated to the Gujarat Technological University)

CERTIFICATE

This is to certify that System Development Project of "**QuickConnect"** developed and submitted to Gujarat Technological University by **Viraj Sanghani (215690694057)** for partial fulfilment of the requirement of MCA **Software Project-2 (639404)** in **SEM-3** in the year 2022-2023.

This is the original work and carried out under guidance and supervision. We further certify that to the best of our knowledge and belief the matter presented in this project report is Bonafide certificate.

Date of Submission:

Prof. Nirali R Sheth	**Prof B.B. Prajapati**	**Dr. Chetan B. Bhatt**
Assistant Professor	Head of Department	Principal
Government MCA College, Maninagar	Government MCA College, Maninagar	Government MCA College ,Maninagar

GOVERNMENT MCA COLLEGE

MANINAGAR, AHMEDABAD

(Affiliated to the Gujarat Technological University)

CERTIFICATE

This is to certify that System Development Project of "**QuickConnect"** developed and submitted to Gujarat Technological University by **Rahul Panchal (215690694028)** for partial fulfilment of the requirement of MCA **Software Project-2 (639404)** in **SEM-3** in the year 2022-2023.

This is the original work and carried out under guidance and supervision. We further certify that to the best of our knowledge and belief the matter presented in this project report is Bonafide certificate.

Date of Submission:

Prof. Nirali R Sheth	**Prof B.B. Prajapati**	**Dr. Chetan B. Bhatt**
Assistant Professor	Head of Department	Principal
Government MCA College, Maninagar	Government MCA College, Maninagar	Government MCA College ,Maninagar

GOVERNMENT MCA COLLEGE

MANINAGAR, AHMEDABAD

(Affiliated to the Gujarat Technological University)

CERTIFICATE

This is to certify that System Development Project of "**QuickConnect"** developed and submitted to Gujarat Technological University by **Masira Mansuri (215690694008)** for partial fulfilment of the requirement of MCA **Software Project-2 (639404)** in **SEM-3** in the year 2022-2023.

This is the original work and carried out under guidance and supervision. We further certify that to the best of our knowledge and belief the matter presented in this project report is Bonafide certificate.

Date of Submission:

Prof. Nirali R Sheth	**Prof B.B. Prajapati**	**Dr. Chetan B. Bhatt**
Assistant Professor	Head of Department	Principal
Government MCA	Government MCA	Government MCA
College, Maninagar	College, Maninagar	College ,Maninaga

Acknowledgement

I would like to take this opportunity to thank “Government MCA College” and University “Gujarat Technological University” for giving us this opportunity to work for this project.

Any project is never an individual effort in isolation rather it is a collective effort in a cooperative environment which is supported by direct and indirect contribution from people, so I would like to thank everyone who has helped me in the accomplishment of our project and I will be grateful to them for providing such a wonderful assistance and guidance towards our project.

I express my gratitude to all those who gave me the opportunity to complete this project. No volume of words is enough to express my gratitude towards my guidance or external guide.

We are grateful to “Government MCA College” under whose direction we were able to complete our project successfully, we take an opportunity to show our appreciation to the faculty members who invested their precious time and inspirational guiding us whenever needed.

I am highly indebted to Prof. Nirali R Sheth for her guidance and supervision as well as for providing necessary information regarding the project.

My thanks and appreciations also go to my friends who helped me out with their abilities.

Panchal Rahul (215690694028)

Viraj Sanghani (215690694057)

Masira Mansuri (215690694008)

TABLE OF CONTENTS

1. Introduction

1.1 Existing System

Chatbot visual builder empowers you to create perfect chatbots quickly and with less coding. Upload conversational elements, and test them in real time to design engaging chatbot stories.

Adding chatbot assistants reduces overhead costs, uses support staff time better and enables organizations to provide customer service during hours when live agents aren't available.

Our support team will help you with chatbot implementation and customization all along the line.

A chatbot is often described as one of the most advanced and promising expressions of interaction between humans and machines. These digital assistants streamline interactions between people and services, enhancing customer experience. At the same time, they offer companies new opportunities to streamline the customer's engagement process for efficiency that can reduce traditional support costs.

Instead of waiting on hold, customers can get answers to their questions in real time. Less service friction can improve the brand experience for customers.

Customers still value the ability to interact with live agents, particularly for more complex queries. Thus, keeping a human in the loop remains essential to the overall chatbot equation

1.2 Need for the New System

Organizations looking to increase sales or service productivity may adopt chatbots for time savings and efficiency, as chatbots can converse with users and answer recurring questions.

As consumers move away from traditional forms of communication, many experts expect chat-based communication methods to rise. Organizations increasingly use chatbot-based virtual assistants to handle simple tasks, allowing human agents to focus on other responsibilities.

1.3 Objective of the New System

- Reduce customer wait times and provide immediate answers;
- Offer customers 24/7 support;
- Remove the potential for unpleasant human-to-human interactions that moods and emotions of both the service or sales representative and the customer dictate;
- Reduce wait times and streamline conversations to minimize the potential for customers' stress and annoyance;
- Improve the redirection of customer queries;
- Add customized elements to the chatbot to advance brand personality.

1.4 Problem Definition

- Difficult workflows and less productivity
- Customer problems solution difficult and slower
- Too much overhead to grow business
- No facilities available for generate and qualify leads automatically
- Less customer experience
- Low conversion rates
- Not understandable questions for customers
- Difficult to connect chatbot with your favourite sites

1.5 Core Components

MERN Stack

MERN Stack is a compilation of four different technologies that work together to develop dynamic web apps and websites.

It is a contraction for four different technologies as mentioned below:

- M - MongoDB
- E - ExpressJS
- R - ReactJS
- N - NodeJS

MERN Stack Components

There are four components of the MERN stack. Let's discuss each of them one by one.

- The first component is MongoDB, which is a NoSQL database management system.
- The second MERN stack component is ExpressJS. It is a backend web application framework for NodeJS.
- The third component is ReactJS, a JavaScript library for developing UIs based on UI components.
- The final component of the MERN stack is NodeJS. It is a JS runtime environment, i.e., it enables running JavaScript code outside the browser.

Let’s learn more about these MERN Stack components:

MongoDB

MongoDB is a NoSQL DBMS where data is stored in the form of documents having key-value pairs similar to JSON objects. MongoDB enables users to create databases, schemas, and tables. It offers the Mongo shell that provides a JS interface for deleting, querying, and updating the records.

ExpressJS

ExpressJS is a NodeJS framework that simplifies writing the backend code. It saves you from creating multiple Node modules. For keeping the code precise, ExpressJS offers a range of middleware.

ReactJS

ReactJS is a JS library that allows the development of user interfaces for mobile apps and SPAs. It allows you to code JavaScript and develop UI components. The JS library uses virtual DOM for doing everything.

NodeJS

NodeJS is an open-source JavaScript runtime environment that allows users to run code on the server. It comes with the node package manager or npm, enabling users to select from a wide selection of node modules or packages. Being developed on the Chrome JavaScript Engine enables Node to execute code faster.

1.6 Project Profile

An all-in-one platform to build and launch conversational chatbots with less coding.

A chatbot is software that simulates human-like conversations with users via text messages on chat. Its key task is to help users by providing answers to their questions.

No matter whether you're a growing company or a market leader, chatbot helps you communicate better with customers and push your business forward.

Users in both business-to-consumer (B2C) and business-to-business (B2B) environments can increasingly use chatbot virtual assistants to handle simple tasks.

Adding chatbot assistants reduces overhead costs, uses support staff time better and enables organizations to provide customer service during hours when live agents aren't available.

It will benefit to;

- POWER UP YOUR MARKETING

Personalize your customer experience.

- SUPERCHARGE YOUR SALES

Scale your efforts without increasing employee headcount.

- OPTIMIZE YOUR SUPPORT

Keep your customer support going after hours.

Chatbot will also provide you live chat feature that will be directly connect user to the agent and will be helpful in doubt solving.

1.7 Assumptions and Constraints

1. Chatbots Can Enhance Sales

It is seen that the customer experience gets even better when they have an assisting agent with them. Your chatbot can assist the clients in shopping while resolving their queries on the go.

Also, any purchase chances are higher when the customer is getting better services throughout the shopping journey

2. Chatbots Can Manage Accounts

Managing accounts requires one to be precise about the data and the information. And you can use a chatbot to manage accounts and retrieve details whenever needed.

These artificially intelligent bots are more precise than humans, and there is a small scope of mistakes when you have the right chatbots deployed.

3. Chatbots are Customer Executives

Most of the websites render online chat options due to their ease and convenience. Chatbots are perfect for deploying as customer executives because they are available around the clock and seamlessly resolve customer issues.

Chatbots have more efficiency in dealing with clients when compared to human agents.

The advantages of chatbots are never-ending. They are replacing humans in many fields and are proving them as a better option.

Now you can place food orders, complain about the quality of food, and claim a refund without any human intervention as a chatbot will resolve all the queries with its smart sense.

Some Undeniable Limitations of Chatbots

Chatbots are directly linked with businesses, so understanding their weaknesses is a crucial part. There are a plethora of limitations users and business owners have complained about them. Also, these limitations of chatbots have stopped various organizations from deploying chatbots on their applications and websites.

1. Chatbots Don't Understand Human Context.

It is one of the significant limitations of chatbots. These chatbots are programmed in a way that they only know what they are taught. They cannot understand humans' context, and this is a massive gap that can even lead to an angry customer.

The AI-powered smart-bots can understand the general context, but 40 out of 100 cases are not related to the broad context.

2. They Don't Do Customer Retention.

Retaining a customer is a vital part of every organization. It holds more importance than getting new customers. A chatbot is significantly less capable of retaining the customers as it only tries up to a level for which it is programmed.

It is seen that human executives are better at customer retention because they can relate to the customers' feelings, which is not the case with chatbots.

3. They Can't Make Decisions.

Another limitation of chatbots is that they lack decision-making. They don't have the right know-how to differentiate between the good and the bad.

On March 23, 2016, the tech biggie Microsoft attracted many controversies due to its chatbot Tay. The chatbot posted offensive Tweets and landed Microsoft in huge troubles. So they have to shut down the chatbot temporarily.

Similarly, chatbots have done a lot of damage to multiple brands due to their poor decision-making capability.

4. Exorbitant Installation

Yes, chatbots save you a lot of money in the long run, but their installation cost can break the bank. You need to hire professionals who have rightly programmed chatbots to match the integrity of your business.

And installing a chatbot service means your business should be ready for substantial investment into Artificial Intelligence and Machine Learning.

5. Chatbots Have the Same Answer For a Query

Most customers don't proceed with the chat when they know they are chatting with a chatbot. Chatbots are easily identifiable because they have the same answer for multiple queries. Suppose you are asking something to a bot that is not available in the data server so that you will get an apology.

The same is the case with other queries; no matter how many different questions you ask, it will deliver you with the same apology, which is quite irritating.

1.8 Advantages and Limitations of the System

1. They help you get to know your customers

Conversational chatbots can help you get to know your customers even better. They allow you to find out what their most common questions and needs are, as well as the products or services that interest them. They help you make strategic decisions to improve the experience and offer a more personalized service with each interaction.

2. They are a sales machine

Your chatbot can offer immediate assistance to your potential customers and help speed up the buying decision process.

3. Customer service around the clock

One of the biggest advantages is that chatbots are available 24 hours a day to help customers. In addition, they respond quickly to all the questions they receive. This guarantees two very important things for almost all companies:

- Satisfied customers with immediate responses and resolutions
- Human agents focused on more important tasks and without work overload due to excessive consultations during non-working hours

4. They help to optimize costs

Implementing a Chatbot with conversational AI is a great way to automate customer service and improve the service provided by agents, which also leads to cost optimization in the medium term.

5. They offer a personalized experience

All people are very different and their way of communicating too. Therefore, the important thing is to offer an adequate and empathetic response to each query.

Some things you should remember:

Chatbot is not a human agent.

A conversational Chatbot is not the same as a human agent, so it does not always understand a query. Its selection of answers may be limited, depending on the information it has uploaded. There may be times when your interactions seem "robotic".

Need maintenance time to time.

The chatbot needs to be fed new and meaningful data (i.e. content) that can answer customer questions and queries.

Deploying, configuring, and learning of the chatbot can take a while.

2. Requirement Determination & Analysis

2.1 Requirement Determination

- Requirement Analysis is the first and important step in the Software development activity for building robust and user-friendly applications.
- I have started working on determining the functionalities that the application should provide.
- It is important to select highly able and new emerging technology for building any applications
- I have done a good amount of research on existing systems and the disadvantages of those. Once the functional requirements are finalized, I did research on the current technologies that are widely used in the industry and decided to use MongoDB, ExpressJS, ReactJS, NodeJS and Socket.io
- Because of we are selected socket.io to create our live chat, it is important to select those programming language which are compatible with socket.io library and easy to implementing.

2.2 Targeted Users

Our targeted users are ecommerce companies, portfolio site owner, product based companies who create new product and want to reach more customers.

It is essential that you invest enough time in the preparations before the actual creation of your own chatbot and think about the desired target group of your chatbot.

Once the chatbot is live and you realize that it does not meet the requirements and expectations of the desired users, it is too late will cost the company dearly.

In a first step, you must always determine the use case, i.e. the function that the chatbot should fulfil. This can also be several.

Secondly, think about the target group that the chatbot should address with the selected use cases. The best way is to define so-called «personas». The term persona is already well known in the field of customer care and comes from Customer Experience Management (CEM).

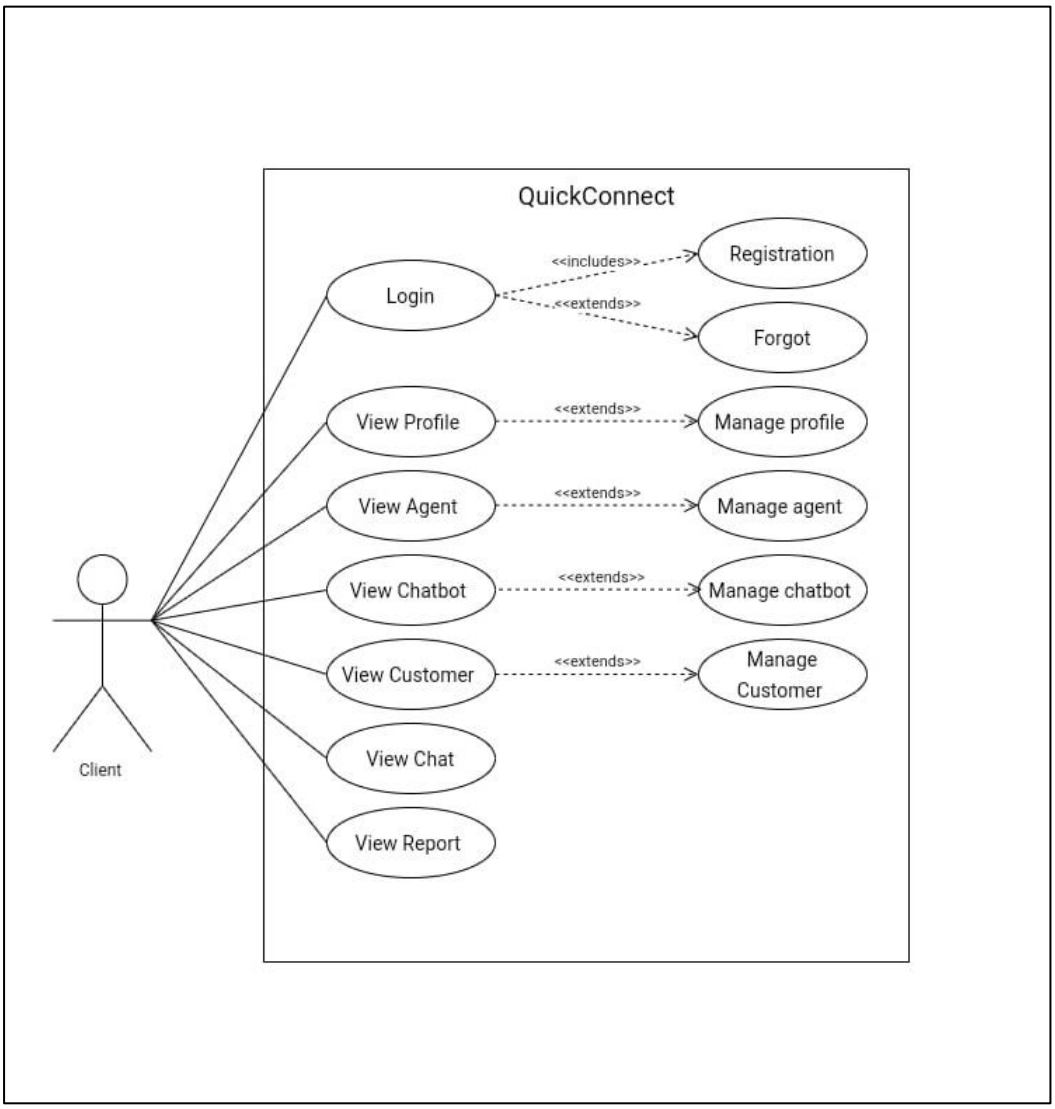
A persona characterizes the people of a certain target group and shows their characteristics. A detailed description with name, appearance, function, career and private life (hobbies), behaviours, preferences and expectations of the users allows you to better put yourself in the shoes of your potential users and develop the chatbot from the corresponding user perspective.

Instead of personas, companies often only determine the customer segments and thus remain very much on the surface. This is often not enough to align the chatbot with the expectations of the desired user group. If you want to go more in depth, it is almost impossible to do without the development of personas.

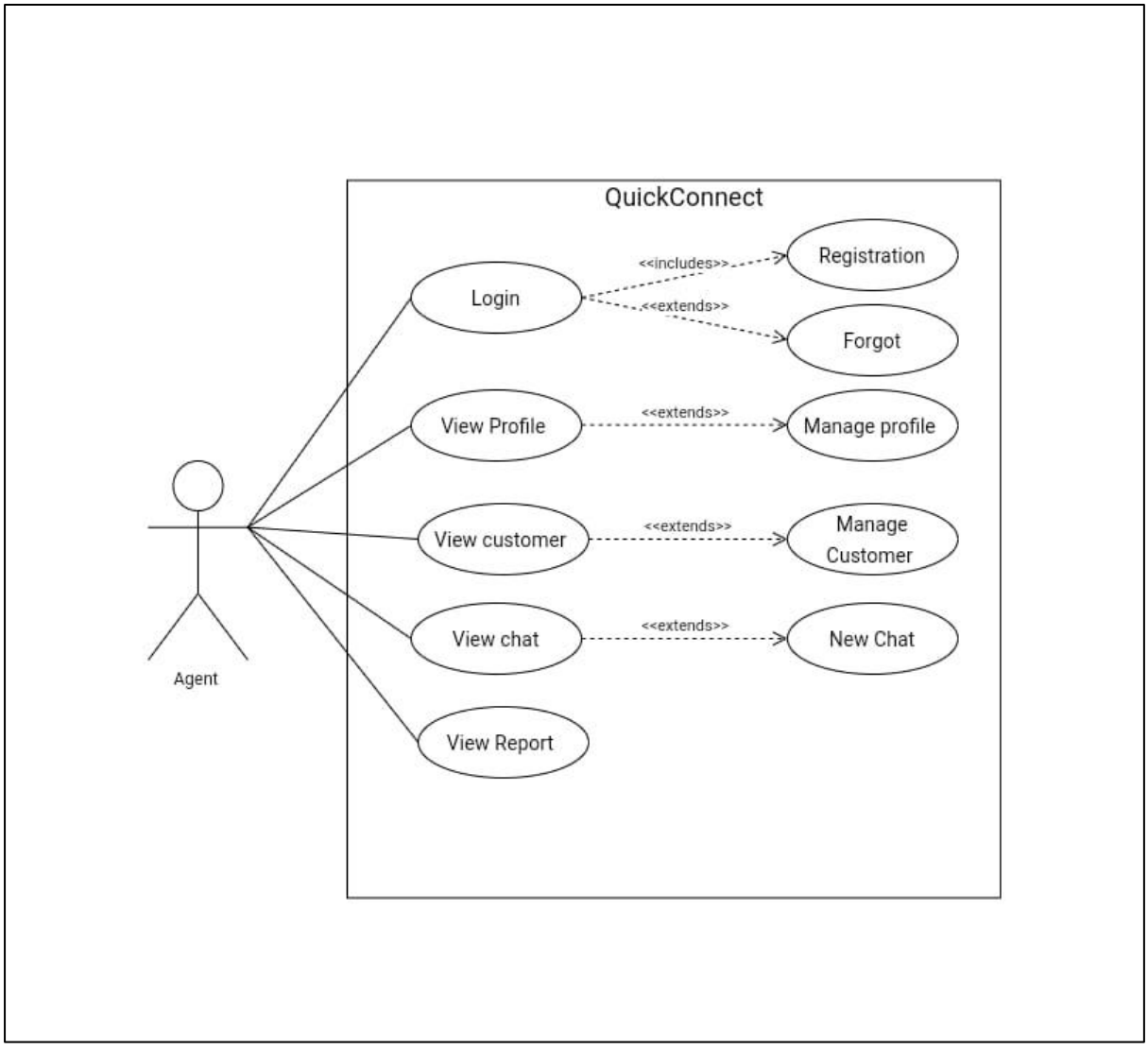
3.System Design

3.1 Usecase Diagrams

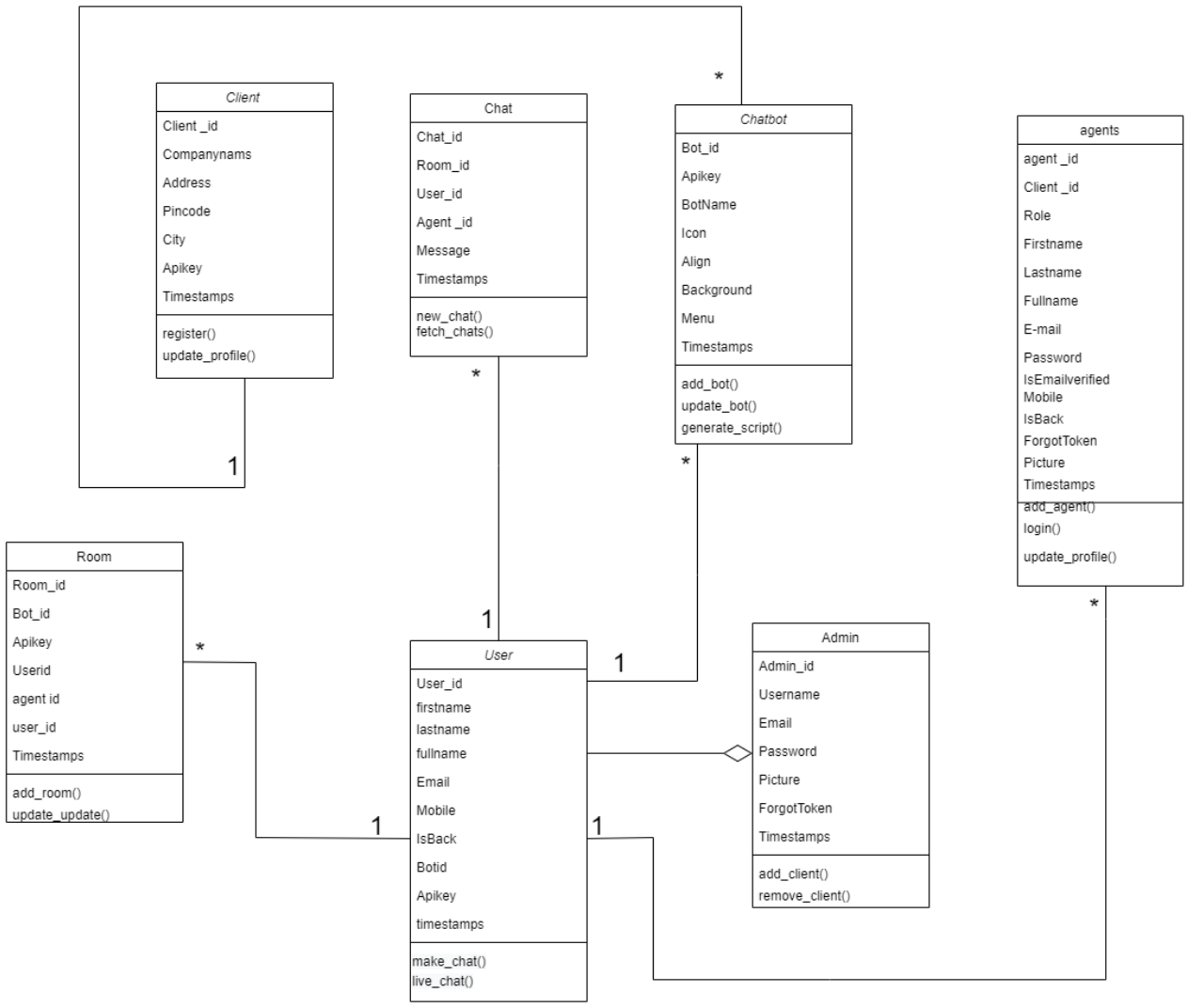
- **Usecase diagram for Client :**



- **Usecase diagram for Agent :**

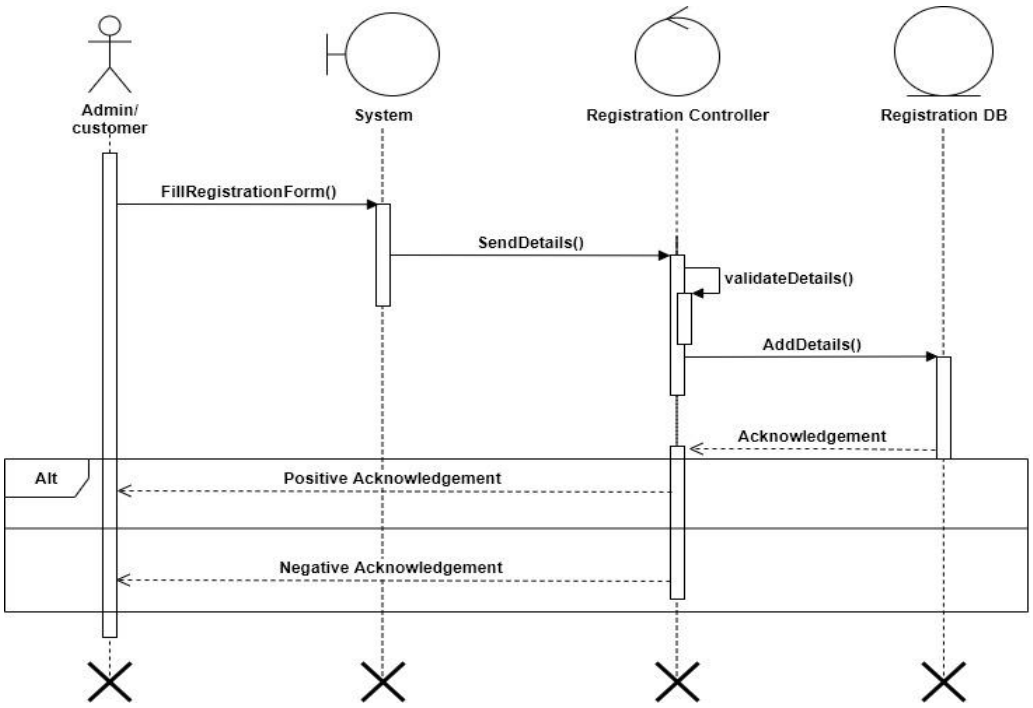


3.2 Class Diagram



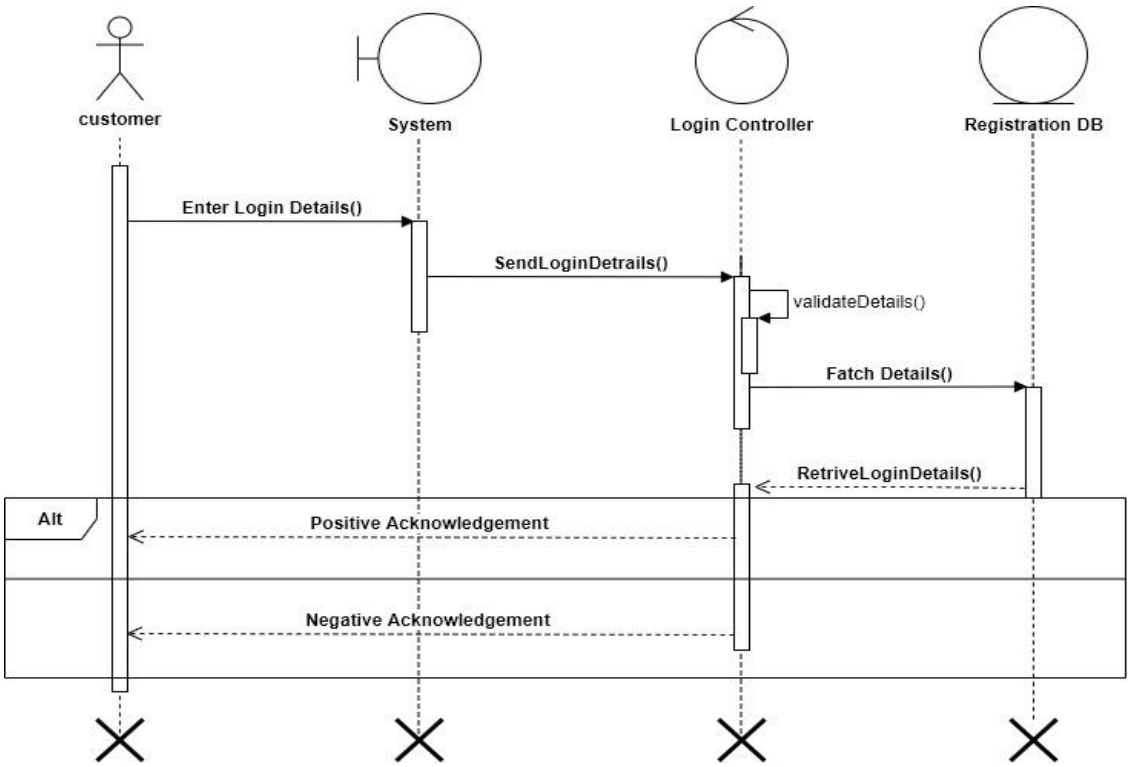
3.3 Interaction Diagram

- **Sequence Diagrams for Registration :**



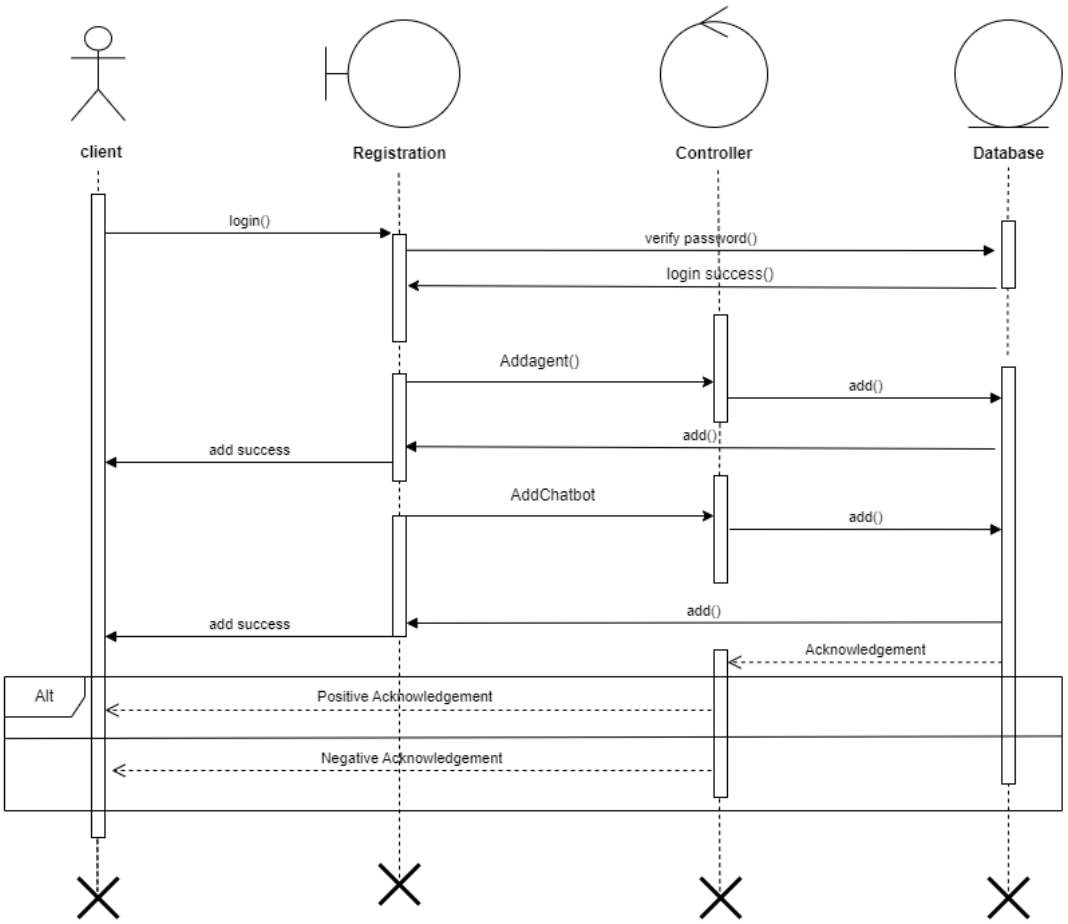
Admin/Customer can fill Registration form then it will system validation and send details Registration Controller then it store Database then it will send acknowledgement Registration Details then Registration successfully

- **Sequence Diagrams for Login:-**



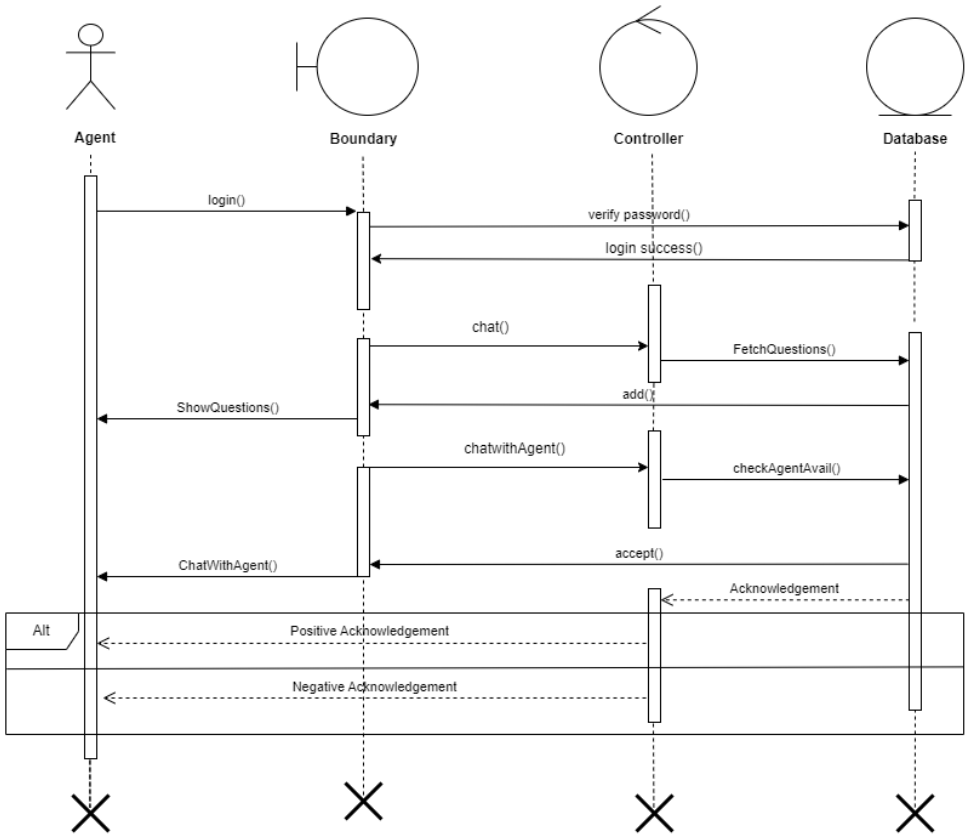
Customer can Login details like username and password then it will system validation and send details Login Controller its validate then it fetch Database then it will send acknowledgement then Login successfully

- **Sequence Diagrams for Client:-**



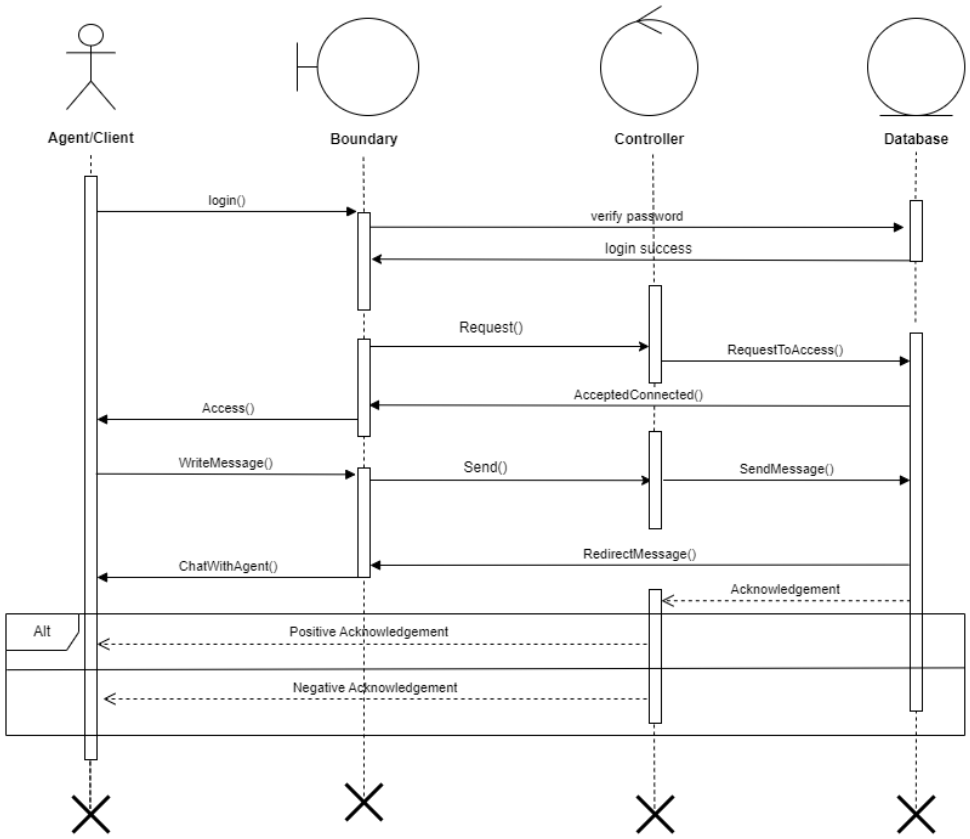
Client can Login details like username and password then it will system validation and send details Login Controller its validate then it fetch Database then it will send acknowledgement then Login successfully. Then Client can Add agent then it will store the database then it will send message to client add successfully. Client also do that add chatbot details like color,position of chatbot ,name agent then it will store the database then it will send message to client add successfully.

- **Sequence Diagrams for Agent :-**



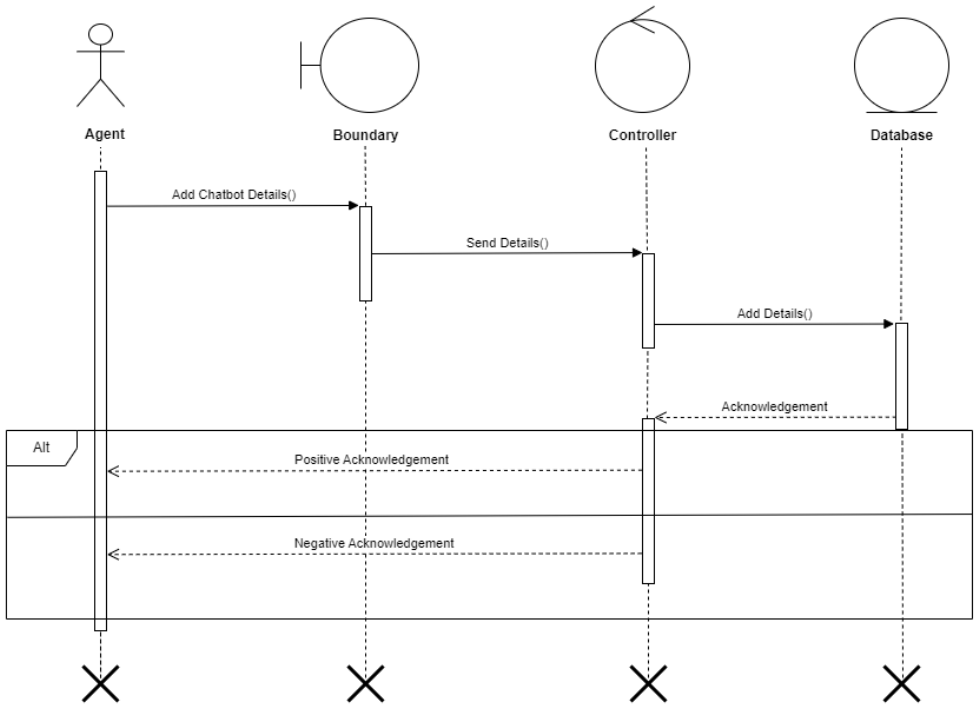
Agent can Login details like username and password then it will system validation and send details Login Controller its validate then it fetch Database then it will send acknowledgement then Login successfully. Then Agent can Add chat then it will fetch the database then it will show question to agent. Client also do that chat with agent if agent is avail then it will chat then it will store the database.

- **Sequence Diagrams for Chat :-**



Client/agent can Login details like username and password then it will system validation and send details Login Controller its validate then it fetch Database then it will send acknowledgement then Login successfully. It send request to controller then it will Request to access through database
If they are avail then it will accept otherwise not. Then it will write message then send message

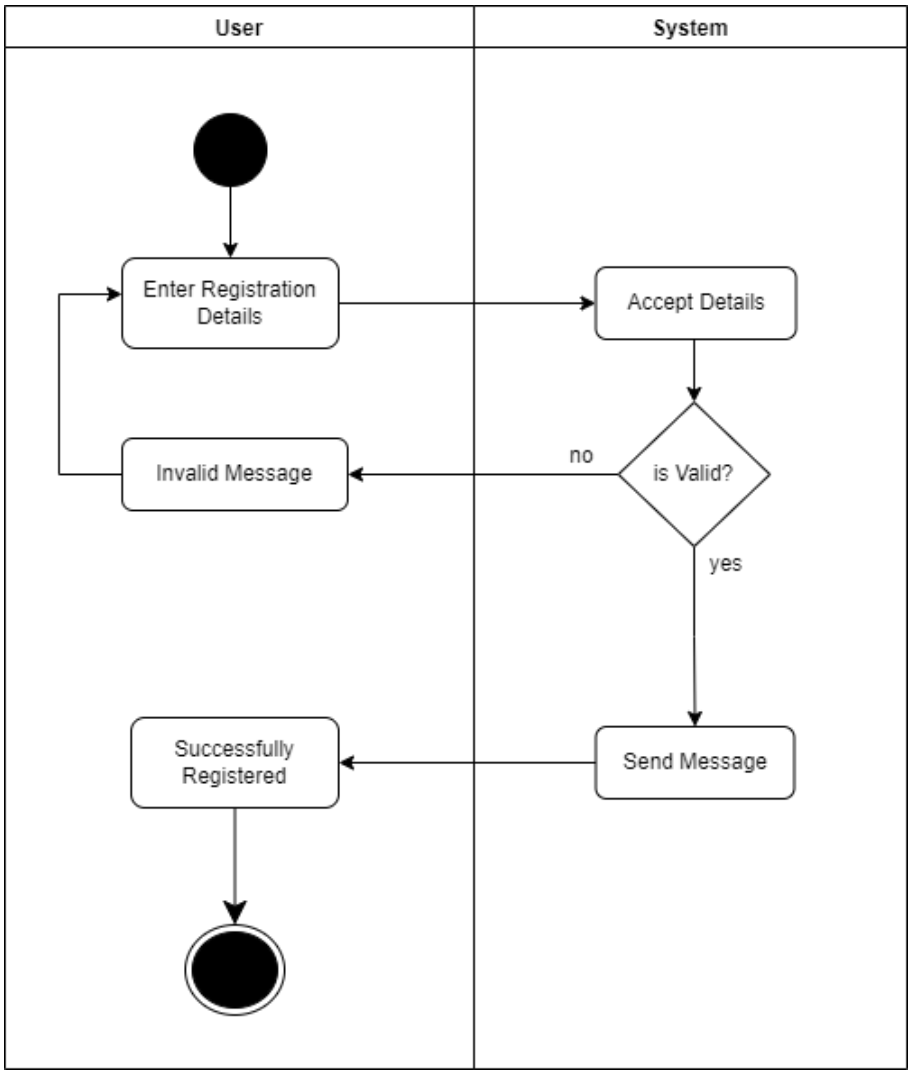
- **Sequence Diagrams for Chatbot :-**



Agent can add chatbot details then send details to controller it will check the details after it will store the database then it will send acknowledge chatbot added successfully

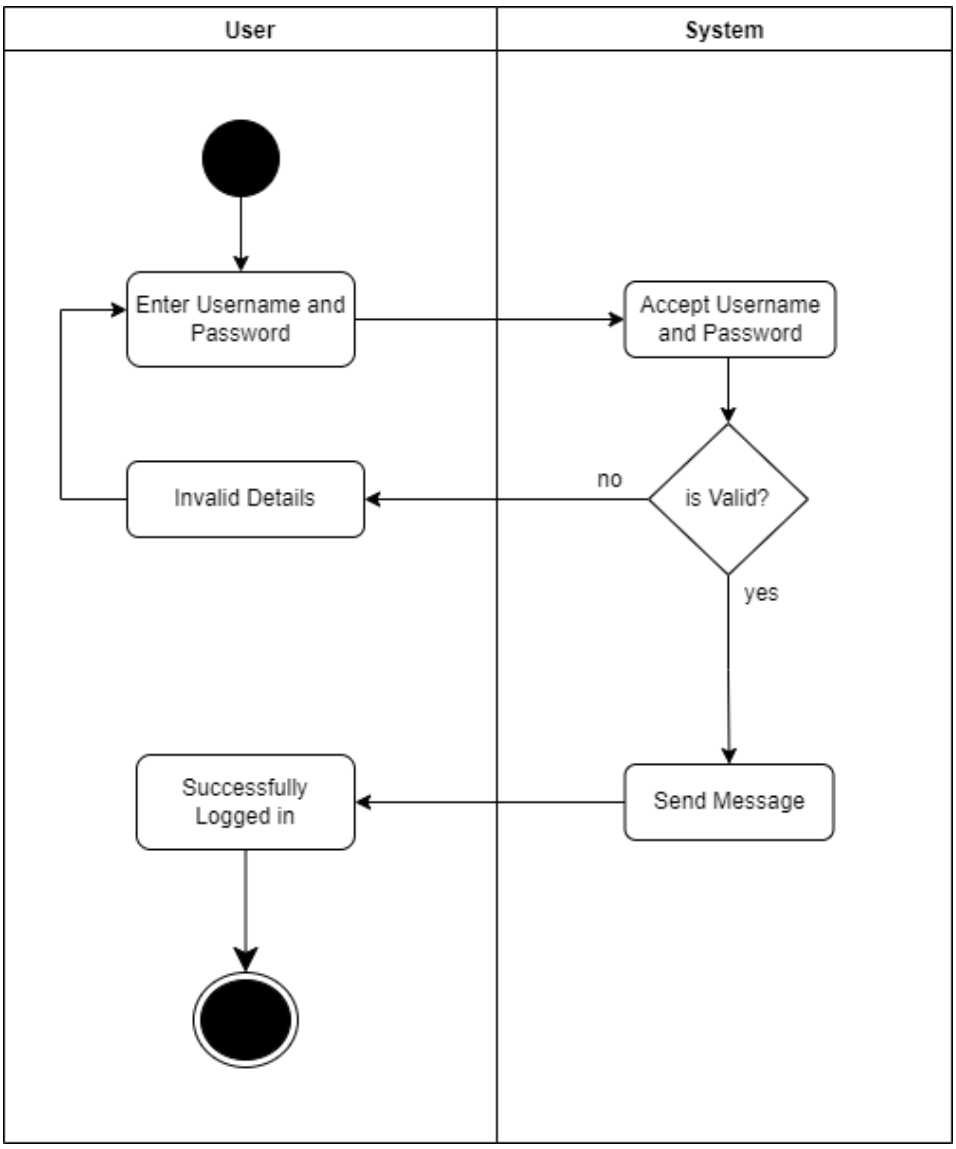
3.4 Activity Diagram

- **Activity Diagrams for Registration:-**



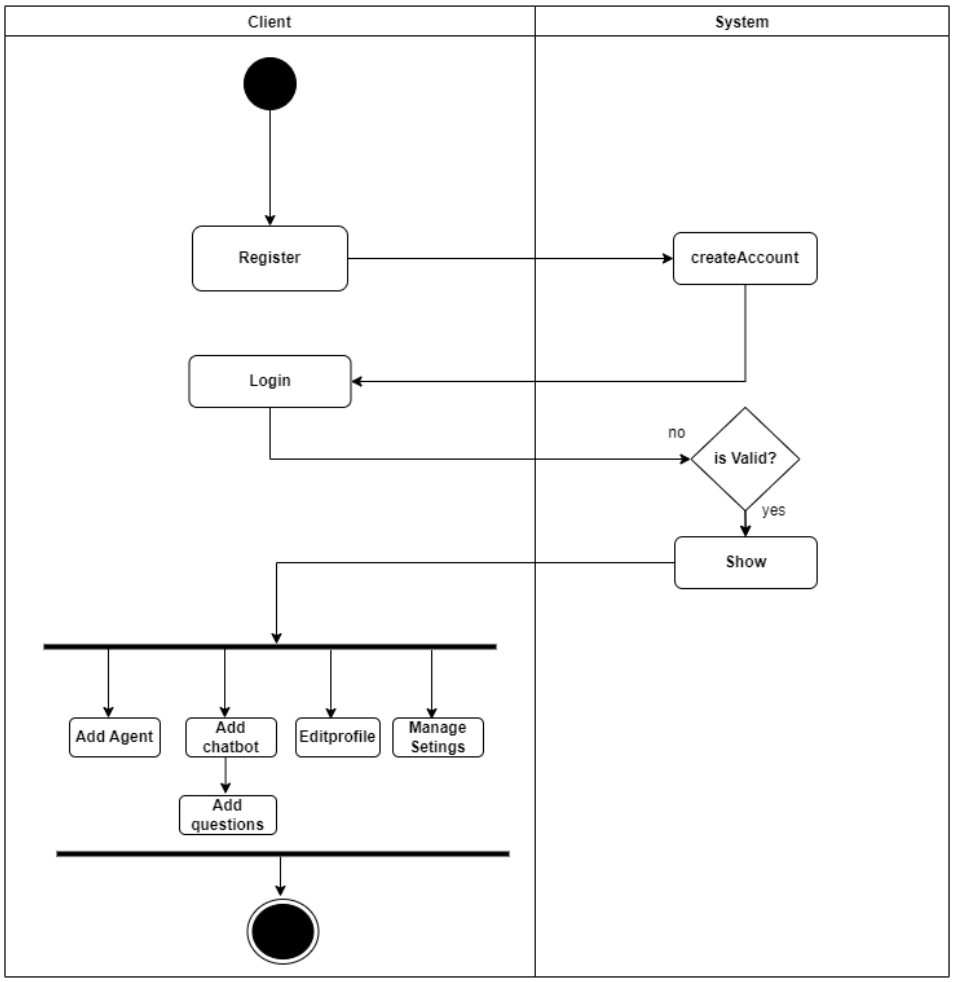
User can enter the registration then System can validating the details then if invalid so show invalid message and if Valid send message Successfully Registered.

- **Activity Diagrams for Login:-**



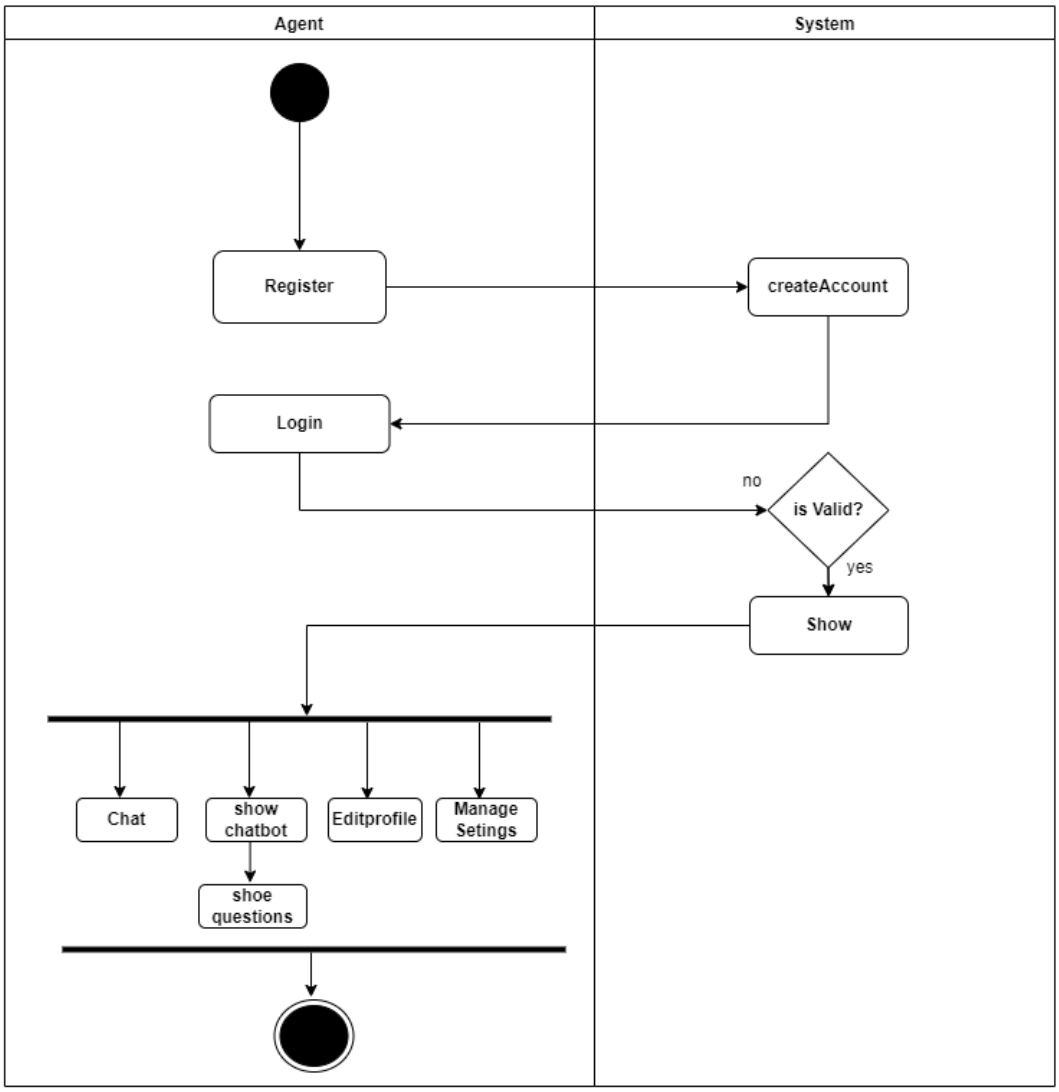
User can Enter Username and Password then System can veriftying the username and password if valid then send Message Logged Successfully and if Invalid so it will send invalid message

- **Activity Diagrams for Client:-**



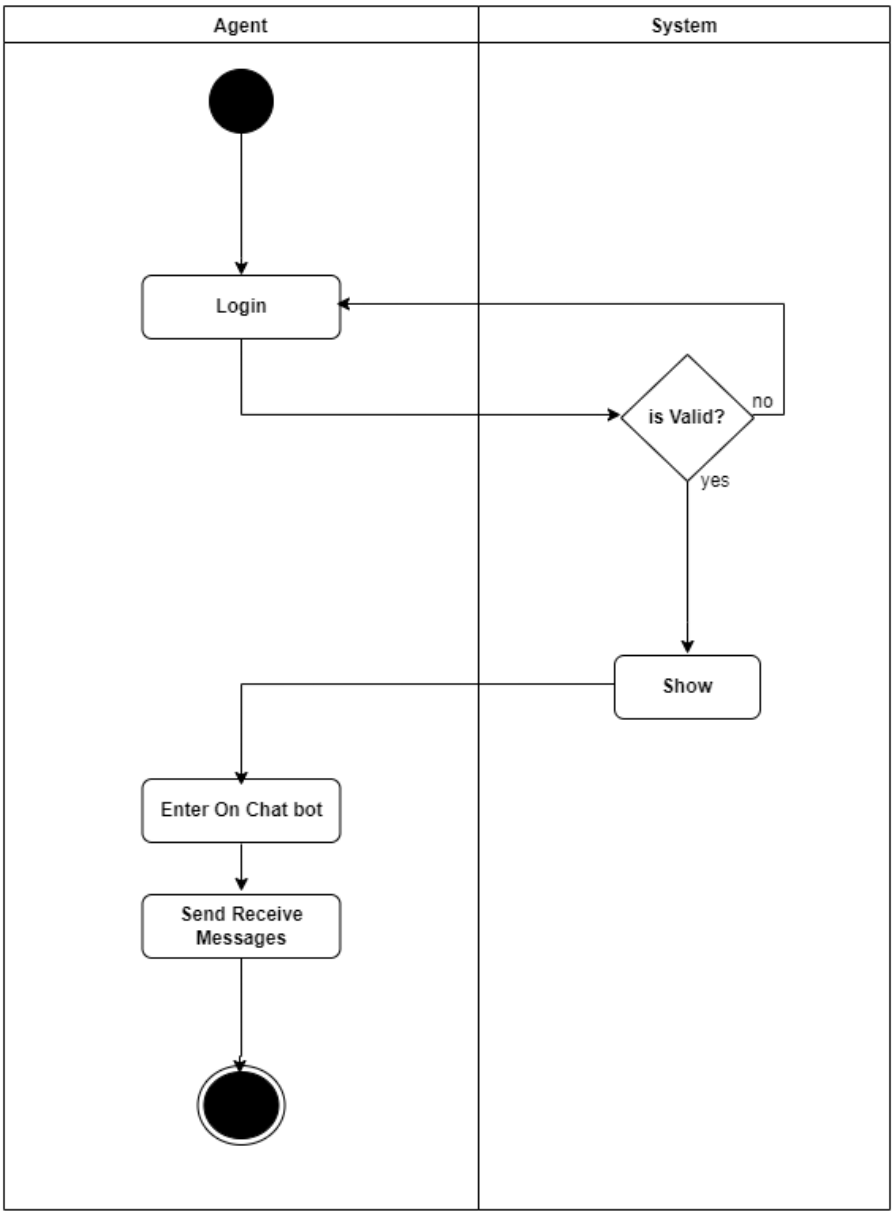
Client can Register and it will check registration details the if registration successfully then go to the Login it will check Username and password if correct so then go to the home page and it will show and perform Add Agent and Add chatbot and Edit profile and Mange settings and add Question if all the process done then logout Client

- **Activity Diagrams for Agent :-**



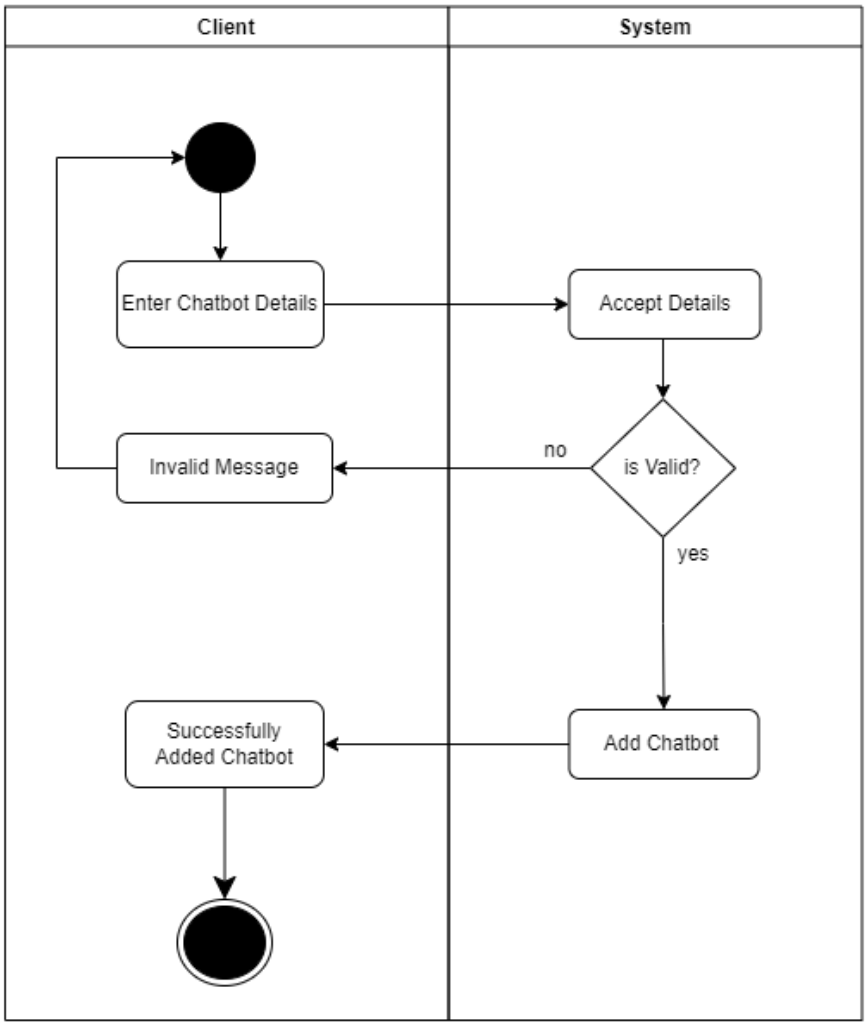
Agent can Register and it will check registration details the if registration successfully then go to the Login it will check Username and password if correct so then go to the home page and it will show and perform chat and show chatbot and Edit profile and Mange settings and show Question if all the process done then logout Age

- **Activity Diagrams for Chat:-**



Agent can login then System will check Username and password if valid then Enter On chatbot, Send and receive message then Agent can logout

- **Activity Diagrams for Chatbot :-**



Client can Enter chatbot details like image and name, choose colour and position of chatbot, System can accept details and add chatbot to database.

3.5 Data Dictionary

- **Data Dictionary for Admin:-**

Field Name	Datatype (Size)	Description
Admin_Id	Number (10)	Auto Generated by the Database admin
user_name	Varchar (20)	Name of user
Email_id	Varchar (48)	It indicates the email Address of admin
Password	Varchar (10)	It indicates admin password
Picture	Varchar(10)	It indicates picture of admin
ForgotToken	Varchar(30)	It indicates forgot token
Timestamps	Enum("T","F")	It indicate time stamp

- **Data Dictionary for Client:-**

Field Name	Datatype (Size)	Description
Client_id	Number(10)	Auto Generated by the Database user.
Company_name	Varchar (30)	Company Name of client
Address	Text	It indicates the address of client
Pincode	Number (10)	It indicates pincode of client
City	Varchar(30)	It indicates city of client
Api_key	Number(10)	It generate api key

- **Data Dictionary for Agent:-**

Field Name	Datatype (Size)	Description
Agent_id	Number(10)	Auto Generated by the Database User.
Client_id	Number (10)	It indicates client id
Role	Varchar (20)	It indicates role of agent
First_name	Varchar(10)	It indicates first name of agent
Last_name	Varchar(10)	It indicates last name
Email_id	Varchar(30)	It indicates emailid
Password	Varchar(30)	It indicates user Password
IsEmailverified	Enum("T","F")	Verify email
mobile	Number(10)	It indicates mobile number
isBlock	Enum(“T”,”F”)	It indicates isBlock or not
forgotToken	Varchar(30)	It indicates forgot token
picture	Varchar(30)	It indicates picture of agent

- **Data Dictionary for Chatbot :-**

Field Name	Datatype (Size)	Description
Bot_id	Number (10)	Auto Generated by the Database.
Api_key	Varchar (40)	It generates api key
Icon	Varchar (40)	It indicates icon of chatbot
align	Varchar(30)	It indicates the align of chatbot
Background	Varchar(30)	It indicates background of chatbot
Menu	Number(10)	Fetch data from chatbot
Timestamps	Enum("T","F")	It indicates time

- **Data Dictionary for Chat:-**

Field Name	Datatype (Size)	Description
Chat_id	Number (10)	Auto Generated by the Database user.
Room_id	Number (10)	Auto
User_id	Number (10)	Fetch data from user table
Agent_id	Number(10)	Fetch data from agent table
Message	Text	It indicates message of chat
Timestamps	Enum("T","F")	It indicates the time of chat

- **Data Dictionary for Room:-**

Field Name	Datatype (Size)	Description
Room_id	Number (10)	Auto Generated by the Database user.
Bot_id	Number(10)	Fetch data from chatbot table
Api_key	Varchar (40)	It generate api key
User_id	Number(10)	Fetch data from user table
Agent_id	Number (10)	Fetch data from agent table

- **Data Dictionary for User:-**

Field Name	Datatype (Size)	Description
User_id	Number (10)	Fetch databfrom user table
First_name	Varchar (20)	It indicates first_name of user
Last_name	Varchar (20)	It indicates the last name of user
Full_name	Varchar (20)	It indicates the full name of user
Email_id	Varchar (10)	It indicates email of user
Mobile	Number(10)	It indicates mobile number of user
IsBlock	Enum("T","F")	It check IsBlock or not
Bot_id	Number (10)	Fetch data from the chatbot table

4.Development

4.1 Coding Standards

React best practices:

1. Follow the coding style of React keeping component codes under 500 lines of code and functions under 80 lines of code.

2. React folder structure

- a. Maintaining a clean and precise folder structure of an React application is regarded as one of the most important factors for judging the quality of an React app. Proper folder structure, when maintained, can help not only in better code readability but can also help to scale the application whenever needed. Below is the way one can understand the folder structure of an React app.
- b. Core module – contains providers for the independent or individual services that are loaded when an application starts.
- c. Shared module – It usually contains the components, directives, and pipes that are used at several places in the app. It should also have the declarations which could be exported to be used in different modules. But we should remember to not put any services at the root level.
- d. Home Module – Here you can create and define components of a page or separate pages depending on the size of the application. You can set up routing in this module's routing module.
- e. Config module – this would simply contain the configuration services and configuration APIs for the app.

Node.js best practices:

1. Use HTTP methods and API routes

2. While creating RESTful APIs for basic CRUD operations, you should use the predefined routes PUT, POST, PATCH, DELETE, GET. Also, the name of the routes should use nouns and must be according to the use case. For eg. A route to delete a student from a college app the route must be DELETE /student/:id

3. Use Stateless Authentications: Your REST APIs and your authentication component should be both stateless. Stateless authentication could be achieved with the help of JWT (JSON Web Token)

4. Properly use status codes: While error handling is an important part of any function, HTTP services must also follow the same. To indicate the status of a request, proper status codes should be used.

MongoDB best practices:

- Understand Schema Differences Between Relational and Document-based Databases.
- Embed Your Data Instead of Relying on Joins.
- Use Indexes For Frequent Operations.
- Properly Size Your Servers.
- Use Replication or Sharding.

4.2 Sample Code

- **Registration:**

```
const register = async function (req, res) {

  const email = await agents.findOne({ email: req.body.email });
  if (email)
    return res
      .status(400)
      .json({ success: false, message: messages.emailExists });

  const phone = await agents.findOne({ mobile: req.body.mobile });
  if (phone)
    return res
      .status(400)
      .json({ success: false, message: messages.phoneExists });
  let hashVal = await hash(req.body.password, parseInt(process.env.JWT_SALT));
  req.body.password = hashVal;
  req.body["fullName"] = `${req.body.firstName} ${req.body.lastName}`;
  req.body["apiKey"] = (Math.random() + 1).toString(36).substring(2);
  const addClient = new clients(req.body);
  addClient.save().then(async (result) => {
    const addAgent = new agents(req.body);
    addAgent.clientId = result.clientId;
    addAgent.role = "Admin";
    addAgent.picture = "default-avatar.png";
    let agentRes;
    try {
      agentRes = await addAgent.save();
    } catch (err) {
      return res
        .status(400)
        .json({ success: false, message: messages.serverError });
    }
    let subject = "Verify email address";
    let text = "Verify email address";
    let tokenDetails = await generateToken({ userId: agentRes._id });
    fs.readFile("html/verifyemail.html", "utf-8", async function (err, data) {
      let html = data.replace("tokenDetails", `${tokenDetails}`);
      html = html.replace("SERVER_URL", process.env.SERVER_URL);
      html = html.replace(/Users/g, req.body.fullName);
      sendMail({ to: req.body.email, subject, text, html });
      return res
        .status(200)
        .json({ success: true, message: messages.userRegister });
    });
  });
};
```

- **Login:**

```
const login = async function (req, res) {
  let Agent = await agents.findOne({ email: req.body.email });
  if (Agent) {
    if (Agent.isBlock)
      return res
        .status(401)
        .json({ success: false, message: messages.userBlock });
    if (Agent.isEmailverified == false)
      return res
        .status(400)
        .json({ success: false, message: messages.emailNotverify });

    let checkPassword = await compare(req.body.password, Agent.password);
    if (checkPassword) {
      let token = jwt.sign(
        {id: Agent.agentId,
          clientId: Agent.clientId},
        process.env.JWT_SECRET,
        {expiresIn: "7d"}
      );
      let client = await clients
        .findOne({ clientId: Agent.clientId }).select("apiKey");
      return res.status(200).json({
        success: true,
        message: messages.login,
        token,
        data: {
          id: Agent.agentId,
          clientId: Agent.clientId,
          apiKey: client.apiKey,
          role: Agent.role,
          firstName: Agent.firstName,
          lastName: Agent.lastName,
          fullName: Agent.fullName,
          picture: Agent.picture,
          gender: Agent.gender,
        },
      });
    } else {
      return res
        .status(401)
        .json({ success: false, message: messages.wrongPassword });
    }
  } else {
    res.status(401).json({ success: false, message: messages.wrongEmail });
  }
};
```

- **Bot Integration:**

```
<script type="text/javascript">
      (function () {
        var s1 = document.createElement("script"),
          s0 = document.getElementsByTagName("script")[0];
        s1.async = true;
        s1.src = "http://localhost:5000/bot/auth/5ukl8zm95vf";
        s1.id = "quickConnect";
        s1.charset = "UTF-8";
        s1.setAttribute("crossorigin", "*");
        s1.setAttribute("apiKey", "5ukl8zm95vf");
        s0.parentNode.insertBefore(s1, s0);
      })();
    </script>
```

- **Socket init admin:**

```
useEffect(() => {
    const ioCon = io(process.env.REACT_APP_API);
    dispatch(socketUpdate(ioCon));
    getData();
    return () => dispatch(socketUpdate(null));
  }, []);

  useEffect(() => {
    if (!socket) return;

    socket.on("connect", () => {
      socket.emit("join", { userId, apiKey, isAgent: true });
    });
    socket.on("disconnect", () => {});
    socket.on("initLiveChat", (data) => {
      dispatch(newCustomer(data));
    });
  }, [socket]);
```

- **Socket init chatbot:**

```
socket = io("http://localhost:5000");

    socket.on("connect", () => {
      socket.emit("join", { botId, userId, isAgent: false });
    });

    socket.on("receiveMes", function (data) {
      console.log(data);
```

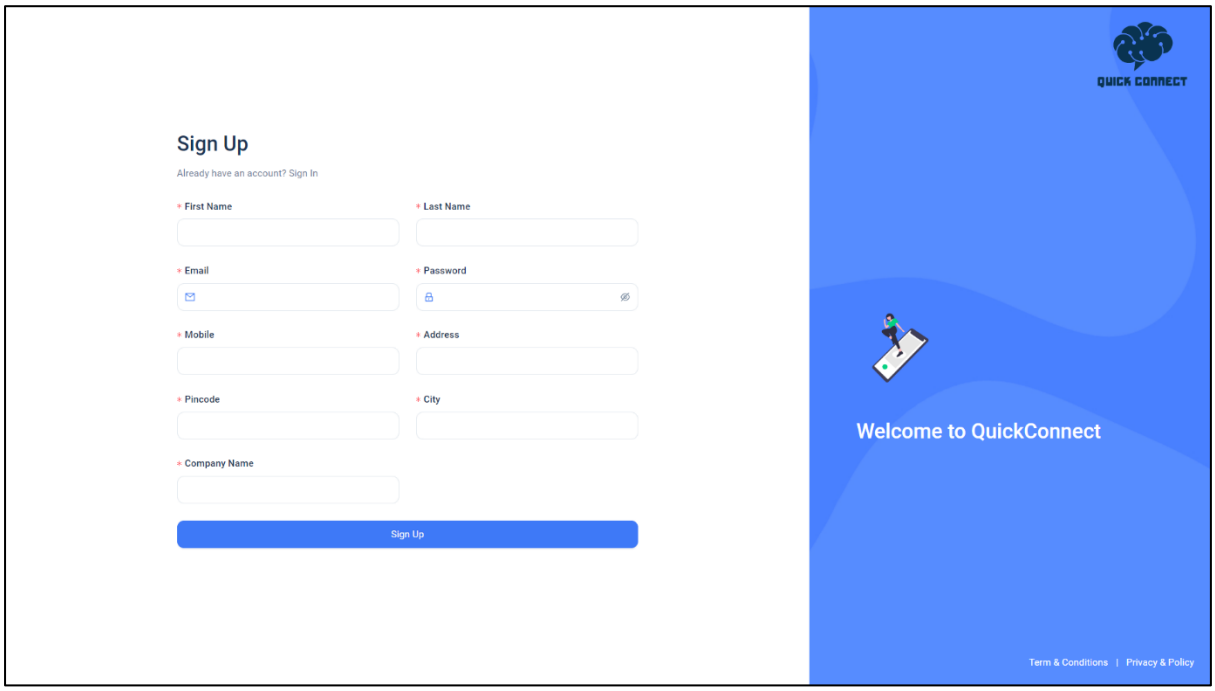
```
    createChat(data?.mes, "l");
  });

  socket.on("agentAllocated", function (data) {
    roomId = data?.roomId;
    agentId = data?.agentId;
    agentName.innerHTML = data?.fullName;
    data?.picture &&
      (AGENT_IMG = API_URL + "api/agent/picture/50/" + data?.picture);
    createChat(data?.fullName + " is connected", "l");
  });
```

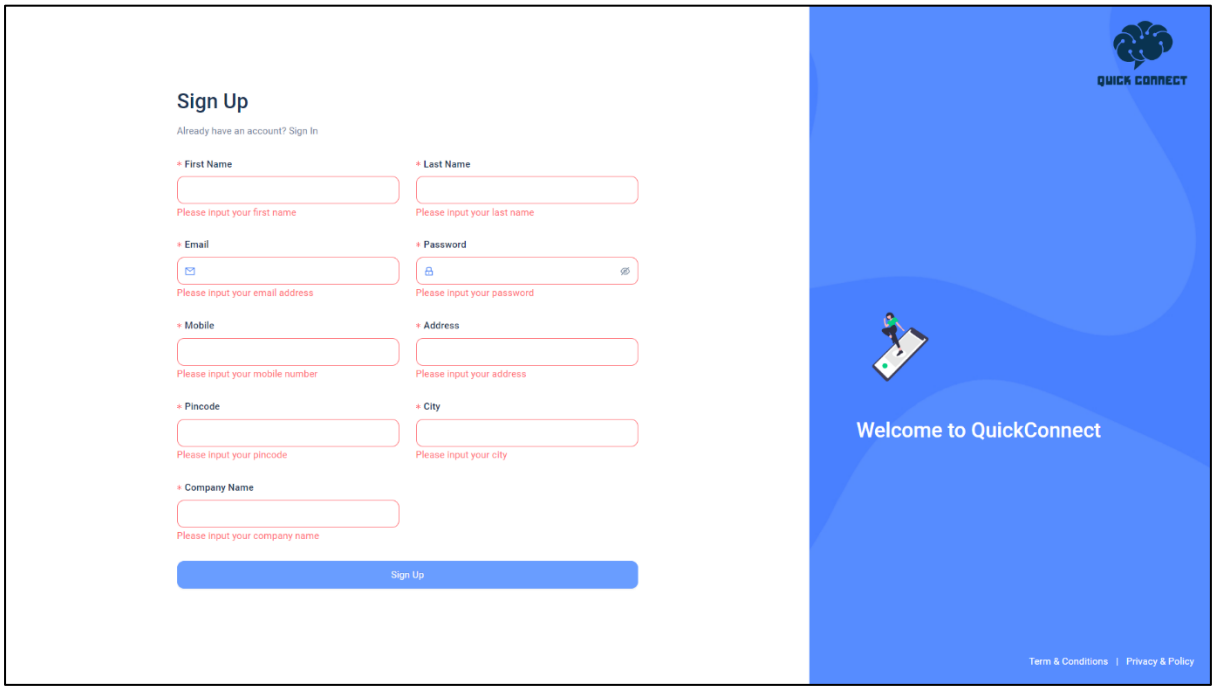
- **Socket init server:**

```
io = require("socket.io")(server, {
    cors: {
      origin: "*", // ["http://localhost:3000", "http://localhost:5500"],
      methods: ["GET", "POST"],
    },
  });
  io.on("connection", (socket) => {
    socket.on("join", function (data) {
      socket.userId = data.userId;
      socket.isAgent = data.isAgent;
      if (data.isAgent) {
        socket.apiKey = data.apiKey;
        if (!agents[data.apiKey]) {
          agents[data.apiKey] = {};
        }
        if (!agents[data.apiKey][data.userId]) {
          agents[data.apiKey][data.userId] = [];
        }
        agents[data.apiKey][data.userId].push(socket.id);
      } else {
        socket.botId = data.botId;
        if (!users[data.botId]) {
          users[data.botId] = {};
        }
        if (!users[data.botId][data.userId]) {
          newUser({
            botId: data?.botId,
            userId: data?.userId,
          });
          users[data.botId][data.userId] = [];
        }
        users[data.botId][data.userId].push(socket.id);
      } });
```

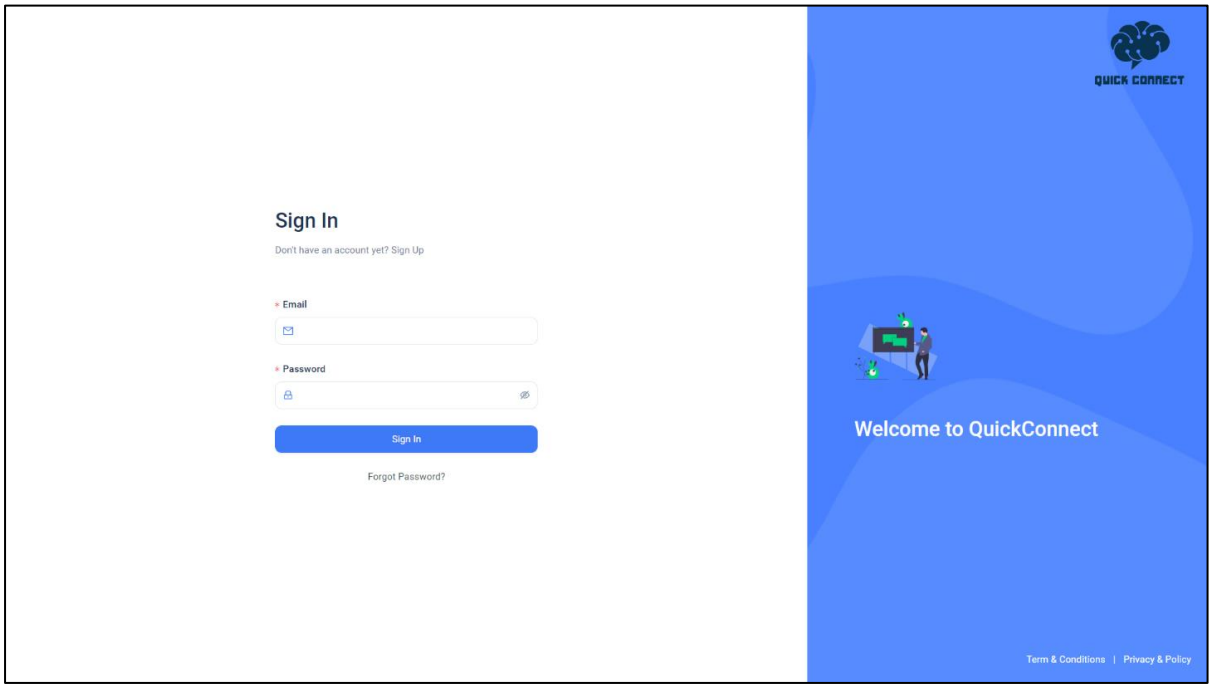
4.3.Working project screenshot

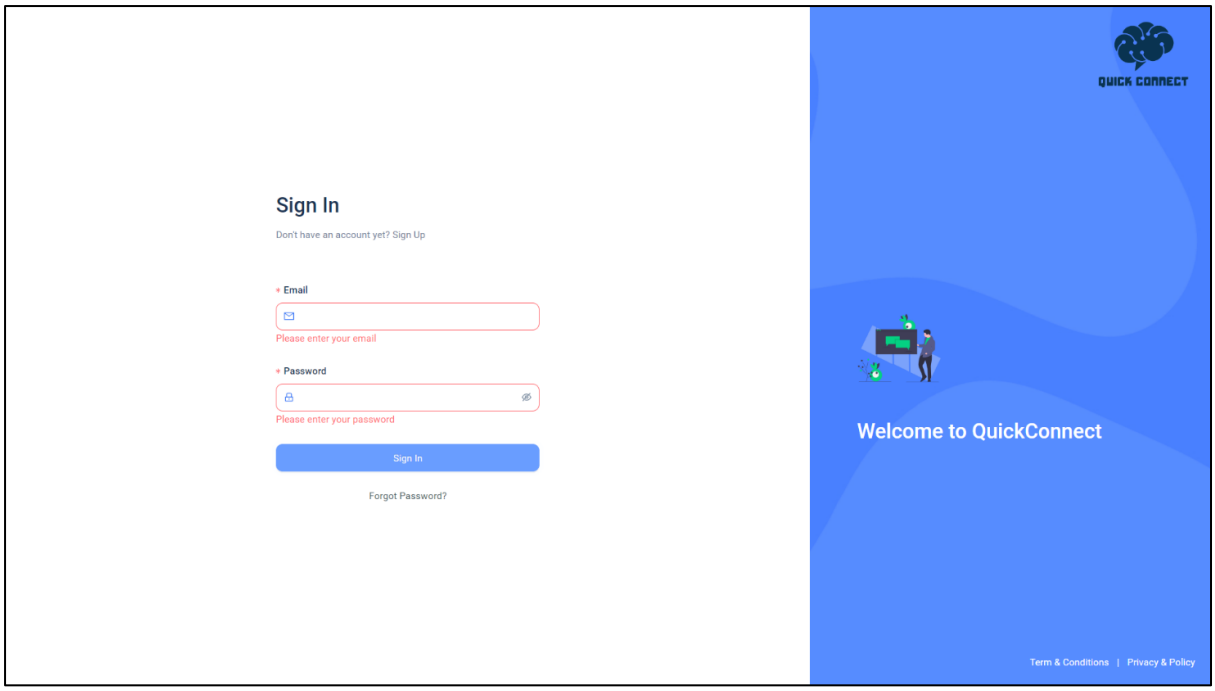
SignUp:-



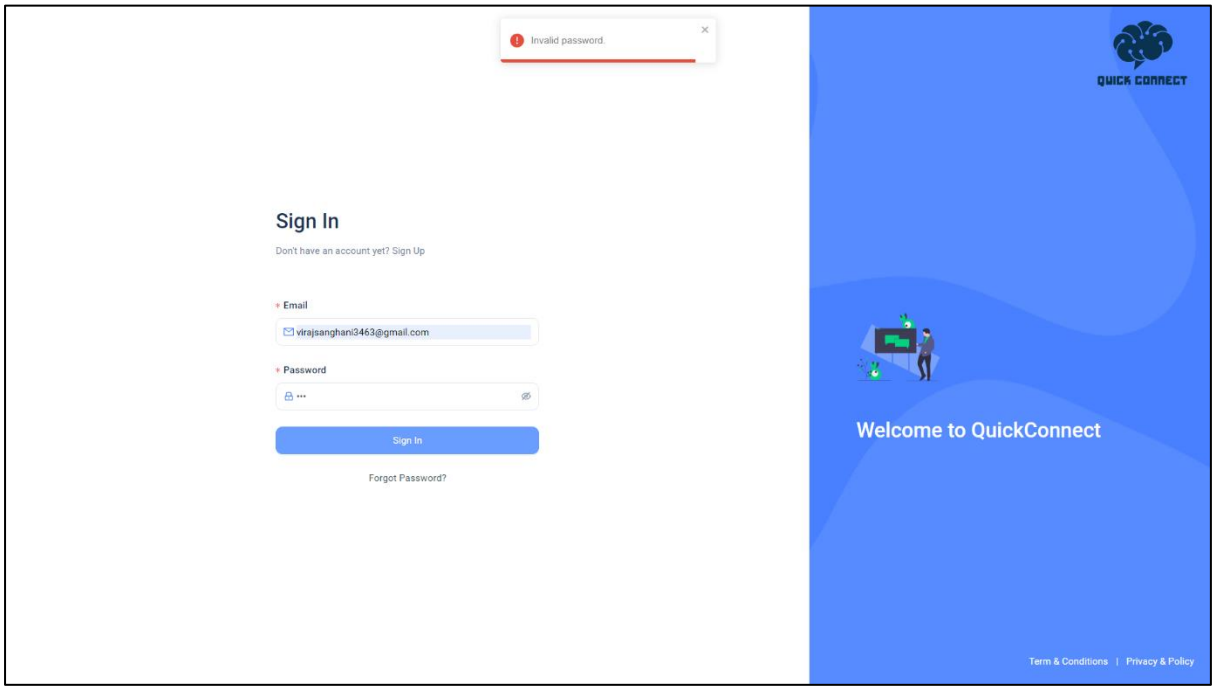


Sign In:-

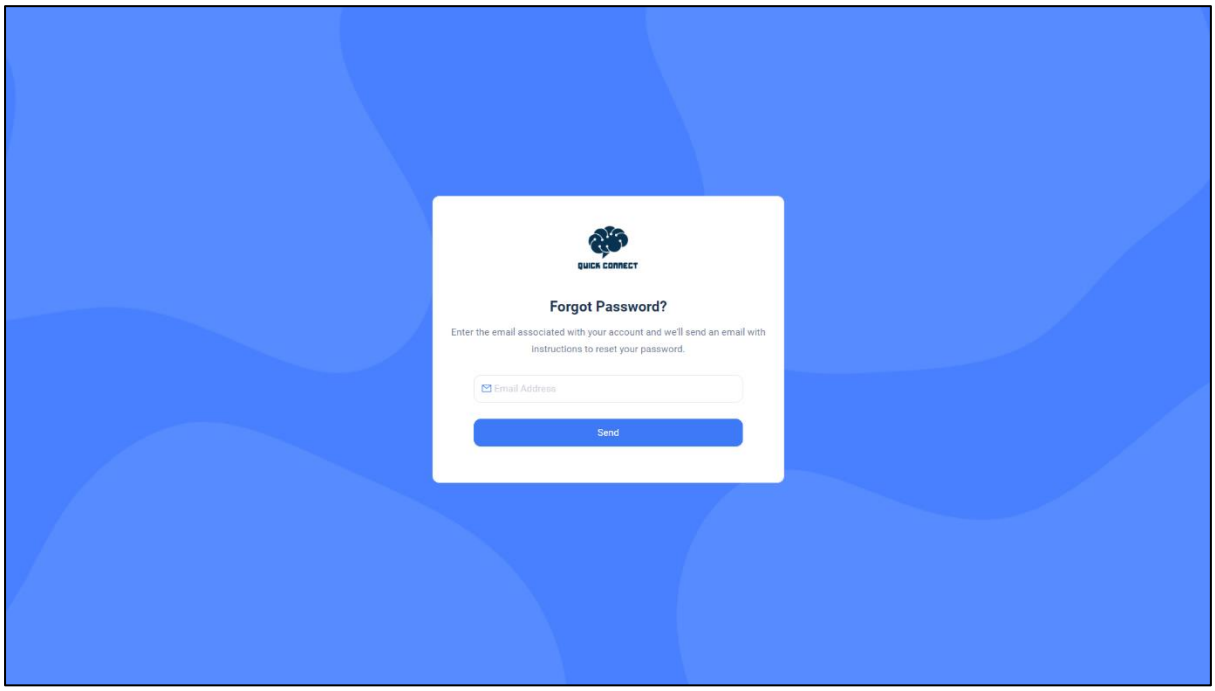

Sign In
Don't have an account yet? Sign Up
* Email
* Password
Sign In
Forgot Password?
QUICK CONNECT
Welcome to QuickConnect
Term & Conditions | Privacy & Policy


Sign In
Don't have an account yet? Sign Up
* Email
Please enter your email
* Password
Please enter your password
Sign In
Forgot Password?
QUICK CONNECT
Welcome to QuickConnect
Term & Conditions | Privacy & Policy

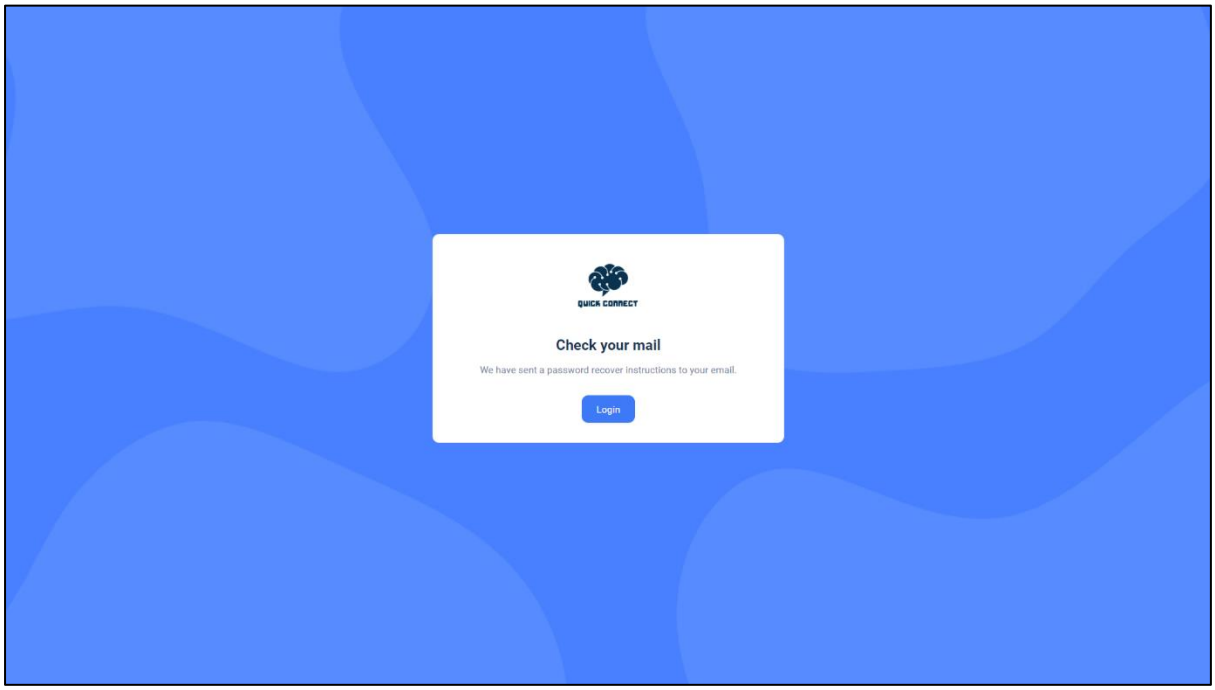
Sign In:-



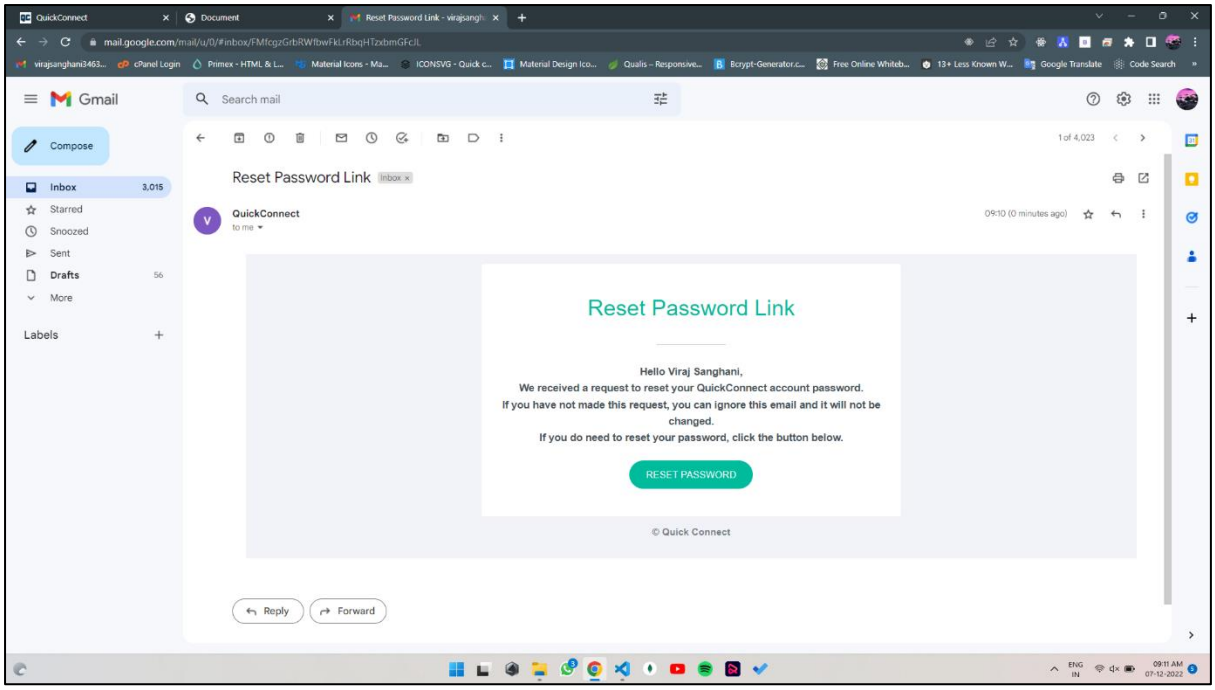
Forgot Password:-



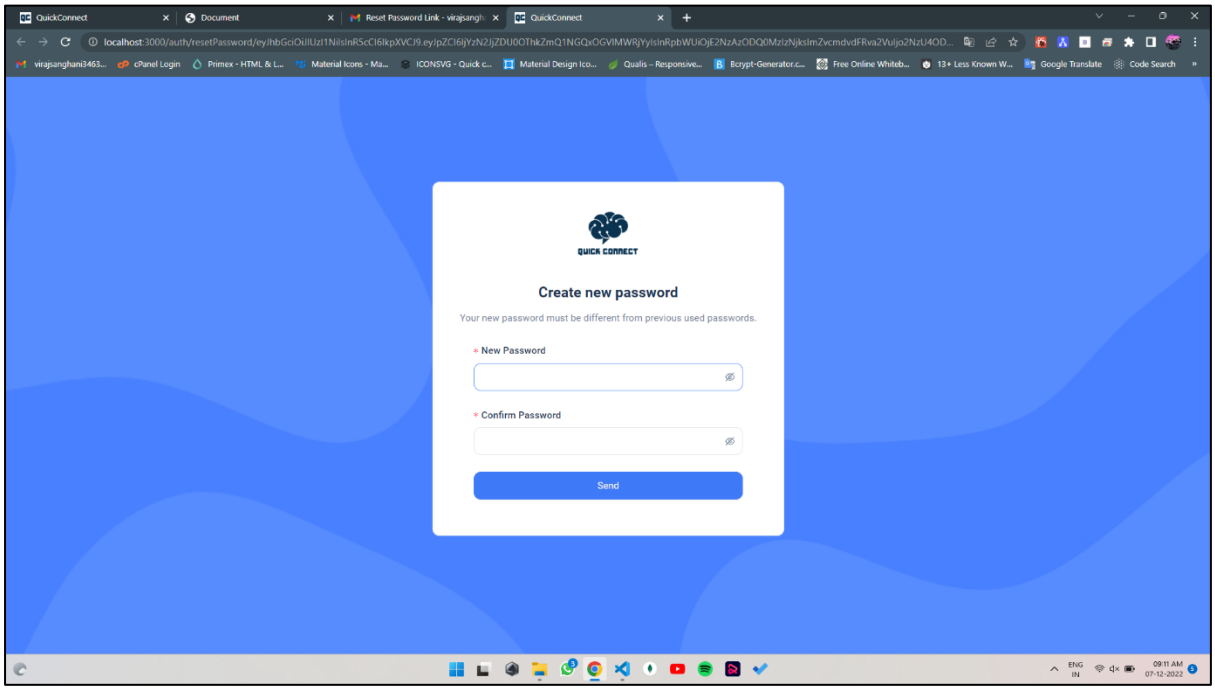
Forgot Password:-

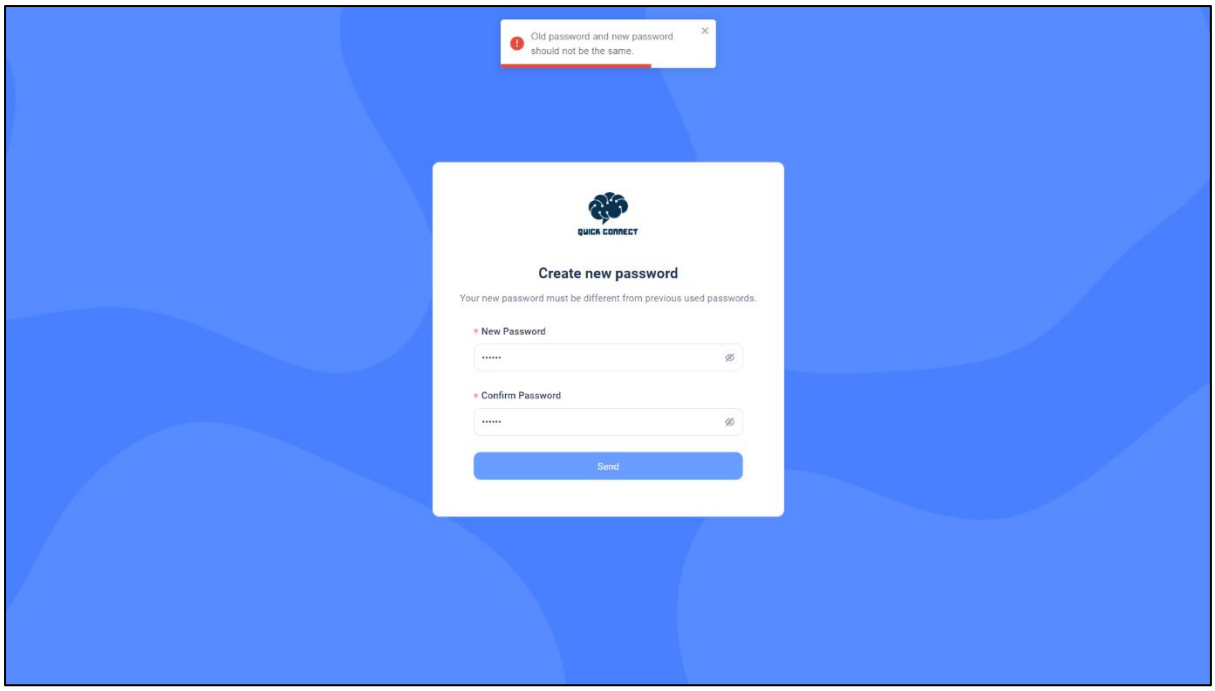


Reset Password Link:-

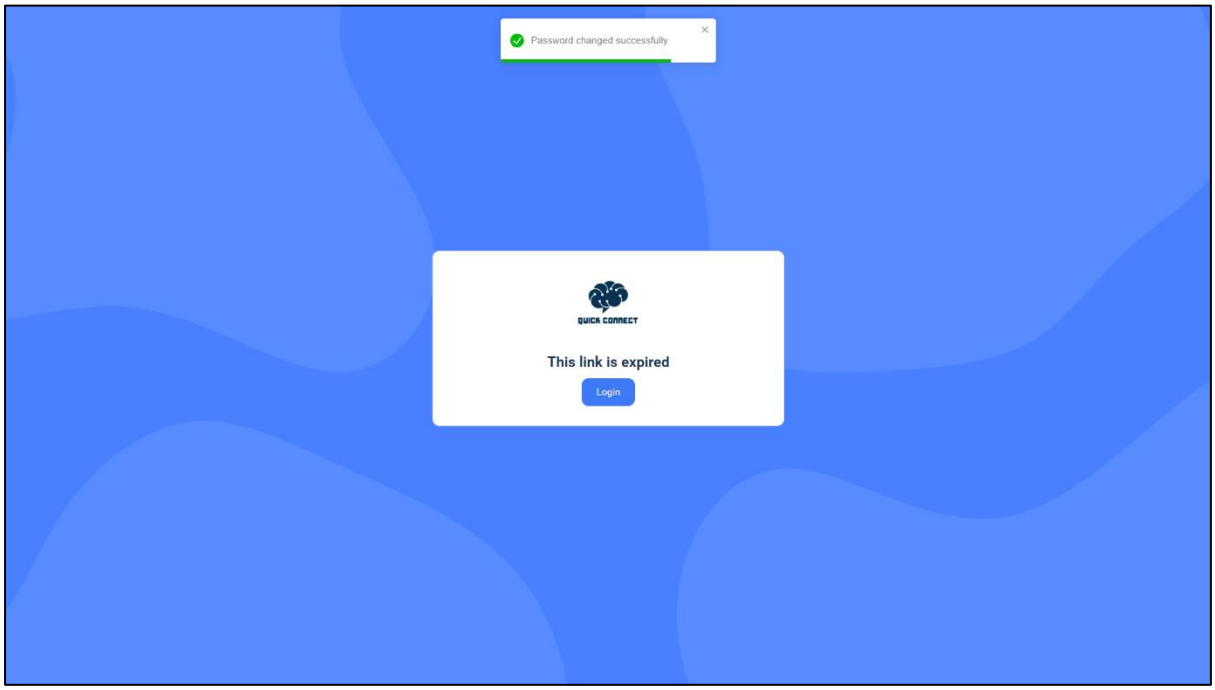


Create new Password:-

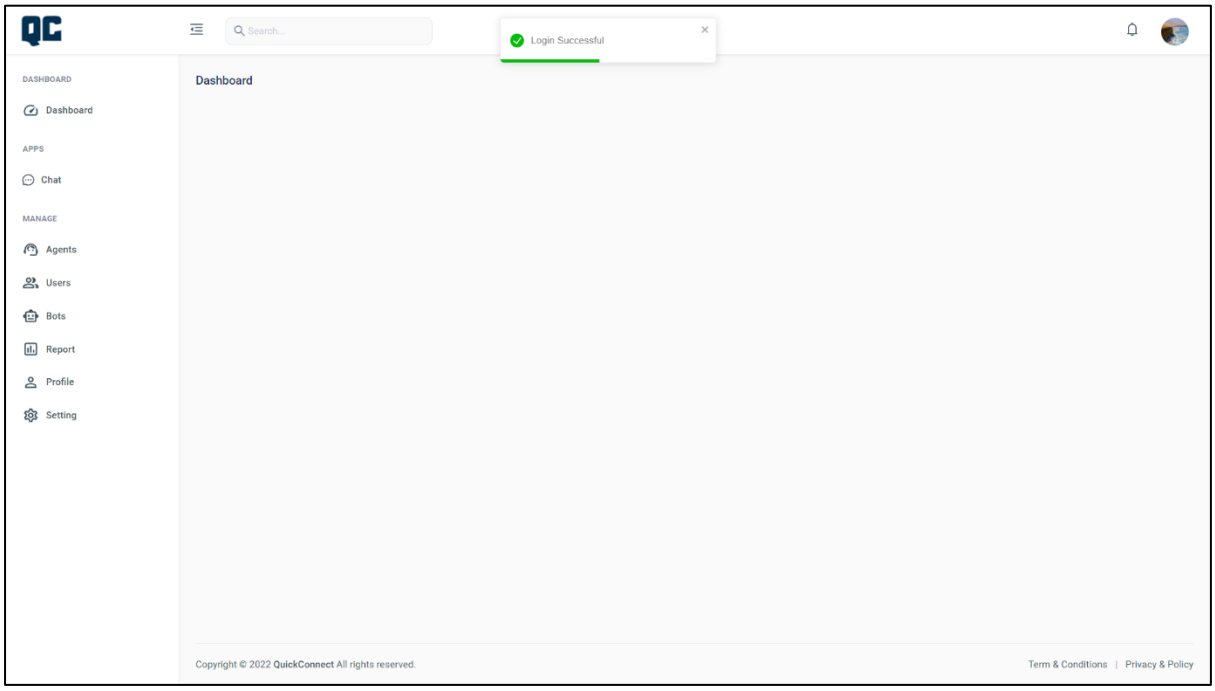




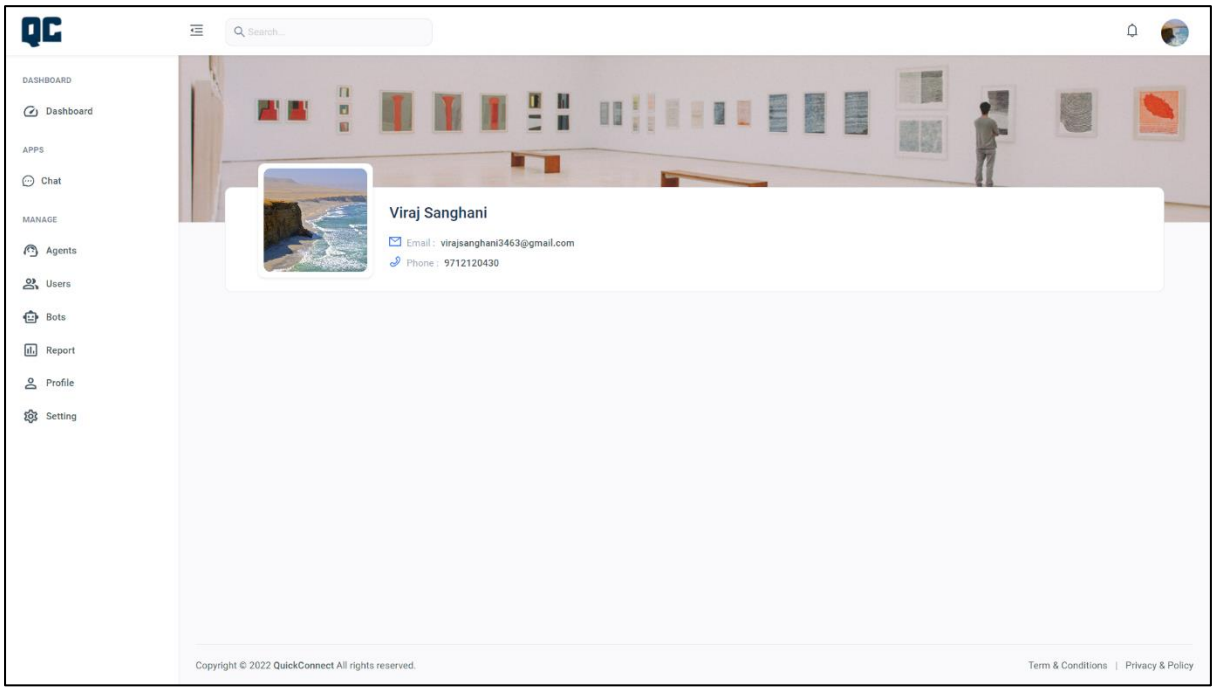
Password change Success:-



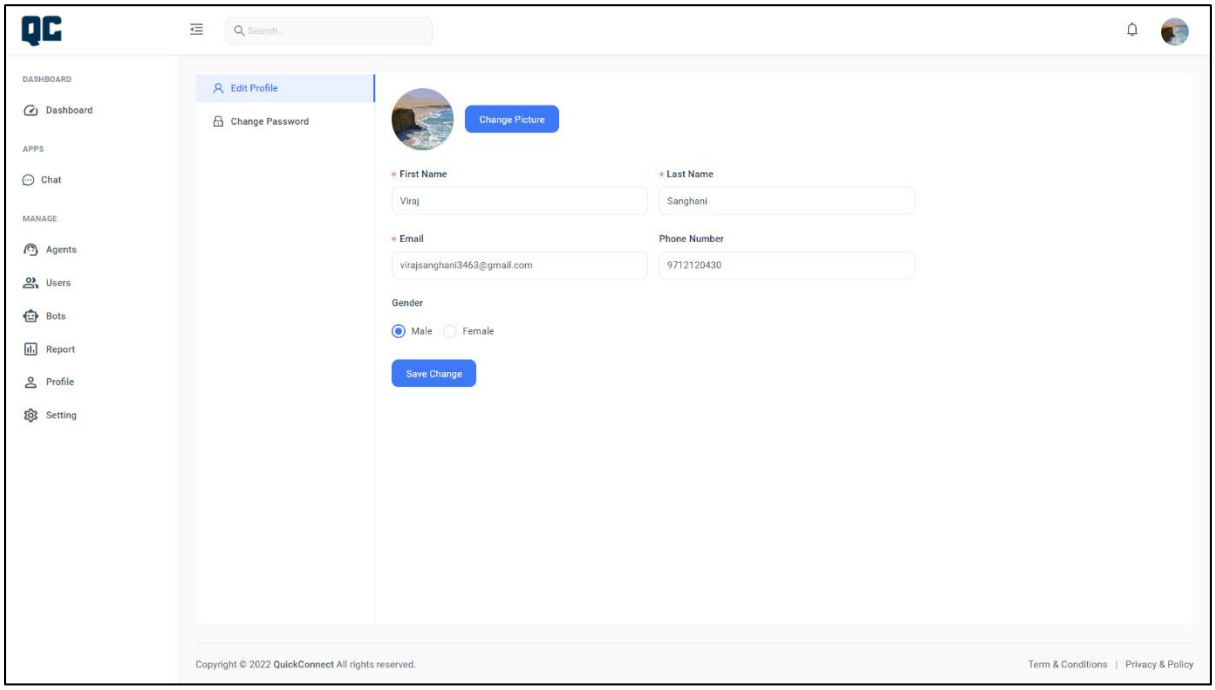
Admin:-



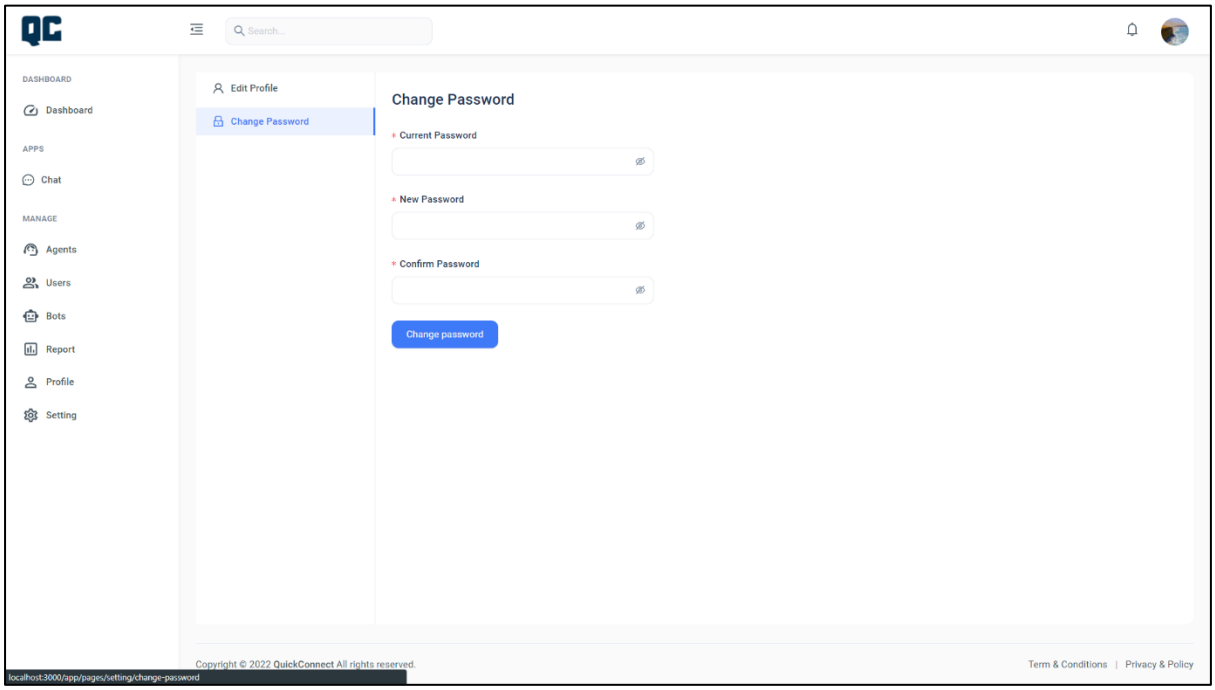
Admin Profile:-



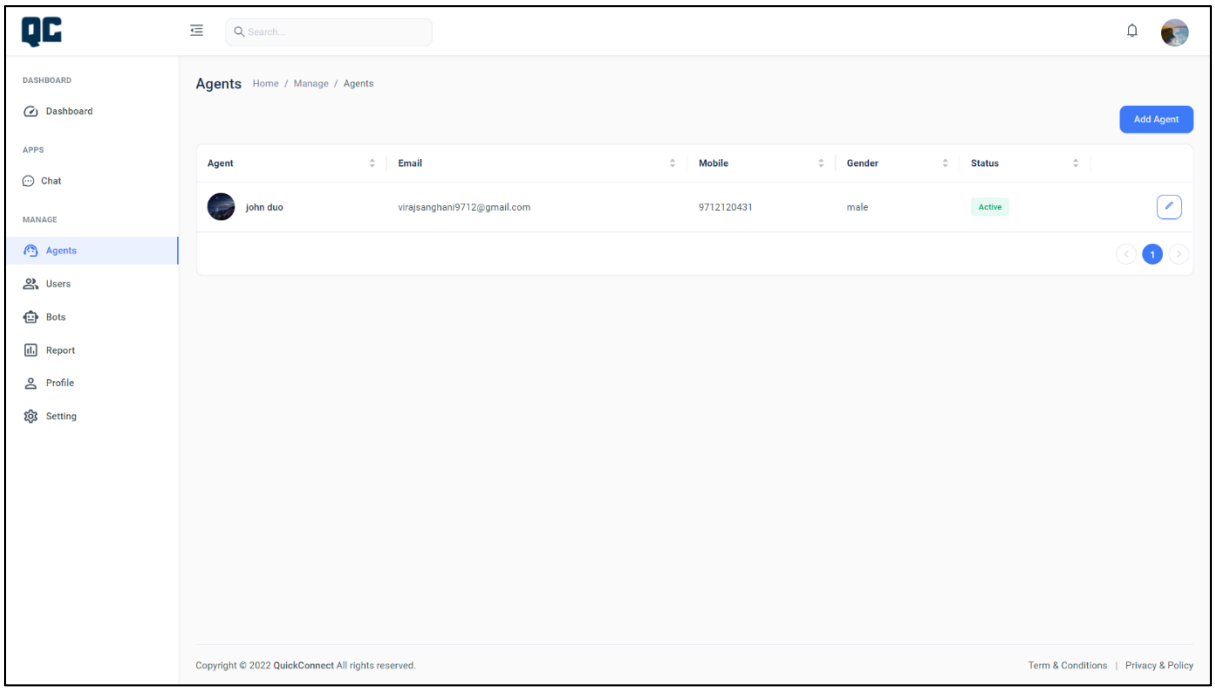
Edit Profile:-



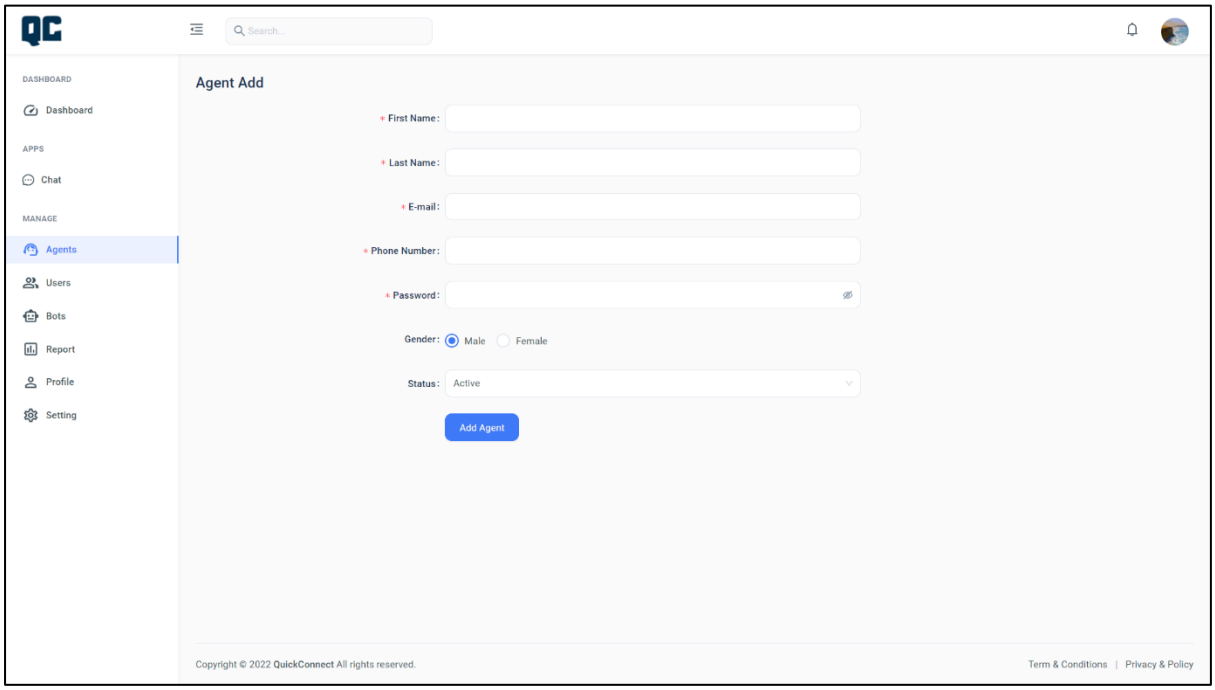
Change Password:-



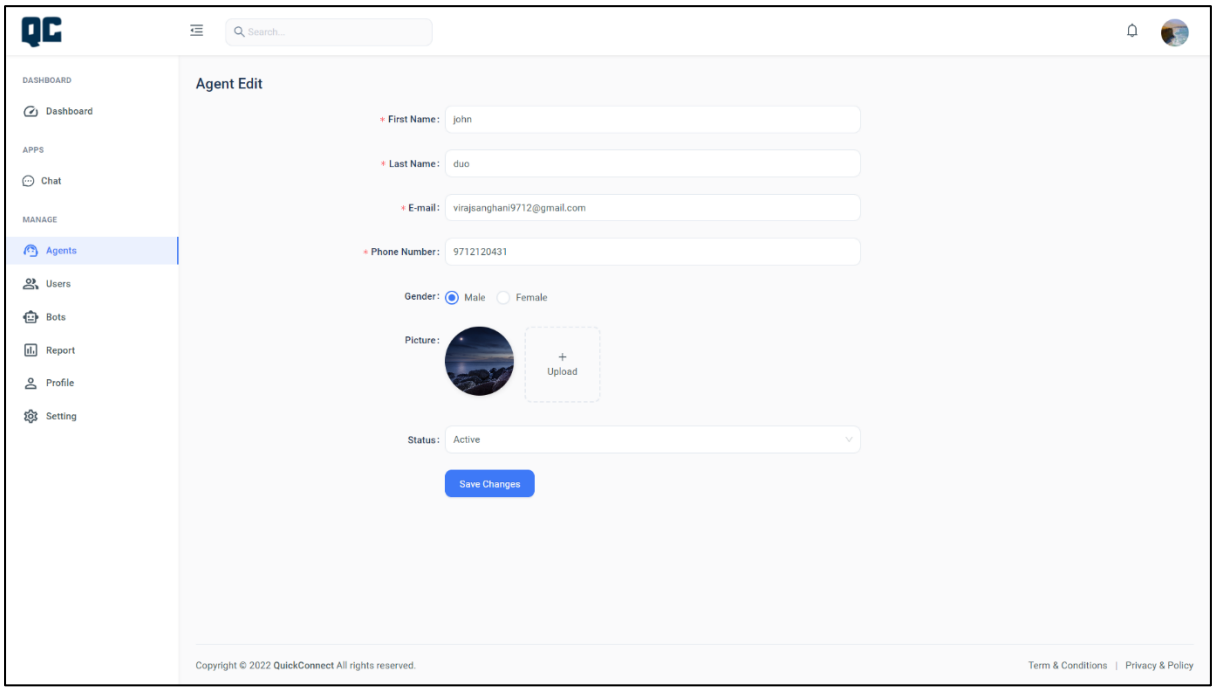
Agent-



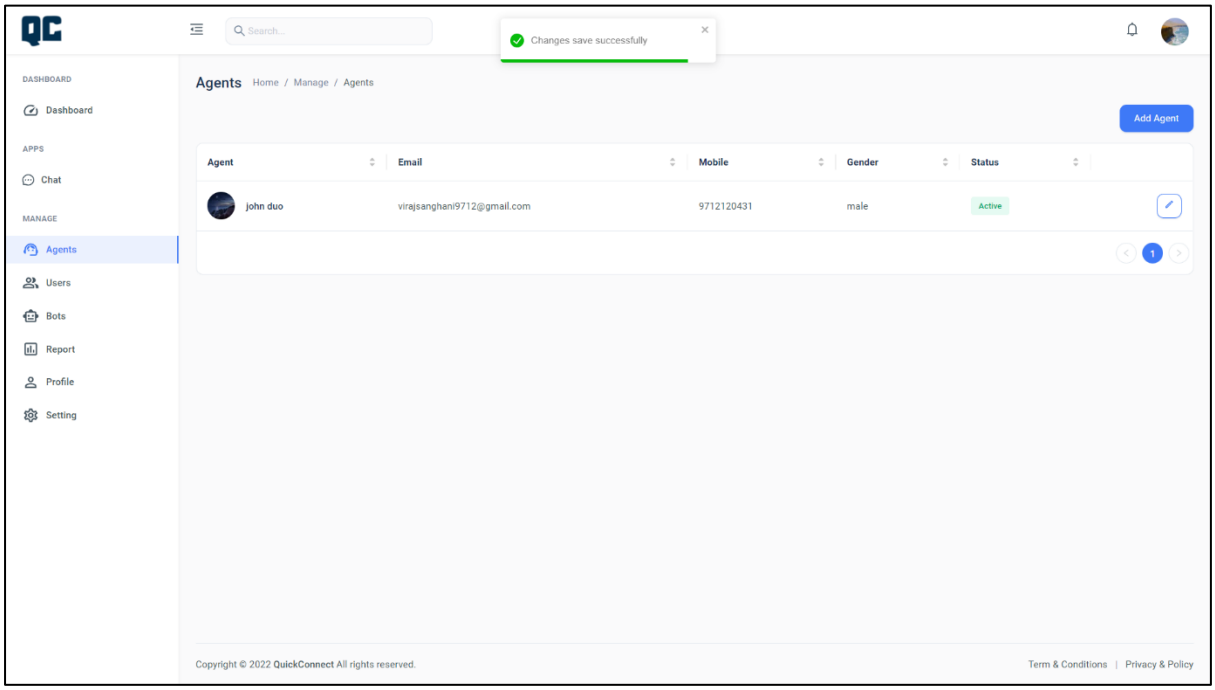
Agent Add:-



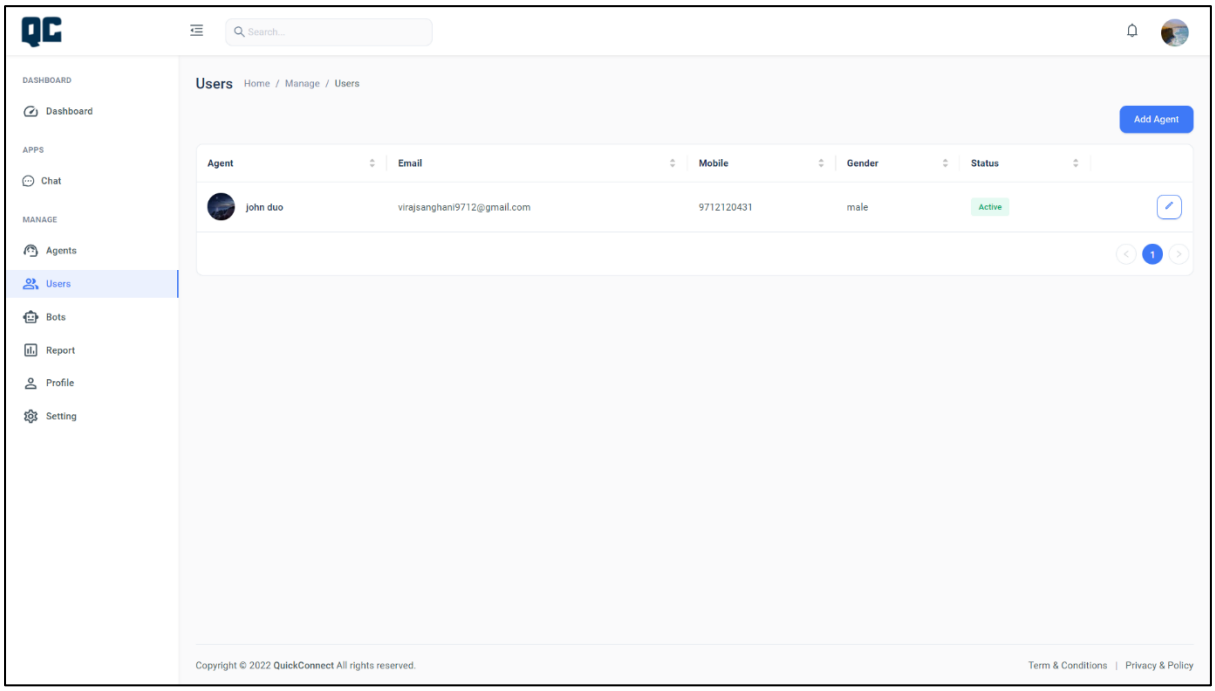
Agent Edit Profile:-



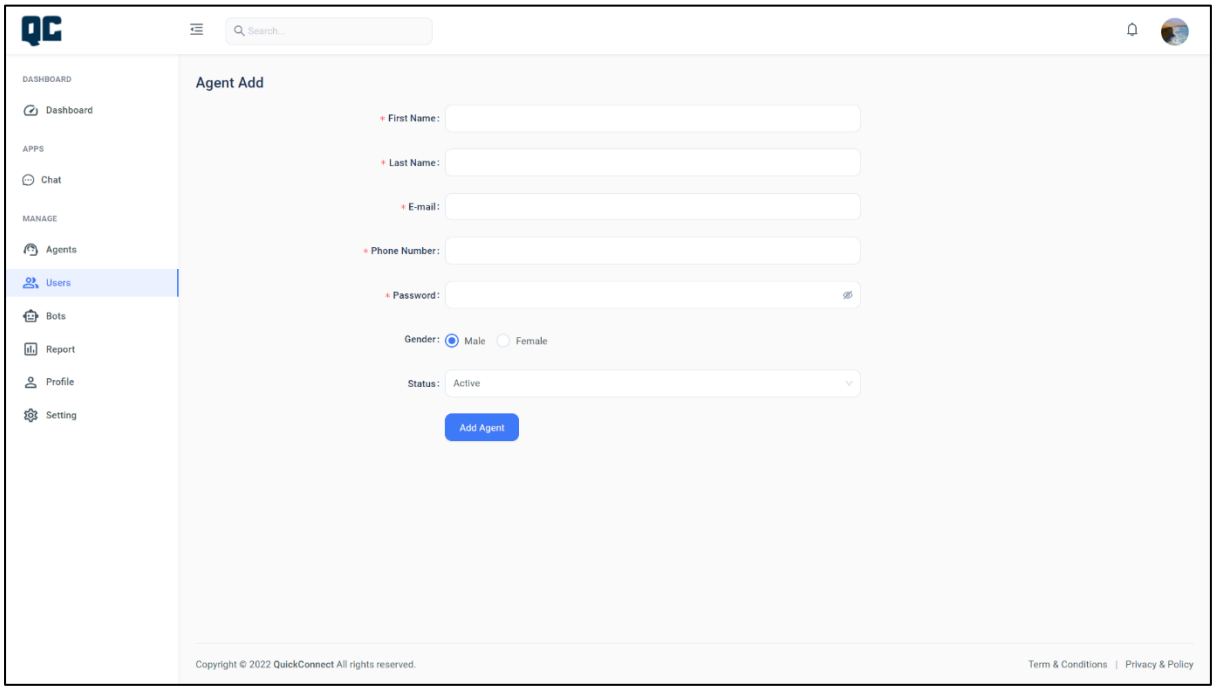
Agent Save Profile:-



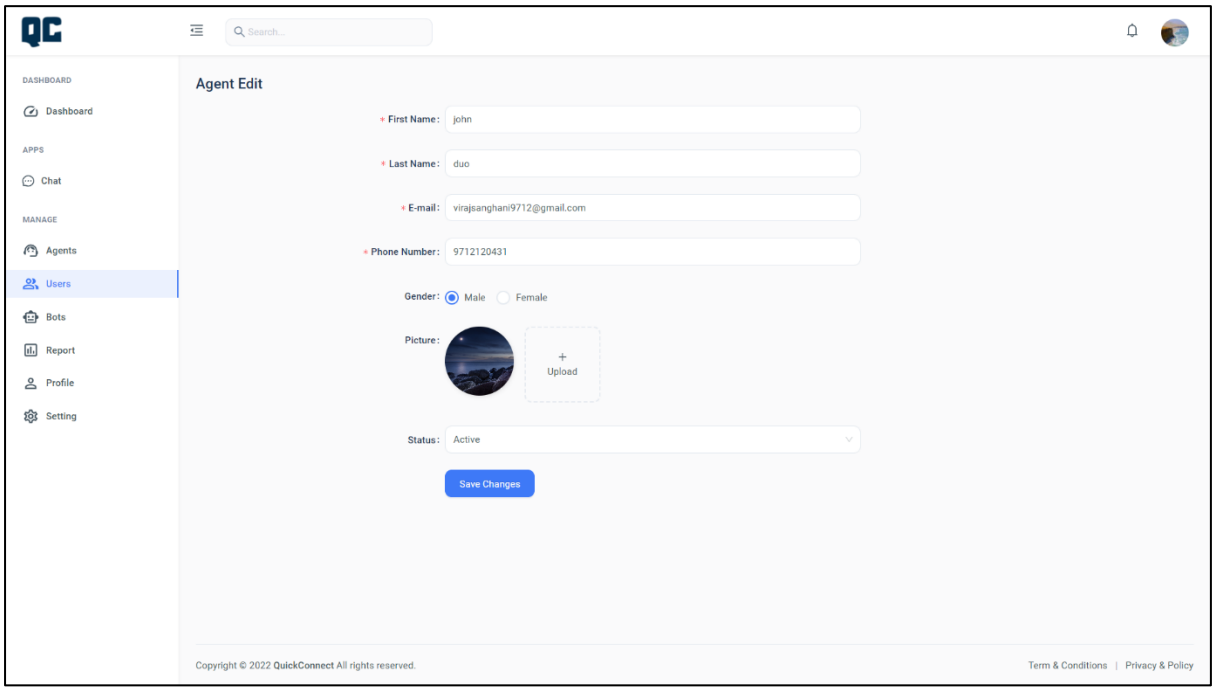
Users:-



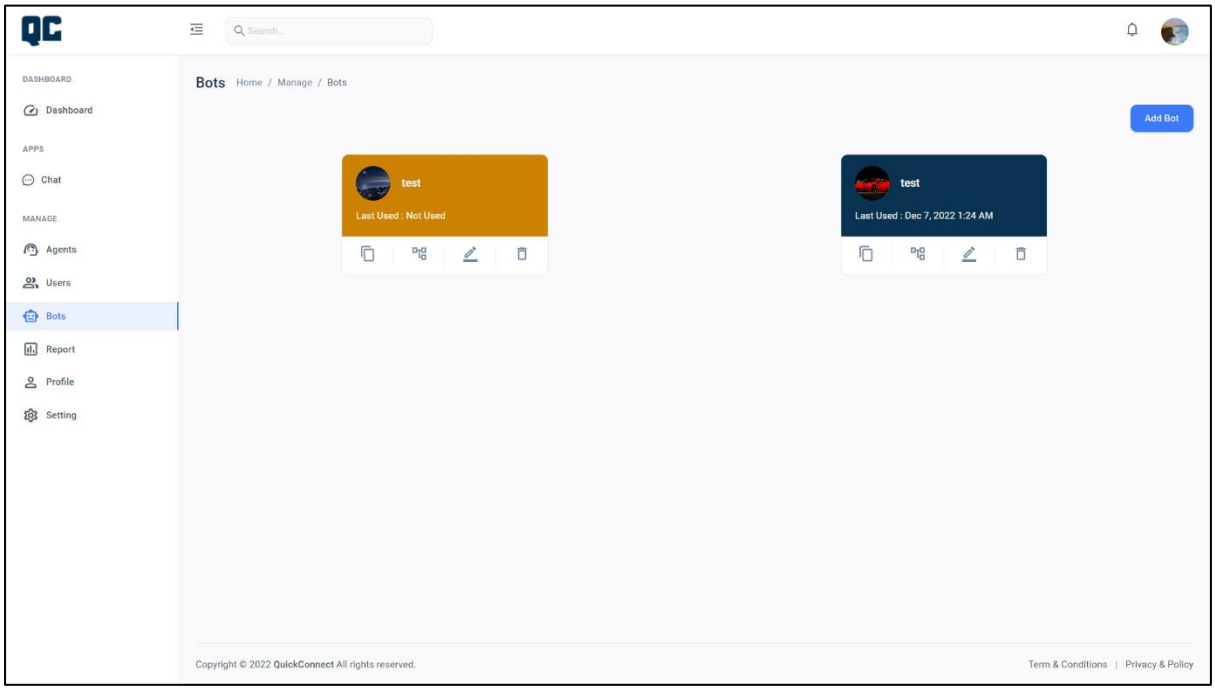
Add Agent :-



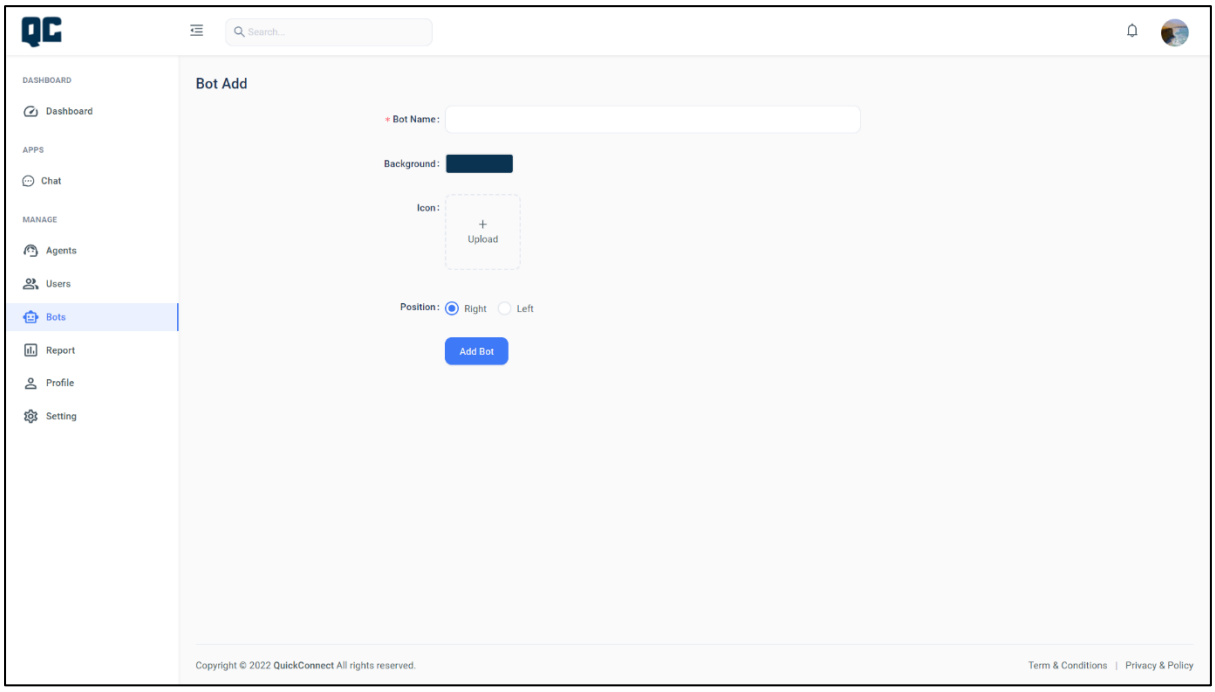
Agent Edit :-



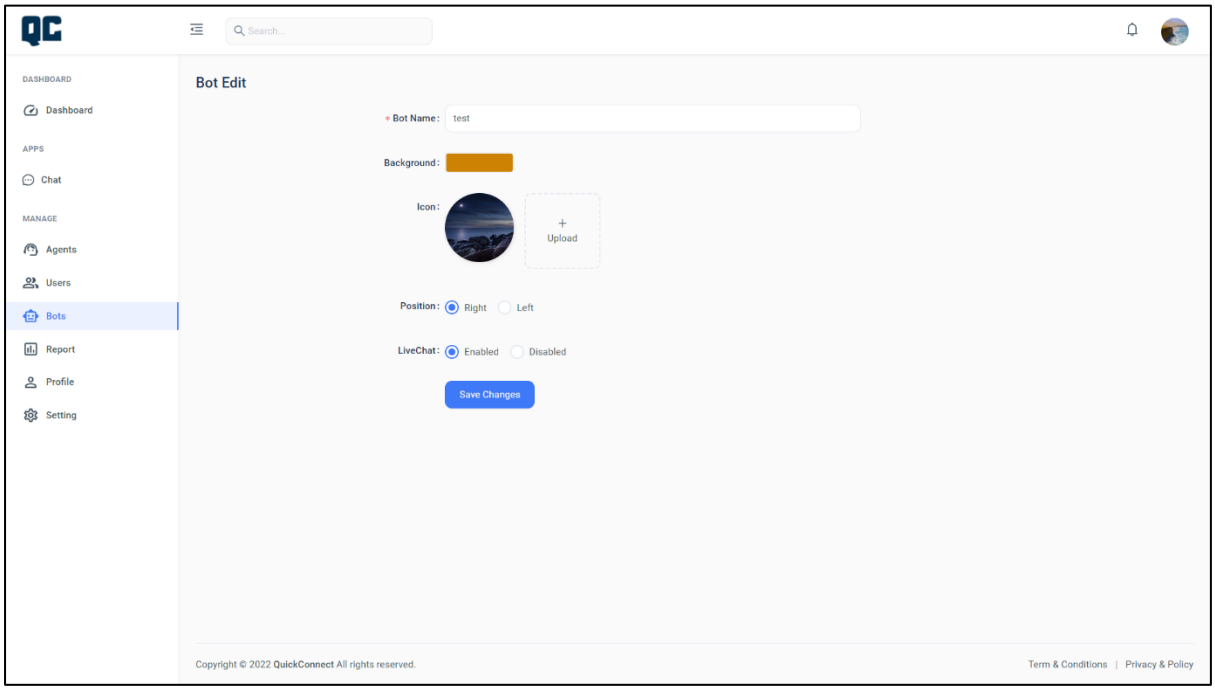
Bot List :-



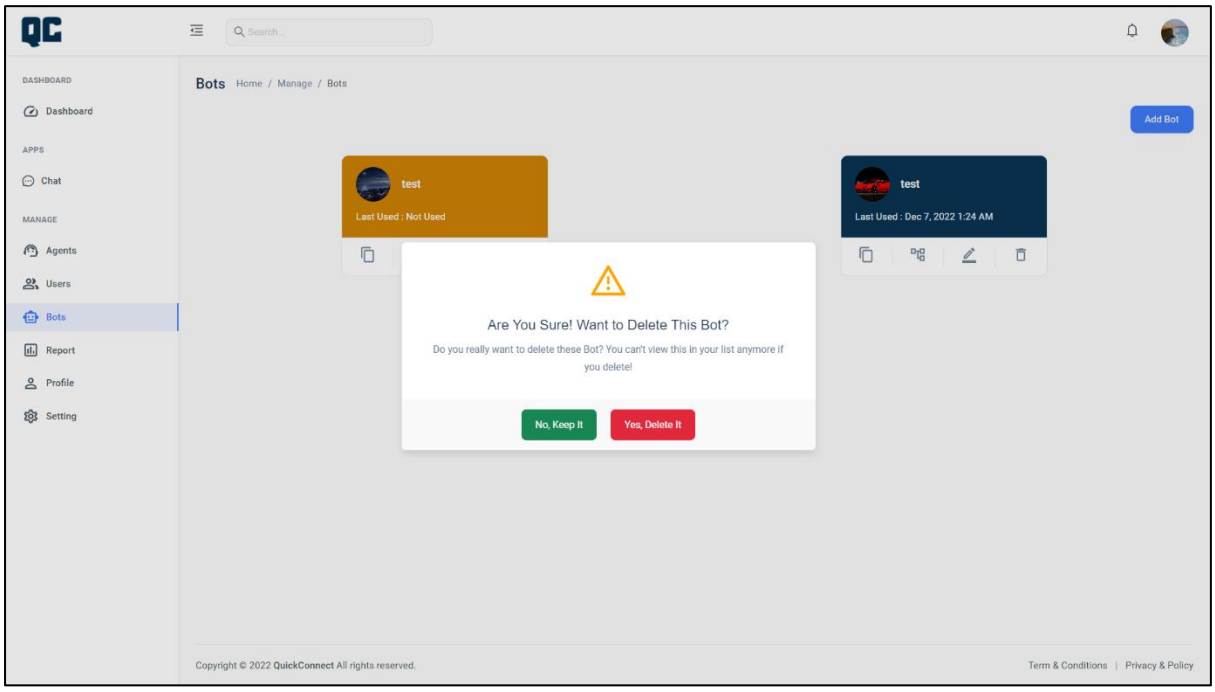
Add Bot :-



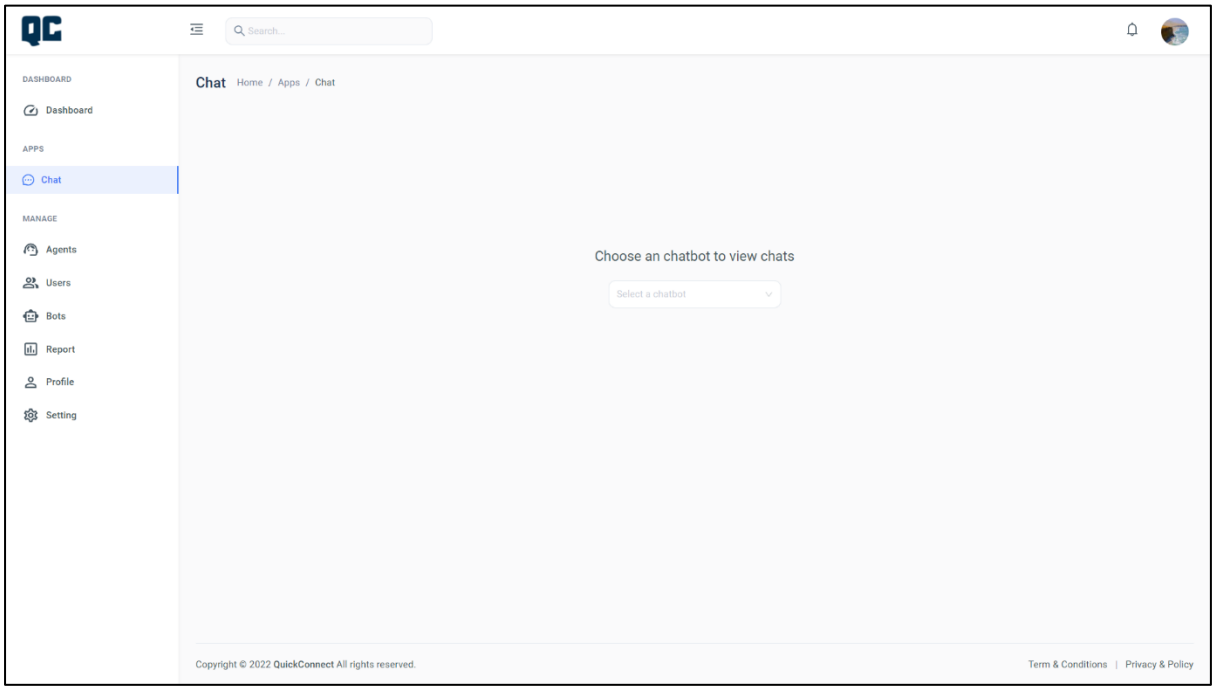
Edit Bot :-



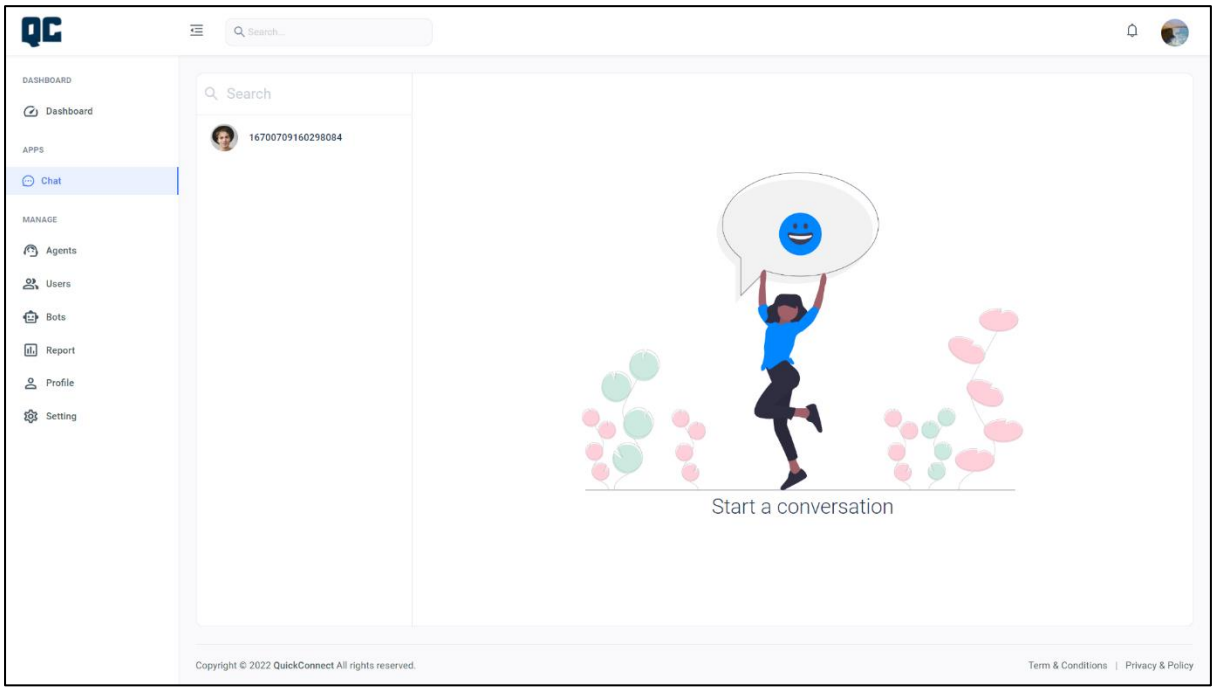
Delete Bot :-



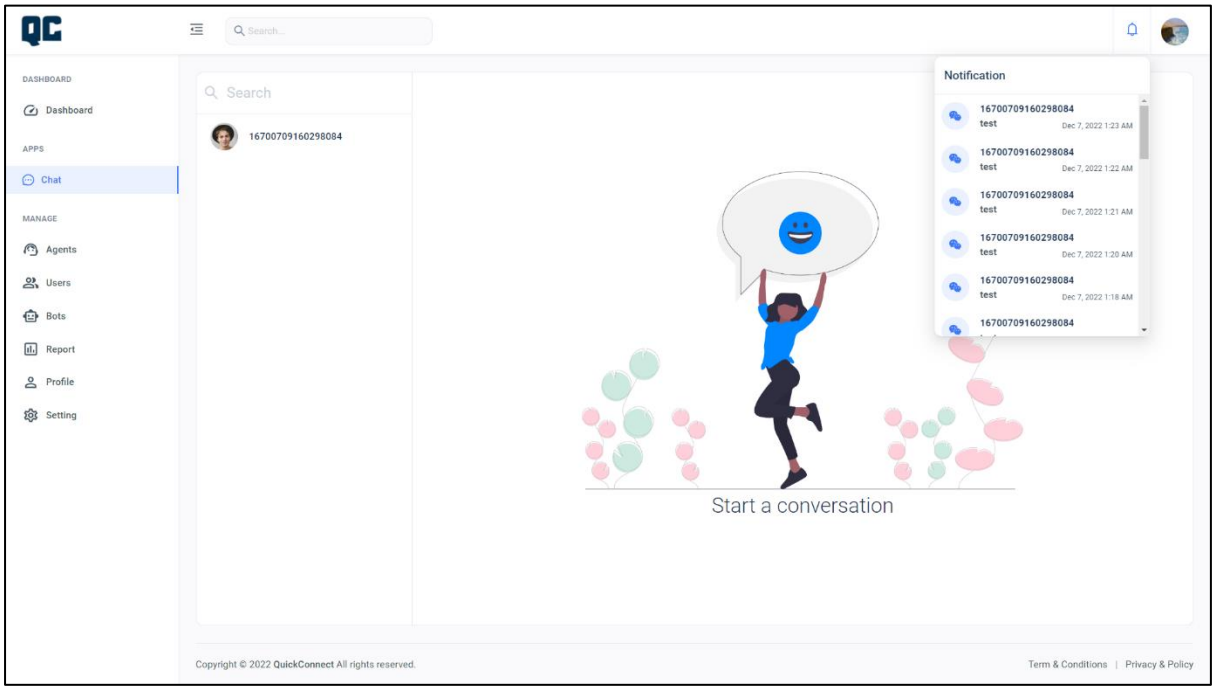
Choose Chatbot:-



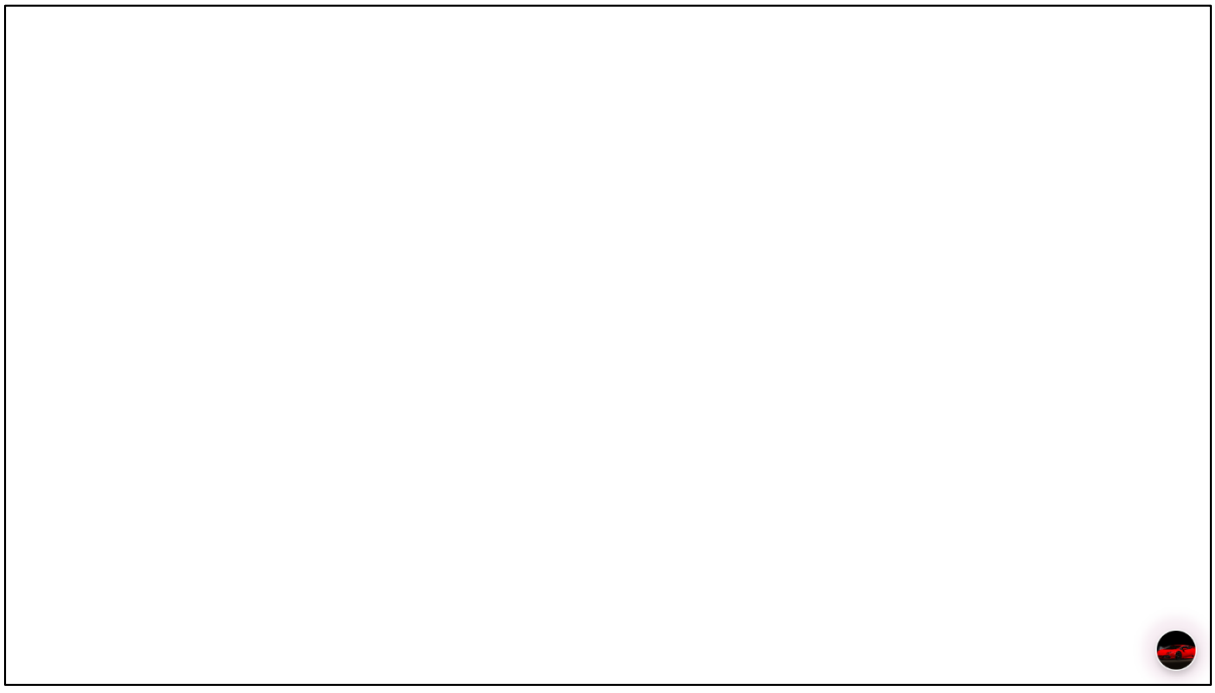
Show Chat:-



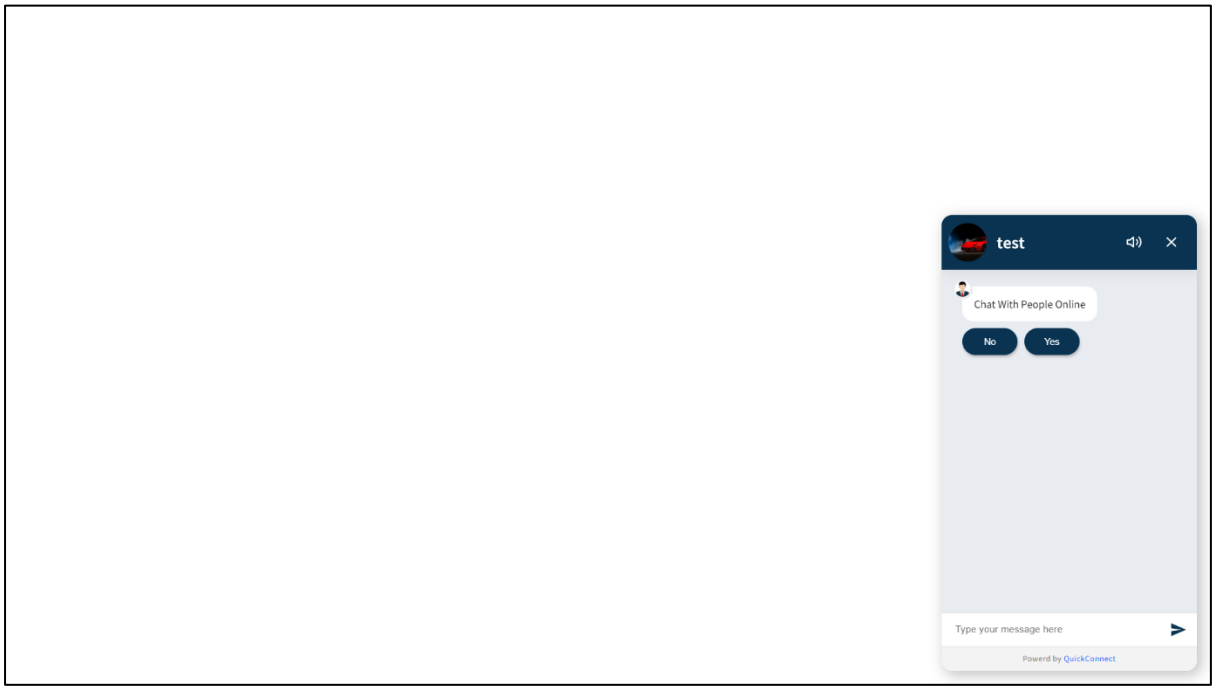
ChatBot Notification :-



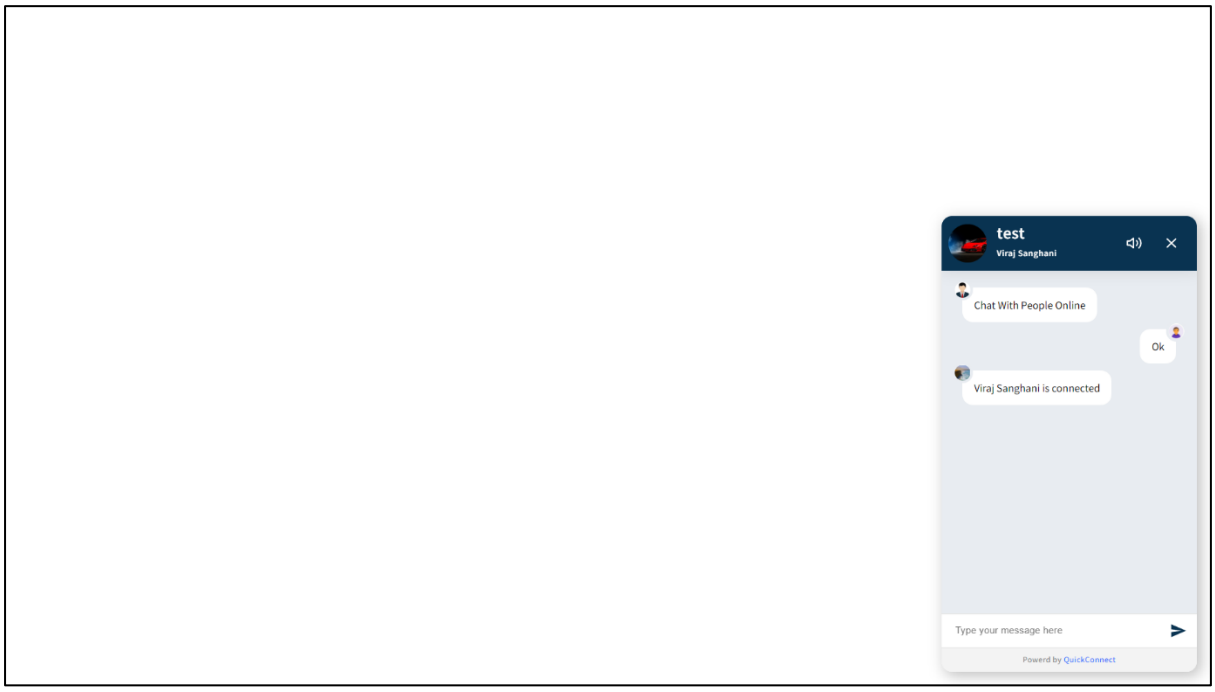
ChatBot Logo:-



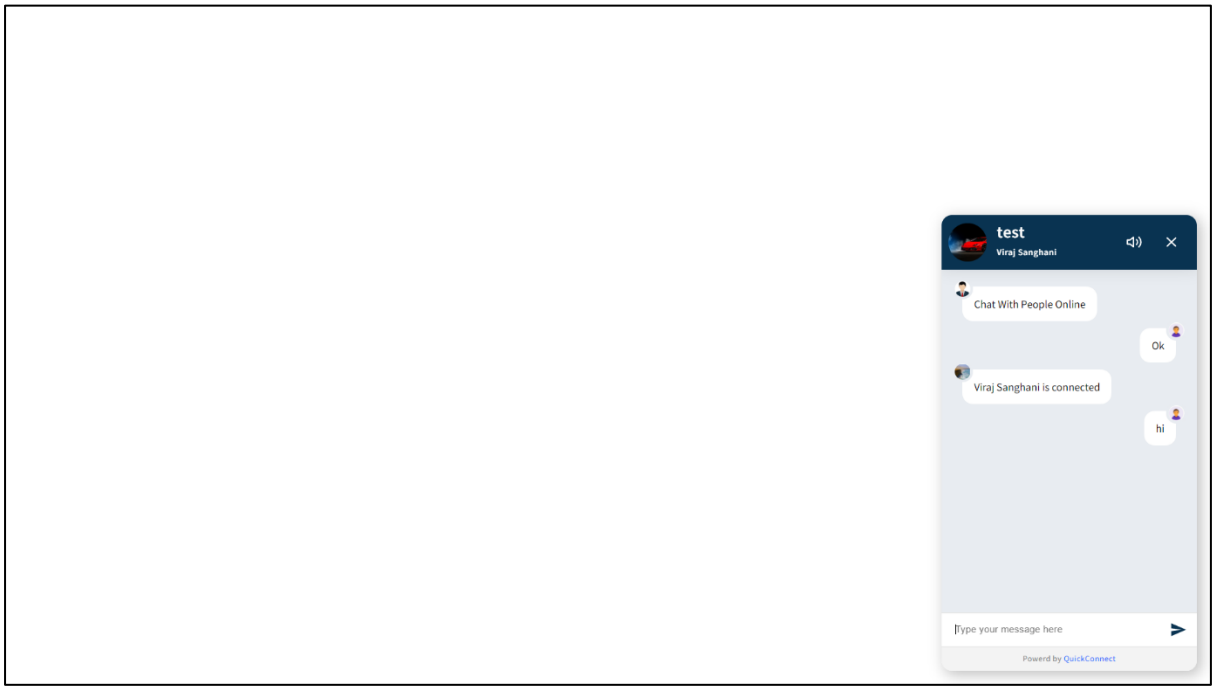
Live Agent Connected:-



Live Agent Connected:-



Chat With Agent:-



5. Agile Documentation

5.1 Agile Project Charter

Project Charter

Project Title:	QuickConnect	Project Guide:	Prof. Nirali R Sheth
Project Start Date:	22-Aug-2022	Project End Date:	06-Dec-2022

Business Need

Chatbot visual builder empowers you to create perfect chatbots quickly and with less coding. Upload conversational elements, and test them in real time to design engaging chatbot stories.

Project Scope	Deliverables
An all-in-one platform to build and launch conversational chatbots with less coding. A chatbot is software that simulates human-like conversations with users via text messages on chat. Its key task is to help users by providing answers to their questions.	It will benefit to; • POWER UP YOUR MARKETING Personalize your customer experience. • SUPERCHARGE YOUR SALES Scale your efforts without increasing employee headcount. • OPTIMIZE YOUR SUPPORT Keep your customer support going after hours.
Risks and Issues	Assumptions/Dependencies
A conversational Chatbot is not the same as a human agent, so it does not always understand a query. Its selection of answers may be limited, depending on the information it has uploaded. There may be times when your interactions seem "robotic."	Our support team will help you with ChatBot implementation and customization all along the line.

Milestone Schedule

Milestone	Target Completion Date	Actual Date
Project Starting Date	22-Aug-2022	22-Aug-2022
Design Approval	05-Sep-2022	05-Sep-2022
Improve	10-Oct-2022	10-Oct-2022
Project Ending Date	06-Dec-2022	06-Dec-2022

Project Team		Approval/Review Committee
Project Guide:	Prof. Nirali R Sheth	NA
Team Members:	Viraj Sanghani, Rahul Panchal, Masira Mansuri	

5.2.Agile Roadmap / Schedule

Quarter 1	Quarter 2	Quarter 3	Quarter 4
Completed to our application designing	Completed API	Completed admin panel	Finally completed our application
50%	70%	85%	100%

5.3.Agile Project Plan

Task	Start	End	Status
Analysis of Project Requirements	22/08	31/08	Complete
Designing chatbot	01/09	10/09	Complete
Designing Admin Auth	11/09	20/09	Complete
Registration and login API	21/09	30/09	Complete
Admin pages designing	01/10	20/10	Complete
Pending API	01/11	15/11	Complete
Testing and Implementation	16/11	20/11	Complete
Documentation	01/12	05/12	Complete

5.4.Agile User Story

User Story ID	As a	I want to	So that I can
1	User	Make UI simple and clean	So, that anyone can access easily
2	User	Live chat support required	So, that i can ask my own question
3	Admin	Manage All Agents, Users	For better management
4	Admin	Create and integrate multiple chatbot in different websites	So, that I can use it in multiple sites
5	Admin	customize color, position, etc.	So, I can modify according to site theme
6	Admin	Live chat notification	For Easy conversation

5.5.Agile Release Plan

Sprint	Task	Start	End	Duration	Status	Release Dt.
1	Analysis	22/08	31/08	10	Released	31/08
2	Designing chatbot	01/09	10/09	10	Released	10/09
3	Designing Admin Auth	11/09	20/09	10	Released	20/09
4	Registration and login API	21/09	30/09	10	Released	30/09
5	Admin pages designing	01/10	20/10	20	Released	20/10

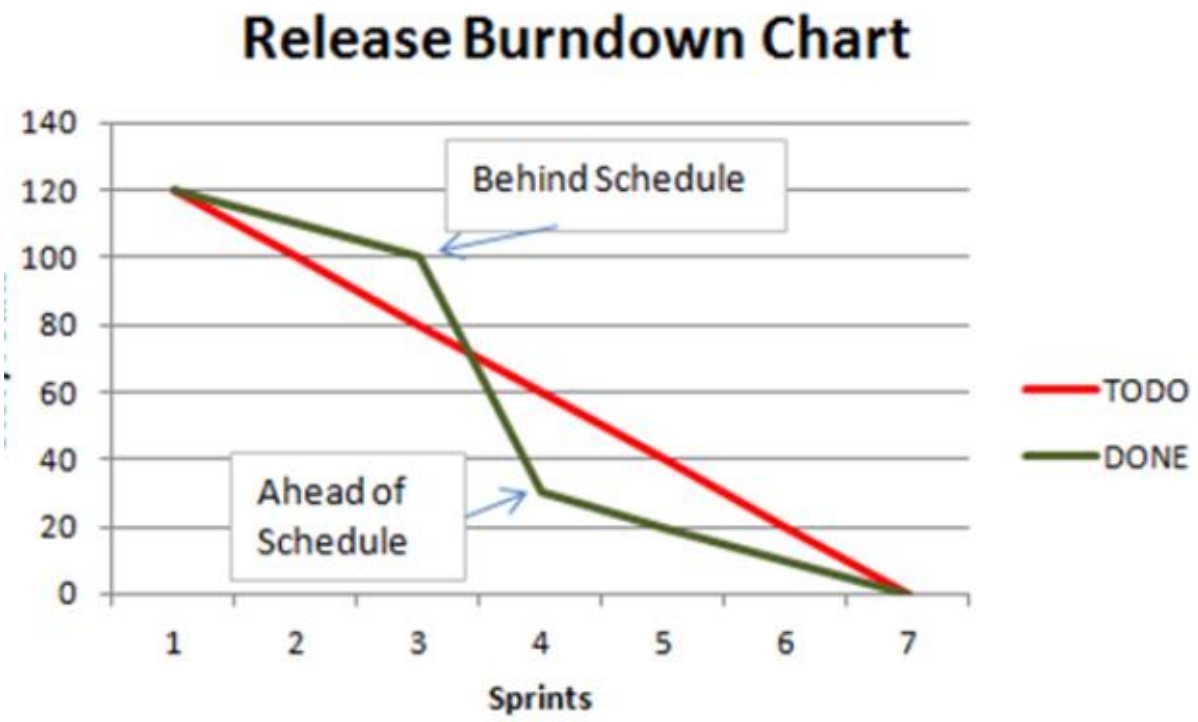
5.6.Agile Sprint Backlog

Task ID	Task	Estimated Effort(hrs)	Planned Effort(hrs)	Status
1	High Level Solution Design	50	50	Complete
2	Low Level Solution Design	50	50	Complete
3	Development	150	150	Complete
4	Live Chat	30	30	Complete
5	Testing	20	20	Complete
6	Bug Solved	60	60	Complete
7	Project Complete	20	20	Complete

5.7.Agile Test Plan

Test	Action	Expected Result	Actual Result	Pass?
1	User Register	Should get login page after register	Admin get login page after register	✓
2	User Login	Should get dashboard page after login	Admin get dashboard page after login	✓
3	Forgot Password	Should able to create new password	Admin able to create new password	✓
4	Manage Multiple Chatbots	Should able to manage chatbot	Admin able to manage chatbot	✓
5	Manage Agents	Should able to manage agents	Admin able to manage agents	✓
6	Manage Users	Should able to manage users	Admin able to manage users	✓
7	Admin/Agent can logout	Should logout from the system	Can logout from the system	✓
8	Integrate Chatbot	Admin should integrate chatbot	Admin can integrate chatbot	✓
9	Request For Live Chat	Customer should request for live chat	Customer can request for live chat	✓
10	Live Chat	Agents able to connect in live chat	Can able to connect in live chat	✓
11	Reports	Admin should able check analytics of any chatbot	Admin can able check analytics of any chatbot	✓

5.8 Agile Burn Charts



6. Proposed Enhancements

Since this Project was started with very little knowledge about the selected technology, we came to know about the enhancement capability during the process of building it. Some of the knowledge we had increased during this project is listed below.

- How to make views in ReactJS
- Write backend code in NodeJS
- Implement Html & CSS for designing
- Implementing JavaScript in entire project
- Generating design using ReactJS
- Making Models and Implementation of MongoDB for storing Data
- Time Management
- Understanding of Team Work
- Understanding of Error Solving

7. Conclusion

It was wonderful and learning experience for us while working on this project. This project took us through the various phases of project development and gave us real time insight into the world of software development. The joy of working an thrill involved while tackling the various problems and challenges gave us a feel of developer industry.

While developing this project we have learned a lot about chat application, we have also learn how to make it user friendly (easy to use and handle) by hiding the complicated parts of it from the users.

During the development process we studied carefully and understood the criteria for making software more demanding, we also realized the importance of maintaining a minimal margin for error.

It was due to this project we came to know how professional software are designed.

We enjoyed each and every bit of work we had put into this project. This project is future extendable.

We would also like to give our gratitude to our project guide and friends who helped us and supported us to complete our project successfully.

8. Bibliography

Website:

https://draw.io.in

https://nodejs.org

https://reactjs.org

https://mongodb.com/home

https://expressjs.com

https://socket.io