

# **CHAT NOVO**

## **MINOR PROJECT REPORT**

**SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR  
THE AWARD OF THE DEGREE OF**

## **BACHELOR OF TECHNOLOGY**

**(Information Technology)**



**Submitted by :**

Ekam Preet Singh (1921028/1905327)

Muskan Sharma (1921067/1905366)

Kunwar Arora (1921058/1905356)

**Submitted to :**

Prof. Pankaj Bhambri

Assistant Professor

**Department of Information Technology,**

**Guru Nanak Dev Engineering College,**

**Ludhiana-141006**

# Table of Contents

ABSTRACT . . . . .	i
ACKNOWLEDGEMENT . . . . .	ii
1 Introduction . . . . .	1
1.1 Introduction to Project . . . . .	1
1.2 Project Category . . . . .	1
1.3 Objectives . . . . .	1
1.4 Problem Formulation . . . . .	1
1.5 Identification of Need . . . . .	2
1.6 Existing System . . . . .	2
1.7 Proposed System . . . . .	2
2 Requirement Analysis and System Specification . . . . .	3
2.1 Feasibility Study . . . . .	3
2.1.1 Technical Feasibility: . . . . .	3
2.1.2 Economical Feasibility: . . . . .	3
2.1.3 Operational Feasibility: . . . . .	3
2.2 Software Requirement Specification Document . . . . .	4
2.2.1 Hardware: . . . . .	4
2.2.2 Software: . . . . .	4
2.3 Validation . . . . .	4
2.4 Expected hurdles . . . . .	4
2.5 SDLC Model to be used . . . . .	5
2.5.1 Planning: . . . . .	5
2.5.2 Analysis: . . . . .	5
2.5.3 Designing: . . . . .	5
2.5.4 Implementation: . . . . .	5
2.5.5 Testing: . . . . .	7
2.5.6 Maintenance: . . . . .	8
3 System Design . . . . .	9
3.1 Design Approach . . . . .	9
3.2 Detail Design . . . . .	9
3.2.1 Front End : . . . . .	9
3.2.2 Back End : . . . . .	10
3.3 System Design . . . . .	10
3.4 Database Design . . . . .	12
3.5 Methodology . . . . .	14
4 Implementation, Testing, and Maintenance . . . . .	15
4.1 Introduction to Languages, IDE's, Tools and Technologies used for Implementation	15
4.2 Coding standards of Language used . . . . .	15

4.3	Project Scheduling using PERT charts . . . . .	16
4.4	Testing Techniques and Test Plans . . . . .	16
5	Results and Discussions . . . . .	<b>17</b>
5.1	User Interface Representation . . . . .	17
5.2	Snapshots of system with brief detail of each . . . . .	17
5.3	Back Ends Representation . . . . .	20
5.3.1	Snapshots of Database Tables . . . . .	20
6	Conclusion and Future Scope . . . . .	<b>21</b>
6.1	Conclusion . . . . .	21
6.2	Future Scope . . . . .	21
	References/Bibliography . . . . .	<b>22</b>

## ABSTRACT

Chat application is a feature or a program on the Internet to communicate directly among Internet users who are online or who were equally using the internet. Chat applications allow users to communicate even though from a great distance. Therefore, this chat application must be real-time and multi platform to be used by many users. It is a method of using technology to bring people and ideas together despite of the geographical barriers. The technology has been available for years but the acceptance was quite recent.

## **ACKNOWLEDGEMENT**

I/WE are highly grateful to the Dr. Sehijpal Singh, Principal, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out the minor project work at college.

The constant guidance and encouragement received from Dr. K.S. Mann H.O.D. IT Department, GNDEC Ludhiana has been of great help in carrying out the project work and is acknowledged with reverential thanks.

I/WE would like to express a deep sense of gratitude and thanks profusely to Prof. Pankaj Bhambri, without his wise counsel and able guidance, it would have been impossible to complete the project in this manner.

I/WE express gratitude to other faculty members of Information Technology department of GNDEC for their intellectual support throughout the course of this work.

Finally, I/WE are indebted to all whosoever have contributed in this report work.

**Ekam Preet Singh**

**Muskan Sharma**

**Kunwar Arora**

# 1 Introduction

## 1.1 Introduction to Project

- A chat-box is a software application used to conduct an on-line chat conversation via text. A chat-box is a type of software that can help customers by automating conversations and interact with them through messaging platforms. Today Developers around the world are making efforts to enhance user experience of using application as well as to enhance the developer's workflow of designing applications to deliver projects and roll out change requests under strict timeline.
- Stacks can be used to build web applications in the shortest span of time. The stacks used in web development are basically the response of software engineers to current demands. They have essentially adopted pre-existing frameworks (including JavaScript) to make their lives easier. While there are many, MEAN and MERN are just two of the popular stacks that have evolved out of JavaScript. Both stacks are made up of open source components and offer an end-to-end framework for building comprehensive web apps that enable browsers to connect with databases. The common theme between the two is JavaScript and this is also the key benefit of using either stack. One can basically avoid any syntax errors or any confusion by just coding in one programming language, JavaScript.
- Another advantage of building web projects with MERN is the fact that one can benefit from its enhanced flexibility. In order to understand MERN stack, we need to understand the four components that make up the MERN stack, namely – MongoDB, Express.js, React and Node.js. This project is focusing on creating a chat bot to be used by students to get their queries responded easily from the college website. The College Enquiry Chat bot has the capacity to make friendly conversations; respond the course and questions. Project URL: <https://mern-gndec-chat-bot.herokuapp.com/>

## 1.2 Project Category

Our project comes under the category of web development

## 1.3 Objectives

- To develop an instant messaging solution to enable users to seamlessly communicate with each other.
- The main purpose of this project is to provide multi chatting functionality through network.

## 1.4 Problem Formulation

Notice Board and circulars provide means for communication among students but these are one-way mechanisms and they do not provide an easy way to carry on a real-time conversation

or discussion with people involved. We have proposed a chatting application for one to one and one to many communications which is fast, streamlined and secure

## **1.5 Identification of Need**

Identifying potential problems before the start of a project can save significant amounts of time and money. Problem analysis is one of the most critical stages of project planning because this stage helps to guide all subsequent analysis and decision making. The needs are checked in the initial step so that no problem can arise during the implementation phase. Observation and information gathering represents two processes. On the other side, gathering information can highlight the processes needed to execute the project. The software and hardware needs are analysed for the project

## **1.6 Existing System**

Windows 11,8GB RAM,512GB SSD hard disk.

## **1.7 Proposed System**

**Feature to communicate :**

- Communicate with peers as well as with faculty members

**Secure communication :**

- Password authentication with hashing of passwords when registering

**Responsive View :**

- Can be accessed from any device irrespective of it's screen size

## 2 Requirement Analysis and System Specification

### 2.1 Feasibility Study

This chapter describes all the feasibility's that come as questions to both the developers and other users during the development of software. The chapter contains technical feasibility, economic feasibility and operational feasibility.

#### 2.1.1 Technical Feasibility:

The tools and technology that were used in building of this chat application:

- Editor: Visual Studio 2017
- Cloud Database: MongoDB
- Programming Language: React JS, Node JS Express JS
- Hosting Platform: Heroku

These technologies mentioned above are completely free for students. Most of my time to the resources required for this project are: Deployment machine's. Any regular laptop/PC with a minimum ram of 4GB and a decent GUI can be used for the deployment of this chat application.

#### 2.1.2 Economical Feasibility:

The resources required for this project are: Deployment machine's. Any regular laptop/PC with a Minimum Ram of 4GB and a decent GUI can be used for the deployment of this e-commerce. Secondly, Technical tools and software: As mentioned previously, the tools needed to develop this software are available to developers at no charge. Heroku Provides free hosting and domain selection for everyone to host their respective application. Hosting and storing data on cloud application ensure to be accessed with the application from around the world

#### 2.1.3 Operational Feasibility:

Operational feasibility is a measure of how well the proposed system will solve the problems, and takes advantage of the opportunities that are identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. Operational feasibility reviews the willingness of the organization to support the proposed system. This is probably the most difficult of the possibilities to gauge. In order to determine this feasibility, it is essential to understand the management commitment towards the proposed project. The project is providing the adequate throughput and response time, make the correct and maximum use of all the available resources.



## **2.2 Software Requirement Specification Document**

### **2.2.1 Hardware:**

- Recommended Processor : Intel core i3 or above
- Recommended Processor Speed : 2GHz CPU
- Recommended RAM : 4 GB or above
- Recommended OS: Windows 7 or above, MacOS, Linux

### **2.2.2 Software:**

- VSCode is a text editor and source code editor that provides an environment for developing
- MongoDB 5.x with native time series collections optimized for IoT and financial apps.
- Node.js v17 / NPM 6.0.0 Express JS 4.17.3
- React JS 18.0.0
- Express JS 4.17.3
- Cloudinary API
- Heroku Inc.

## **2.3 Validation**

Today, small to large scale enterprises understand the need and importance of proper data storage, across all industries. While offering the best products, an effective user interface and secure messaging options may be your first priorities, it's time to take the next step and consider Cloud Hosting for a chat application is to give an extra edge. To be able to smoothly run all aspects of operations, one of the key aspects is easily accessible data. With Cloud Hosting, you can improve the accessibility of your data for all users, be it your students, faculty members or any person with a title else who may require it.

## **2.4 Expected hurdles**

Hosting along with instant messaging without lag from User to API to Database and then a response resulting in the performance of this application

## 2.5 SDLC Model to be used

### 2.5.1 Planning:

We made two applications - the client application and the server application. The client application runs on the user's web browser whereas the server application runs on any hosting servers on the network.

### 2.5.2 Analysis:

We analysed our planning and thinking how to implement instant messaging and designing a database to store each user, chat and group in MongoDB

### 2.5.3 Designing:

In the designing part, we designed the navigation view starting with the home page which contains a Search bar and a name of the website. Our chat application also contains a notification bar and have the option of the profile view.

### 2.5.4 Implementation:

For implementation of our project we made a cluster of files. Here we displayed only one file i.e. Login Page:

```
1 import { Button } from "@chakra-ui/button";
2 import { FormControl, FormLabel } from "@chakra-ui/form-control";
3 import { Input, InputGroup, InputRightElement } from "@chakra-ui/input";
4 import { VStack } from "@chakra-ui/layout";
5 import { useState } from "react";
6 import axios from "axios";
7 import { useToast } from "@chakra-ui/react";
8 import { useHistory } from "react-router-dom";
9
10 const Login = () => {
11   const [show, setShow] = useState(false);
12   const handleClick = () => setShow(!show);
13   const toast = useToast();
14   const [email, setEmail] = useState();
15   const [password, setPassword] = useState();
16   const [loading, setLoading] = useState(false);
17
18   const history = useHistory();
19
20   const submitHandler = async () => {
21     setLoading(true);
22     if (!email || !password) {
23       toast({
```

```

24     title: "Please Fill all the Feilds",
25     status: "warning",
26     duration: 5000,
27     isClosable: true,
28     position: "bottom",
29   });
30   setLoading(false);
31   return;
32 }
33
34 // console.log(email, password);
35 try {
36   const config = {
37     headers: {
38       "Content-type": "application/json",
39     },
40   };
41
42   const { data } = await axios.post(
43     "/api/user/login",
44     { email, password },
45     config
46   );
47
48   // console.log(JSON.stringify(data));
49   toast({
50     title: "Login Successful",
51     status: "success",
52     duration: 5000,
53     isClosable: true,
54     position: "bottom",
55   });
56   localStorage.setItem("userInfo", JSON.stringify(data));
57   setLoading(false);
58   history.push("/chats");
59 } catch (error) {
60   toast({
61     title: "Error Occured!",
62     description: error.response.data.message,
63     status: "error",
64     duration: 5000,
65     isClosable: true,
66     position: "bottom",
67   });
68   setLoading(false);
69 }
70 };
71
72 return (
73   <VStack spacing="10px">
74     <FormControl id="email" isRequired>
75       <FormLabel>Email Address</FormLabel>
76       <Input

```

```

77     value={email}
78     type="email"
79     placeholder="Enter Your Email Address"
80     onChange={(e) => setEmail(e.target.value)}}
81   />
82 </FormControl>
83 <FormControl id="password" isRequired>
84   <FormLabel>Password</FormLabel>
85   <InputGroup size="md">
86     <Input
87       value={password}
88       onChange={(e) => setPassword(e.target.value)}
89       type={show ? "text" : "password"}
90       placeholder="Enter password"
91     />
92     <InputRightElement width="4.5rem">
93       <Button h="1.75rem" size="sm" onClick={handleClick}>
94         {show ? "Hide" : "Show"}
95       </Button>
96     </InputRightElement>
97   </InputGroup>
98 </FormControl>
99 <Button
100   colorScheme="blue"
101   width="100%"
102   style={{ marginTop: 15 }}
103   onClick={submitHandler}
104   isLoading={loading}
105 >
106   Login
107 </Button>
108 <Button
109   variant="solid"
110   colorScheme="red"
111   width="100%"
112   onClick={() => {
113     setEmail("guest@example.com");
114     setPassword("123456");
115   }}
116 >
117   Get Guest User Credentials
118 </Button>
119 </VStack>
120 );
121 };
122 export default Login;

```

### 2.5.5 Testing:

We tested our website many times which meant by operating it on different devices for compatibility and also increases its load by increasing number of users and checking the progress in real-time.

### **2.5.6 Maintenance:**

We try to upgrade our web server and website from time to time by adding new features and making it more dynamic. It's easy to maintain and update the website.

## 3 System Design

System design is the solution of a "how to approach to the creation of the new system". It is composed of several steps. It facilitates the understanding and provides the procedural details necessary for implementation of the system recommended in the feasibility study. Emphasis is given on translating the performance requirements into design specification.

Design goes through logical and physical stages of development. Logical design reviews the present physical system; prepares input and output specification; make editing; security and control specification; details the implementation plan, and prepare logical design walk through. The physical design maps out the details of the physical system; plans the system implementation plan and specifies hardware and software. System design translates the system requirement into the ways of the system as recommended in the feasibility study. Thus, the system design is the translation from user-oriented document to a programmer or a database personal oriented document.

### 3.1 Design Approach

Cloud Hosting is basically object oriented. We had created different classes using NodeJS. In this object of the classes are created and called. When the website is hosted these functions are called in the back-end is made to run on cloud.

### 3.2 Detail Design

#### 3.2.1 Front End :

Having separated the server-side from the client side, SPA's dynamically fetch data from the API as the user is browsing the site, avoiding to refresh the whole page whenever the user has filled in a form or navigated to another part of the site.

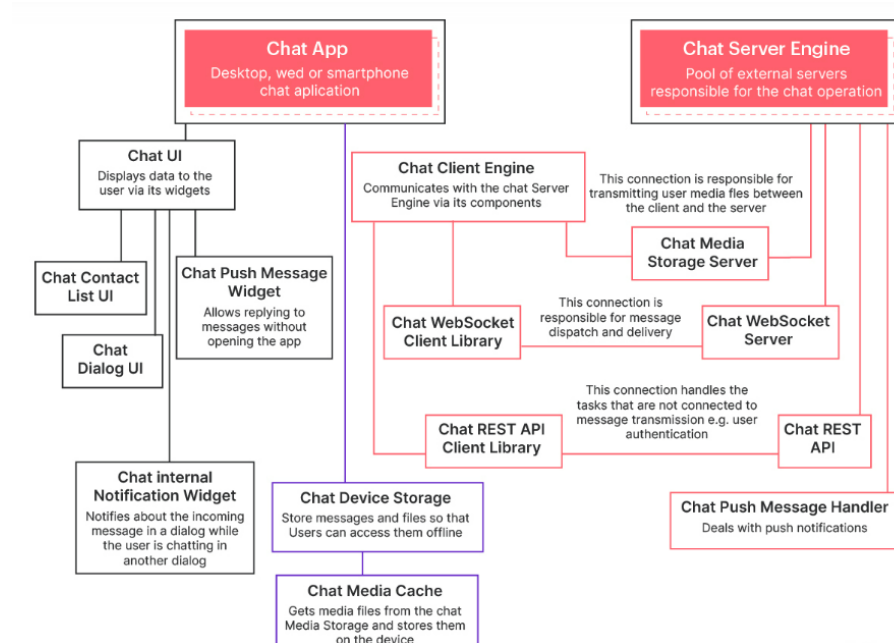
- **User Registration:** User must be able to register for the application through an Email, Username and Password. The user's email will be the unique identifier of his/her account on Chat Application.
- **Send Message :** User should be able to send instant message to any contact on his/her Chat Application contact list.
- **Broadcast Message:** User should be able to create groups of contacts. User should be able to broadcast messages to these groups.

### 3.2.2 Back End :

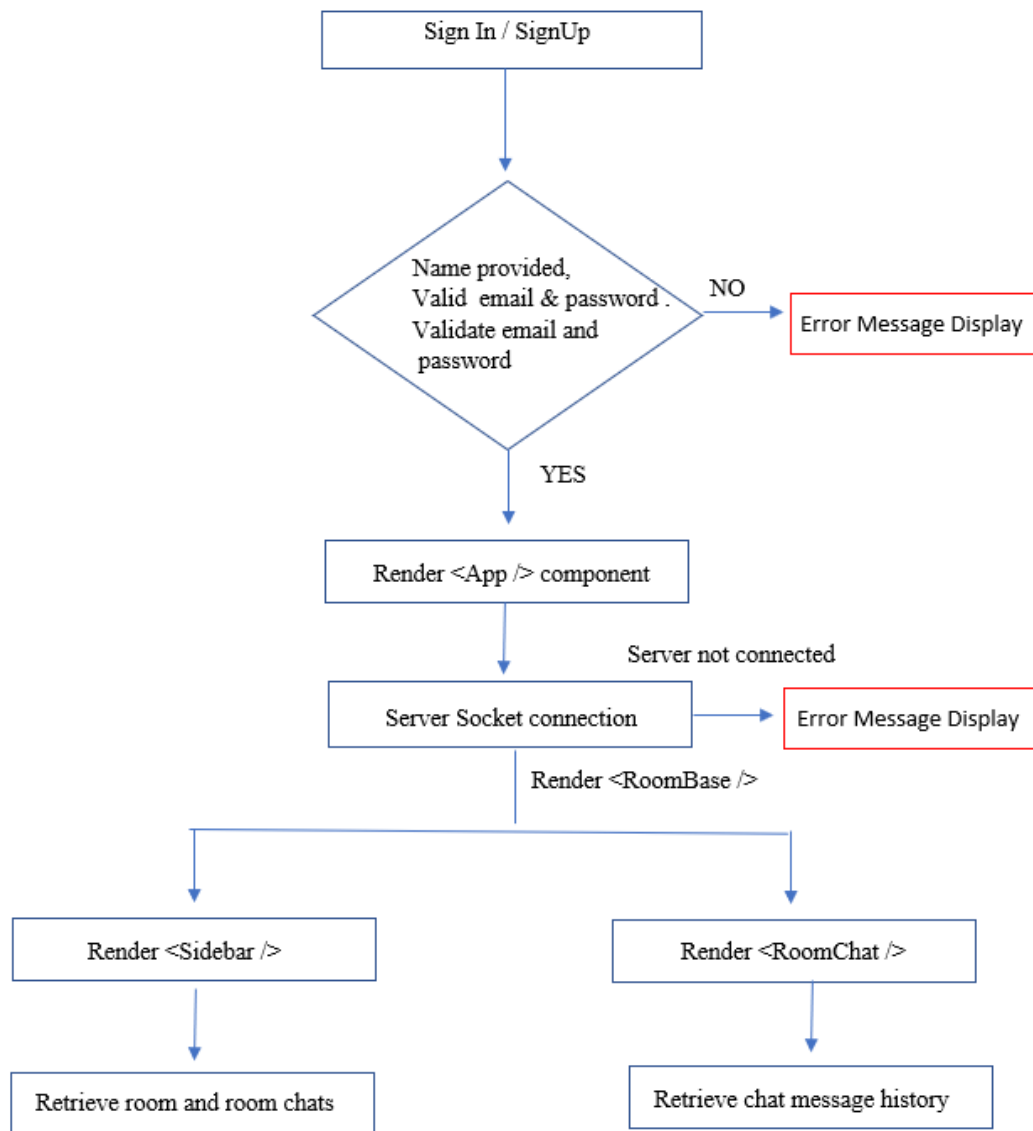
The "back end" refers to the logic and data layers running on the server side. In our case, the back end makes sure that the data introduced through the client application (the front end), is valid. we have to make sure that all the requests we receive are first verified by the server: the requested URL is supported, the user has the appropriate permissions,etc.

- **Express:** A Node.js framework which makes web development fast. It abstracts most of the complexity behind the web server and acts as an HTTP route handler.
- **Mongoose:** A MongoDB high-level library. By using objects as database models,it handles inserts, updates, and deletes, as well as the validation for each of its fields.
- **Node.js:** Node.js provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent
- **Robustness:** In case user's system crashes, a backup of their chat history must be stored on remote database servers to enable recover-ability.
- **Performance:** Application must be lightweight and must send messages instantly
- **Dependencies:** The third-party package or modules installed using npm are specified in this segment.

### 3.3 System Design



*Figure 3.1: System Design*



**Figure 3.2:** Flowchart



### 3.4 Database Design

```
const mongoose = require("mongoose");

const chatModel = mongoose.Schema(
  {
    chatName: { type: String, trim: true },
    isGroupChat: { type: Boolean, default: false },
    users: [{ type: mongoose.Schema.Types.ObjectId, ref: "User" }],
    latestMessage: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Message",
    },
    groupAdmin: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
  },
  { timestamps: true }
);

const Chat = mongoose.model("Chat", chatModel);

module.exports = Chat;
```

*Figure 3.3: Chat Model*

```
const mongoose = require("mongoose");

const messageSchema = mongoose.Schema(
  {
    sender: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
    content: { type: String, trim: true },
    chat: { type: mongoose.Schema.Types.ObjectId, ref: "Chat" },
    readBy: [{ type: mongoose.Schema.Types.ObjectId, ref: "User" }],
    ipAddress: { type: String },
  },
  { timestamps: true }
);

const Message = mongoose.model("Message", messageSchema);

module.exports = Message;
```

*Figure 3.4: Message Model*

```

const mongoose = require("mongoose");
const bcrypt = require("bcryptjs");

const userSchema = mongoose.Schema(
  {
    name: { type: "String", required: true },
    email: { type: "String", unique: true, required: true },
    password: { type: "String", required: true },
    pic: {
      type: "String",
      required: true,
      default:
        "https://icon-library.com/images/anonymous-avatar-icon/anonymous-avatar-icon-25.jpg",
    },
    isAdmin: {
      type: Boolean,
      required: true,
      default: false,
    },
  },
  { timestamps: true }
);

userSchema.methods.matchPassword = async function (enteredPassword) {
  return await bcrypt.compare(enteredPassword, this.password);
};

userSchema.pre("save", async function (next) {
  if (!this.isModified) {
    next();
  }

  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
});

const User = mongoose.model("User", userSchema);

module.exports = User;

```

*Figure 3.5: User Model*

### 3.5 Methodology

For this project we will be referring to the waterfall model of software engineering.

1. Phase one : System Planning
2. Phase two : Problem Analysis
3. Phase three : System Design
4. Phase four : System Implementation
5. Phase five : System Testing
6. Phase six : Operation and Maintenance

- Step 1: In this phase, we will plan on design of the UI for the web app. We will research on how we can implement various technologies and connect to work together. The decision of which language will be used for the front-end and back-end will be done in this phase. Also, the selection of database will be done in this phase. We have done our earlier research on this but will look into more robust technologies and working methods.
- Step 2: In this phase we will work implementing the decided features in our app. We will focus on making an easy-to-use UI/UX and robust back-end so that it will break and will be able to handle errors on its own. In phase we will create our minimum viable product in this phase. Which will be ready for testing and feedback purpose in the next phase.
- Step 3: In this phase we will test our web app for various conditions and environments and will work on its improvement if needed.
- Step 4: In this phase we will deploy our web app to the server so that it can used by the faculty and students.

## 4 Implementation, Testing, and Maintenance

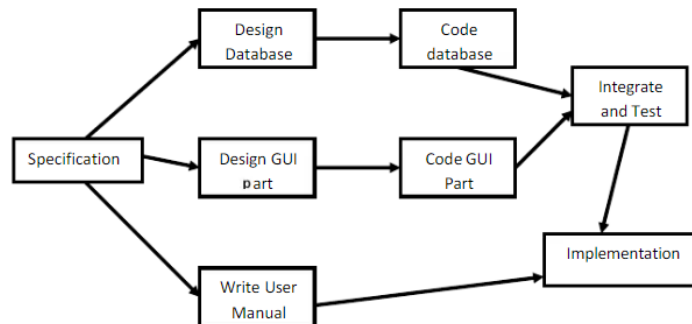
### 4.1 Introduction to Languages, IDE's, Tools and Technologies used for Implementation

- **Client-Side Rendering:** By default, React app will be client-side rendered. This means basically, all of the source code will live in JavaScript files referenced in a single HTML file that initializes your app.
  1. **Components:** These components can be reusable which help you to maintain the code when working on larger scale projects.
  2. **One-way Data Binding:** The benefits of one-way data binding give you better control throughout the application.
  3. **Simplicity:** ReactJS uses JSX file which makes the application simple and to code as well as understand.
  4. **Performance:** ReactJS is known to be a great performer. This feature makes it much better than other frameworks out there today. The reason behind this is that it manages a virtual DOM.
- **Visual Studio acts as a tool :** Debug code right from the editor. Launch or attach to your running apps and debug with breakpoints, call stacks, and interactive console.
- **Cloud Database: MongoDB :** MongoDB is a scalable, flexible NoSQL document database platform designed to overcome the relational databases approach .MongoDB Atlas is the leading global cloud database service for modern applications.
- Using Atlas, developers can deploy fully managed cloud databases across AWS, Azure, or Google Cloud. Best-in-class data security and privacy standards practices means that developers can rest easy knowing that they have instant access to the availability, scalability, and compliance they require for enterprise-level application development.
- MongoDB provides developers with a number of useful out-of-the-box capabilities, whether you need to run privately on site or in the public cloud.

### 4.2 Coding standards of Language used

Script name is same as class name, if we make static variable in a script which means we use that variable in another script with same names but if we don't use static variables and uses same name variables it sometime show error, so we need to rename the variable with different variables. Express is a routing and middleware web framework that has minimal functionality of its own: An Express application is essentially a series of middleware function calls.

### 4.3 Project Scheduling using PERT charts



PERT Chart representation

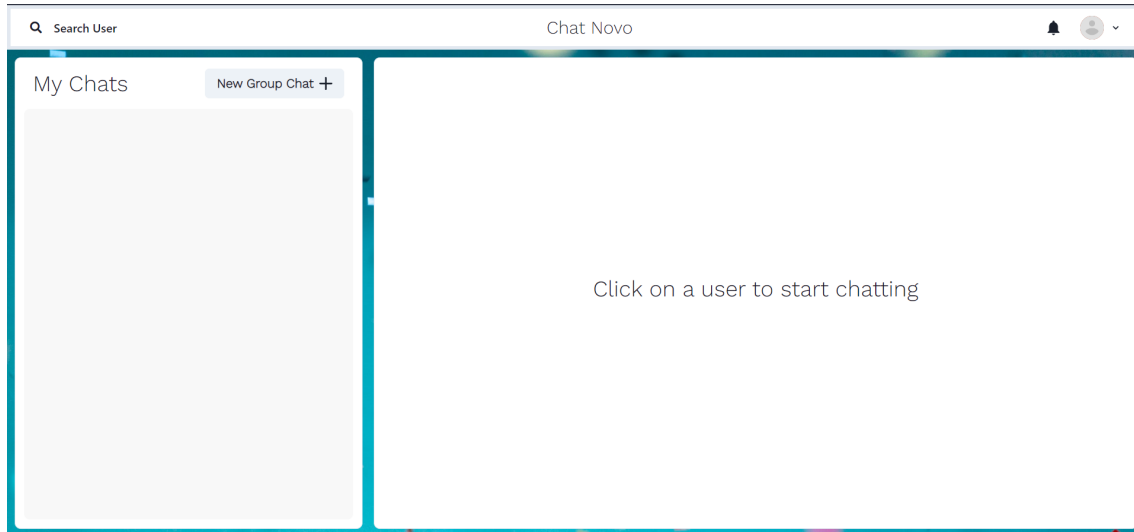
*Figure 4.1: PERT Chart*

### 4.4 Testing Techniques and Test Plans

We tested the browser compatibility as different browsers have different configurations and setting that your webpage should be compatible with. Data consistency is also very important in a web application. Check for data integrity and errors while you edit, delete, modify the forms or do any DB-related functionality. Check if all the Database query are executed correctly, data is retrieved and also updated correctly.

## 5 Results and Discussions

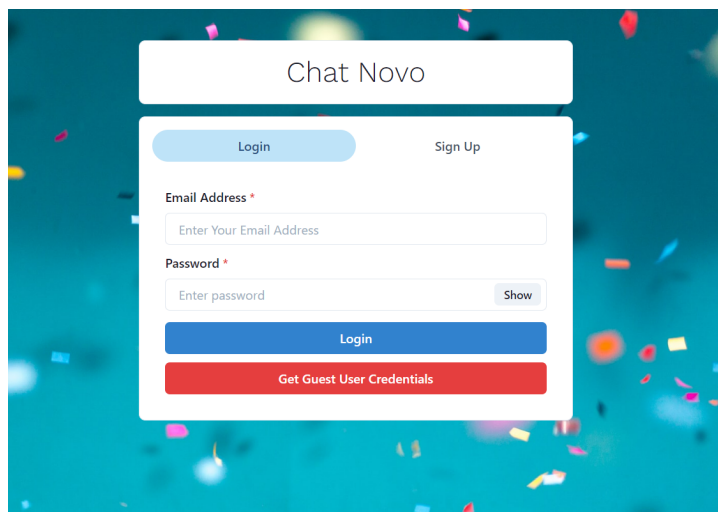
### 5.1 User Interface Representation



**Figure 5.1:** *User Interface Representation*

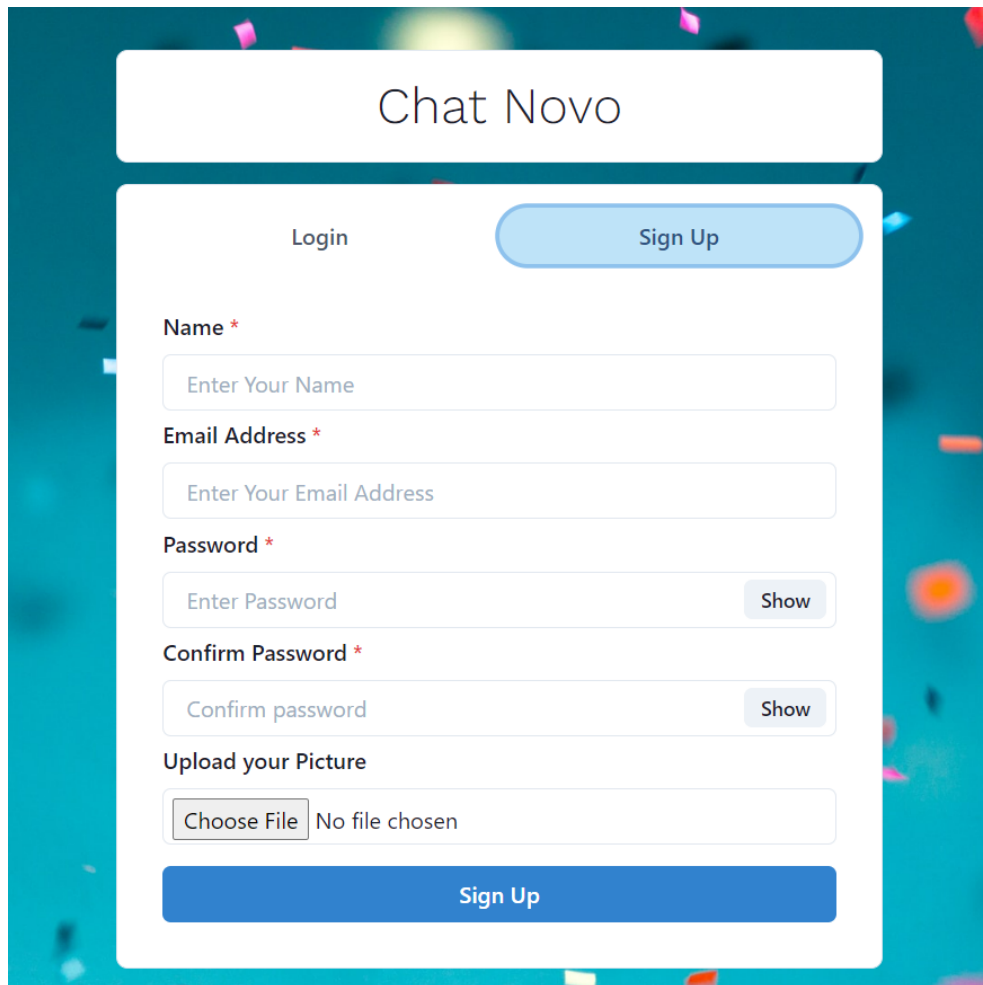
User Interface can be seen through the Home Page. In home page we can see a search bar, notification bar and profile button. On the left side user name is there and on the right side we can see the chat of the particular user as well as group chat.

### 5.2 Snapshots of system with brief detail of each



**Figure 5.2:** *Login Page*

If you are a user then you can directly login using your email id and password otherwise you will be redirected to the registration page.

The image shows a registration form for a website called "Chat Novo". The form is centered on a dark blue background with colorful confetti. At the top, the text "Chat Novo" is displayed in a white box. Below this, there are two tabs: "Login" and "Sign Up", with "Sign Up" being the active tab. The form fields include: "Name \*" with a placeholder "Enter Your Name"; "Email Address \*" with a placeholder "Enter Your Email Address"; "Password \*" with a placeholder "Enter Password" and a "Show" button; "Confirm Password \*" with a placeholder "Confirm password" and a "Show" button; and "Upload your Picture" with a "Choose File" button and the text "No file chosen". A large blue "Sign Up" button is at the bottom of the form.

# Chat Novo

LoginSign Up

**Name \***

**Email Address \***

**Password \***

Show

**Confirm Password \***

Show

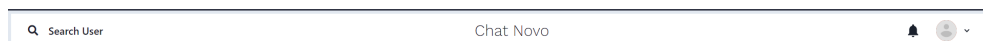
**Upload your Picture**

Choose File No file chosen

Sign Up

**Figure 5.3:** Registration Page

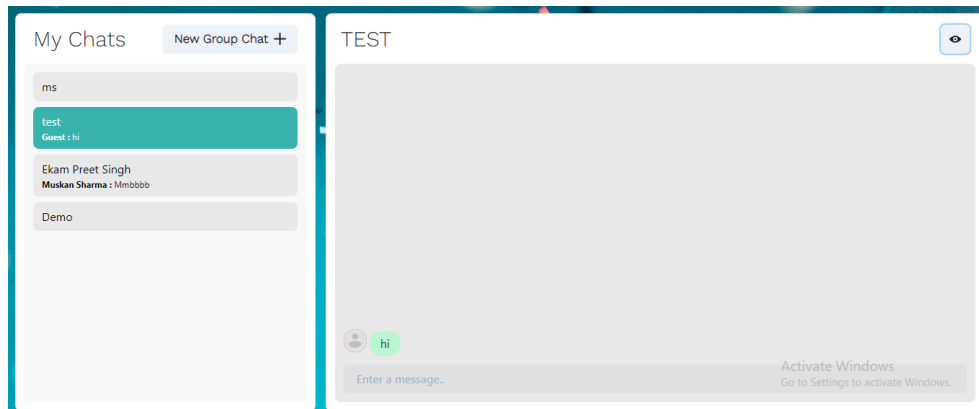
If you are a new user then firstly you have to register yourself here using name, email id and password after that you will be redirected to login page from where you can login into our website.

The image shows the title bar of the Chat Novo website. It is a horizontal bar with a light blue background. On the left, there is a search bar with a magnifying glass icon and the text "Search User". In the center, the text "Chat Novo" is displayed. On the right, there are three icons: a bell icon for notifications, a user profile icon, and a dropdown arrow.

Search UserChat Novo🔔👤▼

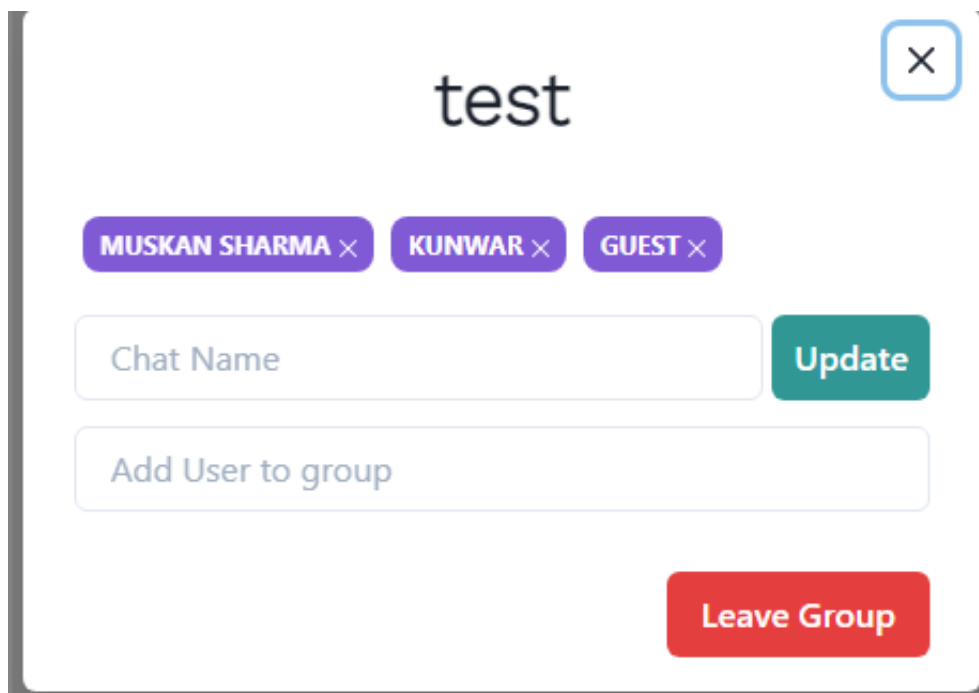
**Figure 5.4:** Title bar

Here the title bar of our website shows the name of our website and also include some other features like a search bar in which we can search the name of the user we want to chat on, the other side there is a bell shaped icon for a notification and the last icon is the profile view and we also logout from here.



**Figure 5.5:** Chat Section

There is the chat section in which all the users are seen who did chat with other user . Here all the personal and group chats are visible.



**Figure 5.6:** Group Details

Here is the information of group we can update the name of the group and also add and remove the users from the group.



## 5.3 Back Ends Representation

### 5.3.1 Snapshots of Database Tables

```
_id: ObjectId('627b8792c065895d8ccc39dd')
readBy: Array
sender: ObjectId('62736033a9c52e5898d34467')
content: "hi"
chat: ObjectId('627b8024aba4853f38d3be3c')
IpAddress: "127.0.0.1"
createdAt: 2022-05-11T09:53:22.506+00:00
updatedAt: 2022-05-11T09:53:22.506+00:00
__v: 0
```

*Figure 5.7: Storing chats*

```
_id: ObjectId('626ed0c535ff5612dc6219a1')
pic: "https://icon-library.com/images/anonymous-avatar-icon/anonymous-avatar..."
isAdmin: false
name: "Ekam Preet Singh"
email: "e@g.com"
password: "$2a$10$gtzBU3Hgx3GCIdLyy97V9.s8QBAq8olPtfw3jQlft5UtOXz1/e"
__v: 0
```

```
_id: ObjectId('626ed0ee35ff5612dc6219a2')
pic: "https://icon-library.com/images/anonymous-avatar-icon/anonymous-avatar..."
isAdmin: false
name: "Muskan Sharma"
email: "m@g.com"
password: "$2a$10$2GM2ygU3CoD6P.q8Q9VKnOD25A518LWW3.RgJ31gl1PmEc.NqrRgu"
__v: 0
```

```
_id: ObjectId('626ed15435ff5612dc6219a3')
pic: "https://icon-library.com/images/anonymous-avatar-icon/anonymous-avatar..."
isAdmin: false
name: "Runwar"
email: "k@g.com"
password: "$2a$10$Lctgpc2CWLXfnd9WcC13uu0ocK5wx2f7R6S1yQLWMeJ7R7PAqkt2g"
__v: 0
```

*Figure 5.8: User Information*

## **6 Conclusion and Future Scope**

### **6.1 Conclusion**

In this application, we have implemented chat functionality. There is a login facilitated with password checking feature. We have added profile view where user can see the display picture along with the email and username. This web application is solely developed with a purpose of solving a problem and to make the coming batches to interact with faculty and seniors along with their peers. So, we would like to conclude that we worked in a direction of betterment of college

### **6.2 Future Scope**

There is always a room for improvements in any apps. Right now, we are just dealing with text communication. There are several chat apps which serve similar purpose as this project, but these apps were rather difficult to use and provide confusing interfaces. A positive first impression is essential in human relationship as well as in human computer interaction. This project hopes to develop a chat service Web app with high quality user interface. In future we may be extended to include features such as:

- 1) Voice Message
- 2) File Transfer

## References/Bibliography

- [1] <https://www.mongodb.com/docs/realm/schemas/>  
Accessed: 05-04-2022
- [2] <https://expressjs.com/en/guide/routing.html>  
Accessed: 01-04-2022
- [3] <https://www.techtarget.com/searchdatamanagement/definition/CRUD-cycle>  
Accessed: 25-03-2022
- [4] <https://yellow.systems/blog/guide-to-the-chat-architecture>  
Accessed: 05-04-2022
- [5] Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node
- [6] <https://www.cometchat.com/blog/chat-application-architecture-and-system-design>  
Accessed: 21-05-2022
- [7] Learn MERN stack development by building modern web apps using MongoDB, Express, React, and Node.js, 2nd Edition