Simple App using NodeJS and MongoDB Atlas

Project Report Submitted to National Skill Training Institute for Women, Trivandrum in the Partial Fulfillment and Requirements for IBM Advanced Diploma(Vocational) in IT, Networking and Cloud

By

SHEHINA S ADIT/TVM/19/013

ADIT (2019-2021)

Under the supervision of

Mr. Poovaragavan Velumani (Master Trainer, Edunet Foundation)

GOVERNMENT OF INDIA

MINISTRY OF SKILL DEVELOPMENT & ENTREPRENEURSHIP

DIRECTORATE GENERAL OF TRAINING

January 2021

ABSTRACT

To create a simple application using NodeJS and MongoDB Atlas as its database.. This App is a simple application where users can save and add their contacts. Using these free apps it’s very easy to save the contact in a PC .And the contact saved by the user which includes the user's name and phone number are stored in the MongoDB Atlas database.

<u>CONTENTS</u>

1. INTRODUCTION

The objective of this project is for Building a Real-time App With NodeJS, Express, Socket.IO, and MongoDB . This application will provide user's to add contacts, and the user name and their phone numbers they are saving will be stored in MongoDB Atlas database. . This App can be accessed by different types of users. The username and messages stored in database are managed by **Admin**

Admin will have the authority to view all the username and contact numbers saved. Admin will also be responsible to provide user id and password for accessing the database. Admin will have the authority to delete the stored information. Users will have the privileges for adding and can retrieve,edit,view and can delete the contacts saved and added .

1.3 PROJECT DESCRIPTION

To build a simple app where users can store the contact numbers. Any number of users can use this app and the username and their contact numbers are visible to the admin and the details are stored in the MongoDB Atlas Database.

This app is going to have the following set of features.

- Add username and contact numbers
- Delete details entered on the app
- The details entered are stored on the MongoDB Atlas Database

1.4 SCOPE OF WORK

Thinking of an application where users can save their contacts in real-time. These contacts should appear for all users' applications as soon as the contacts are saved. User's can add and can save more number of contacts and can edit or can delete from the saved contacts.This will make users for retrieving their contacts very easily as soon as per their needs.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

At present, more than three billion people are using Smartphones with numerous apps installed on them. And it consists of different features also for saving the contacts.

2.2 PROPOSED SYSTEM

We are using the

- Node
- Express
- MongoDB and MongoDB Atlas
- CSS and nodemon packages to build the application. Nodemon is a package that restarts the server every time we make a change to the application code. It eliminates the need to manually stop and start the server every time we make a change. Unlike the other packages, will install nodemon as a development dependency since we use it only for development purposes.

3. SOFTWARE DEVELOPMENT ENVIRONMENT

In a world where the value of time is steadily increasing, building applications that users can interact with in real-time has become a norm for most of the developers. Most of the applications we see today, whether they are mobile, desktop, or web applications, have at least a single real-time feature included. As an example, real-time messaging and notifications are two of the most commonly used real-time features used in applications.

In this project, I am introducing you to the development of real-time applications using Node.js by building a real-time app. In fact, Node is one of the best programming languages out there to build real-time applications due to its event-driven and asynchronous nature. Before diving into building a real-time application head-first, we'll see what kind of real-time applications we can build using Node.js.

MongoDB Atlas is a Fully Managed Database Service which takes care of time-consuming and costly administration tasks so we can get the database resources we need, when we need them.

MongoDB Atlas is a cloud-based NoSQL database service developed by MongoDB Inc. It was developed to offer a flexible, scalable, and on-demand platform to eliminate the need for costly infrastructure, configurations, and maintenance.

MongoDB Atlas provides all the features of MongoDB—without the need to worry about database administration tasks such as:

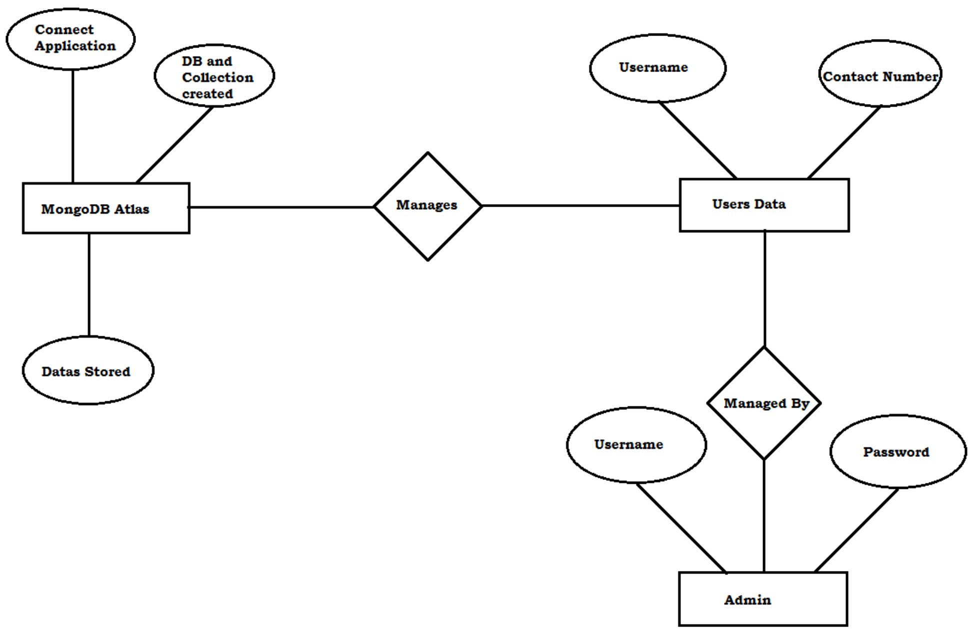
- Infrastructure provisioning
- Database configurations
- Patches
- Scaling
- Backups

The datas which are being entered in the application is stored on MongoDB Atlas Database .

4. SYSTEM DESIGN

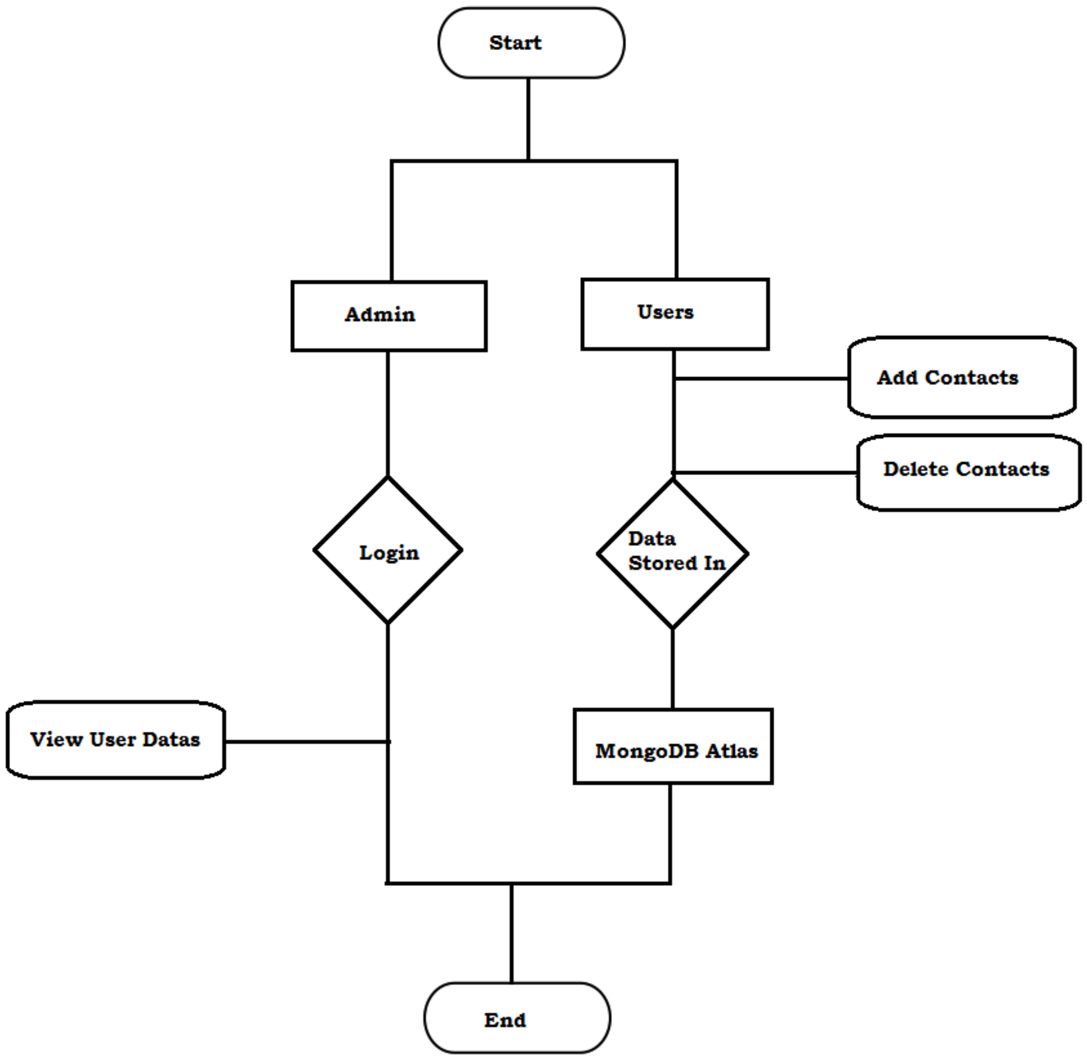
4.1 ER DIAGRAM

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database.



4.2 FLOW CHART

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence.



5. SYSTEM REQUIREMENTS

5.1 SOFTWARE SPECIFICATION

Operating System : Windows 7 or above

Front End : HTML, CSS, Java Script

Back End : MongoDB Atlas, express, NodeJS

Code Editor : , Visual Code

Browser : Google Chrome

5.2 HARDWARE SPECIFICATION

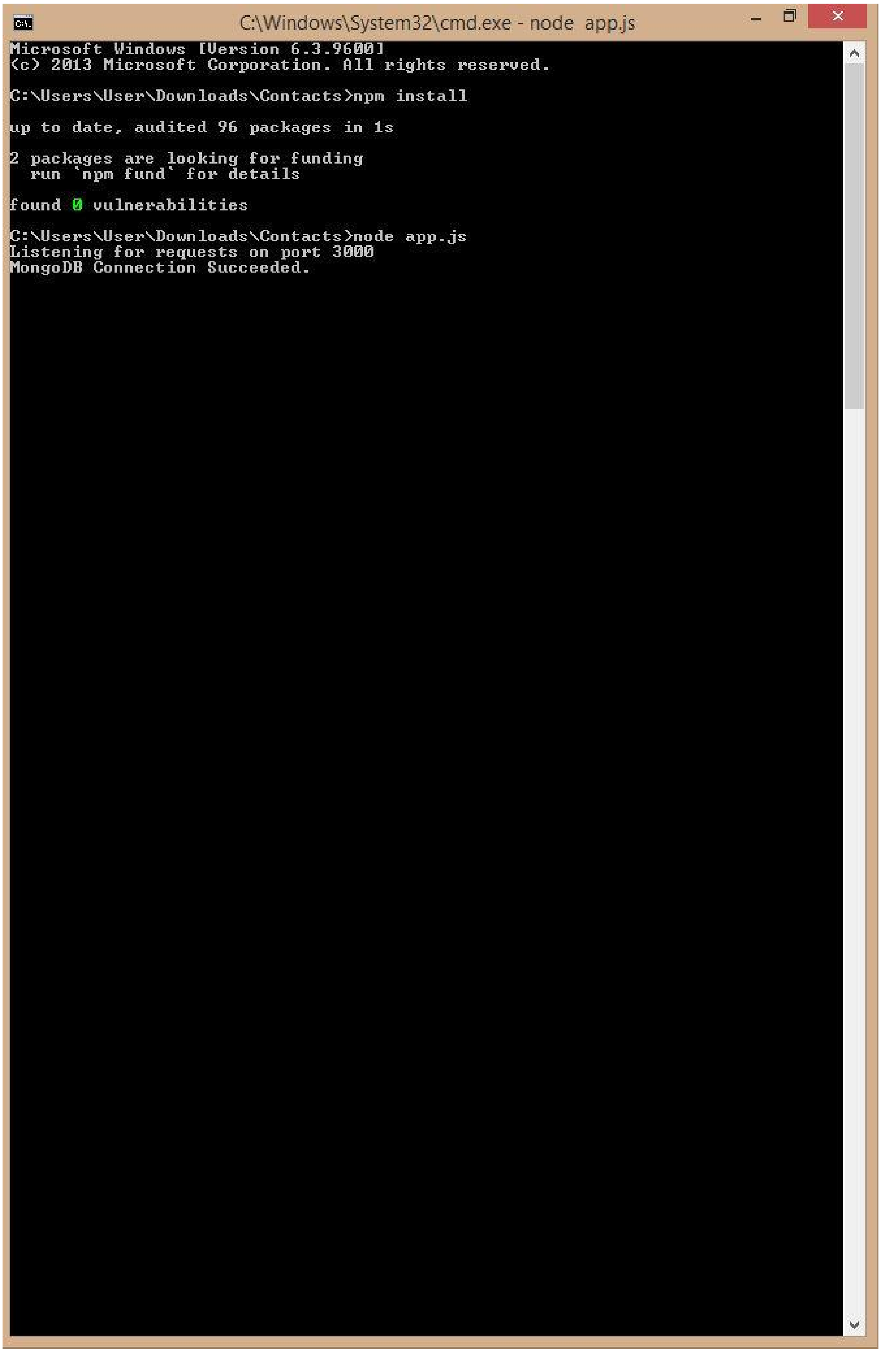
RAM : 1 GB or above

Processor : 1 GHz or more

Hard Drive : 32 GB or above

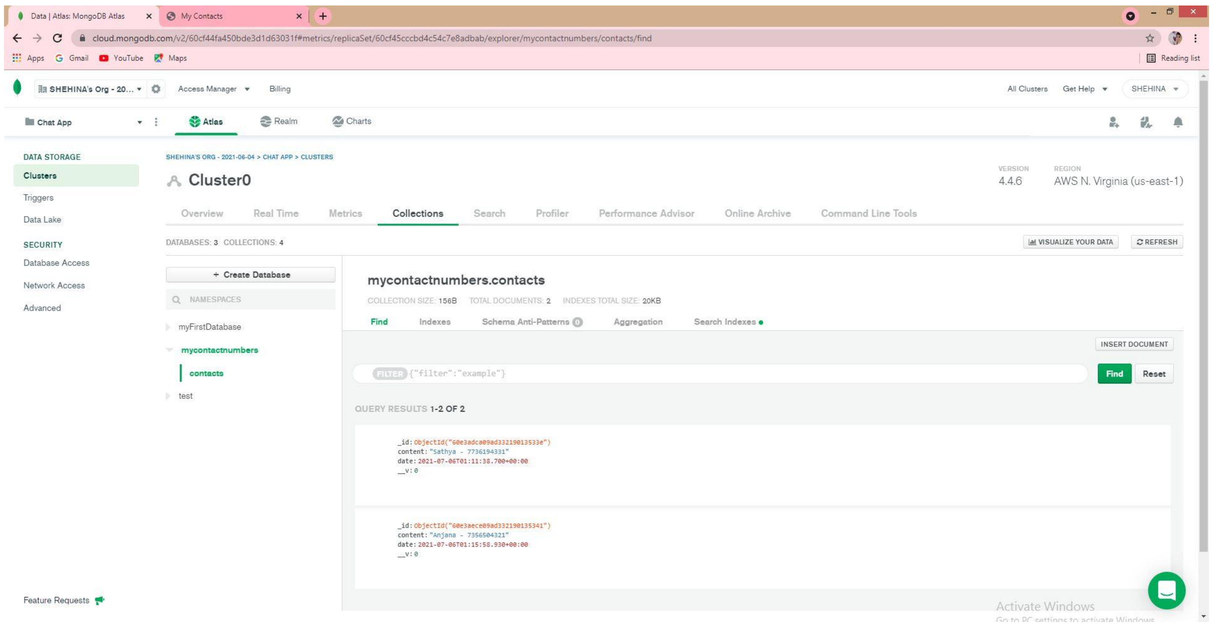
Network Connectivity: LAN or Wi-Fi

6. Command Prompt



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\User\Downloads\Contacts>npm install

up to date, audited 96 packages in 1s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\User\Downloads\Contacts>node app.js
Listening for requests on port 3000
MongoDB Connection Succeeded.
```

7. MongoDB Atlas Database


cloud.mongodb.com/v2/60cf44fa450bde3d1d63031f#metrics/replicaSet/60cf45cccbd4c54c7e8adbab/explorer/mycontactnumbers/contacts/find

SHEHINA'S ORG - 2021-06-04 > CHAT APP > CLUSTERS

Cluster0

VERSION 4.4.6

REGION AWS N. Virginia (us-east-1)

DATABASES: 3 COLLECTIONS: 4

mycontactnumbers.contacts

COLLECTION SIZE: 156B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 20KB

QUERY RESULTS 1-2 OF 2

```
_id: ObjectId("60e3adca09ad33219013533e")
content: "Sathya - 7736194331"
date: 2021-07-06T01:11:38.700+00:00
__v: 0
```

```
_id: ObjectId("60e3aece09ad332190135341")
content: "Anjana - 7356504321"
date: 2021-07-06T01:15:58.930+00:00
__v: 0
```

8. SOURCE CODE

Contact.ejs

```
<!DOCTYPE html>

<html>

<head>

<title>My Contacts</title>

<link        rel="stylesheet"        href="/static/stylesheets/style.css"
type="text/css">

<link              href='https://fonts.googleapis.com/css?family=Akronim'
rel='stylesheet'>

<link                                                   rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.8.2/css/all.css"

integrity="sha384-oS3vJWv+0UjzBfQzYUhtDYW+Pj2yciDJxpsK1OYPAYjqT085Qq/1c
q5FLXAZQ7Ay" crossorigin="anonymous">

</head>

<body>
```

```html
  <div align="center" class="contacts-container">

    <br>

    <p id="header">My Contacts</p>

    <br>

    <br>

    <div class="contact">

    <form action="/" method="POST" class="contact-header">

       <input  id="input"  type="text"  name="content"  placeholder="Add  a
contact...">

    <button type="submit"><span class="fas fa-plus"></span></button>

    </form>

    </div>

    <br>

    <br>
    <ul class="contact-list">

      <% contactsToAdd.forEach(details => { %>

      <li class="contact-list-item">

      <div class="contact-list-item-name"><%= details.content %></div>

      <a href="/edit/<%= details._id %>" class="edit">

        <span class="fas fa-edit"></span>

        </a>

        <a href="/remove/<%= details._id %>" class="remove">

        <span class="fas fa-times"></span>

        </a>

        </li>

      <% }) %>

    </ul>

  </div>
```

```
</body>

</html>
```

Style.css

```
* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

  }



body {

  padding-top: 30px;

  background: #8bcce6;

  font-family: sans-serif;

  color: black;
```

```
}

#header {
  font-size: 80px;
  width: 500px;
  text-align: center;
  color: white;
  font-family: akronim;
}

#input {
  width: 380px;
  height:35px;
  font-size: 18px;
  text-align: center;
}


#back {
  font-size: 20px;
  color: white;
}


.contact-container {
  color: rgb(83, 81, 81);
  position: absolute;
  top:50%;
  left: 50%;
  padding: 20px;
  width: 600px;

  transform: translate(-50%,-50%);

  background:  rgb(54, 150, 214);

  border-radius: 20px;

  display: flex;

  flex-direction: column;

  align-items: center;

}



.contact {

  width: 420px;

}



.contact-container h2 {
  padding: 10px;

  font-family:  sans-serif;

  font-size: 2em;

}



.contact-header input {

  width: 90%;

  border: 3px;

  border-radius: 5px;

  height: 35px;

  font-family: sans-serif;

}



.contact-header button {
  color:  rgb(9, 74, 117);

  border: none;

  width: 25px;

  border-radius: 50%;

  cursor: pointer;

  padding: 5px;

  background-color:  rgb(227, 234, 240);

}



.contact-container ul {

  list-style: none;

}



.contact-list-item {
  padding: 10px;

  margin-top: 10px;

  border: 5px;

  border-radius: 5px;

  font-family: sans-serif;

  font-size: 16px;

  background: white;

  display: flex;

  flex-direction: row;

  width: 90%;

}



.contact-list-item form {

  width: 100%;
```

```css
  display: flex;

}




.contact-list-item input {

  flex-grow: 2;

  margin-right: 20px;

  border: none;

  border-radius: 5px;

  font-family: sans-serif;

}




.contact-list-item button {

  color: white;
  border: none;

  border-radius: 5px;

  cursor: pointer;

  padding: 5px;

  font-family: sans-serif;

  background-color: rgb(37, 212, 243);

  font-size: 15px;

}



.contact-list-item .cancel {

  color: white;

  text-decoration: none;

  border: none;

  border-radius: 5px;
```

```css
  cursor: pointer;

  padding: 5px;

  font-family:  sans-serif;

  background-color: rgb(37, 212, 243);

  margin-left: 5px;

}




.contact-list-item div {

  flex-grow: 2;

}

.contact-list-item .edit {

  color : rgb(37, 212, 243);

  margin-right: 10px;
  cursor:pointer;

  text-decoration: none;

}

.contact-list-item .remove {

  color : rgb(54, 150, 214);

  margin-right: 5px;

  cursor:pointer;

  text-decoration: none;

}
```

App.js

```javascript
const express = require("express");

const app = express();

const dotenv = require('dotenv');

const Contact = require("./models/Contact");
```

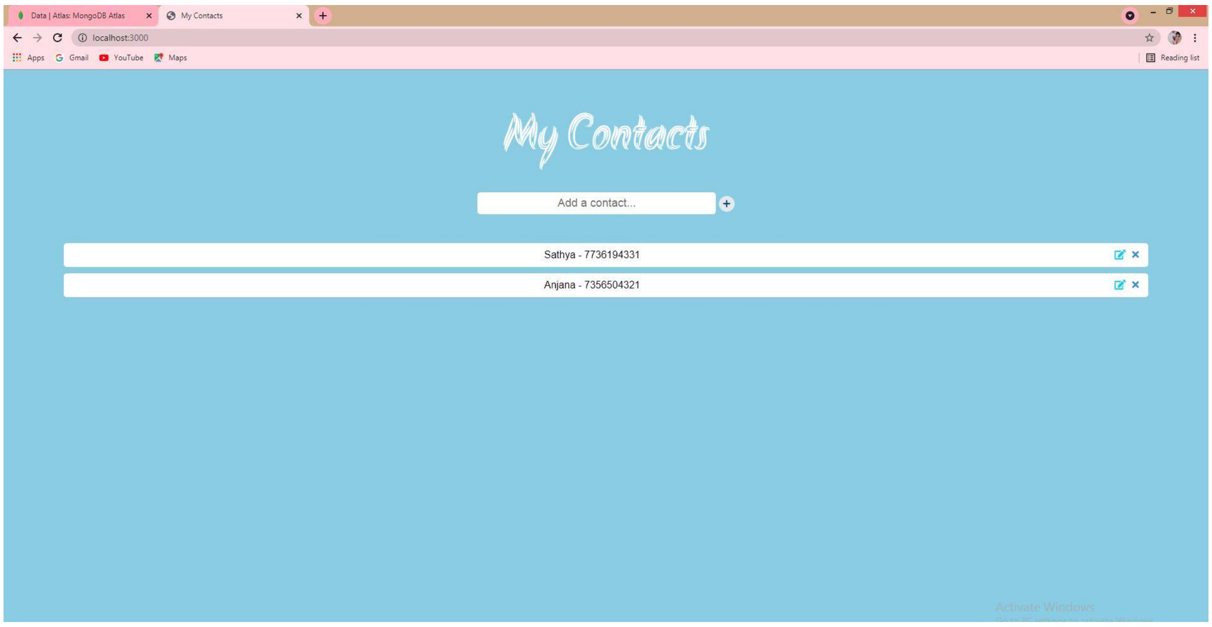
```
dotenv.config();

app.set("view engine", "ejs");

app.use(express.urlencoded({ extended: true }));

app.use('/static', express.static('public'));

const mongoose = require("mongoose");

mongoose.set("useFindAndModify", false);

mongoose.connect('mongodb+srv://Shehina:shehina1998@cluster0.w1zdm.mong
odb.net/mycontactnumbers?retryWrites=true&w=majority', {

  useNewUrlParser: true,

  useUnifiedTopology: true

}, (err) => {

  if (!err) {

    console.log('MongoDB Connection Succeeded.');

  } else {
    console.log('Error in DB connection : ' + err);
  }
});
app.listen(3000, function() {
  console.log("Listening for requests on port 3000")
});
app.get('/', (req, res) => {
  Contact.find({}, (err, contacts) => {
  res.render('contacts.ejs', { contactsToAdd: contacts});
});
});
app.post('/', async (req, res) => {
   const contact = new Contact({
      content: req.body.content
    });
    try {
      await contact.save();
      console.log(content)
        res.redirect("/");
        } catch (err) {
        res.redirect("/");
        }
    });
app.route("/edit/:id")
  .get((req, res) => {
    const id = req.params.id;
    Contact.find({}, (err, contacts) => {
         res.render("edit.ejs", { contactsToAdd: contacts, idContact:
```

```
id});

    });

})

  .post((req, res) => {

  const id = req.params.id;

  Contact.findByIdAndUpdate(id, { content: req.body.content }, err => {

    if (err) return res.send(500, err);

    res.redirect('/');

  });

  });

app.route("/remove/:id").get((req, res) => {

  const id = req.params.id;

  Contact.findByIdAndRemove(id, err => {

  if (err) return res.send(500, err);
  res.redirect("/");

  });

});
```

Package.json

```
{

  "name": "mycontacts",

  "version": "1.0.0",

  "description": "",

  "main": "app.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "node app.js"

  },

  "author": "",
  "license": "ISC",

  "dependencies": {

    "dotenv": "^8.2.0",

    "ejs": "^3.1.3",

    "express": "^4.17.1",

    "mongoose": "^5.9.17"

  }

}
```

9. SCREENSHOT



Data | Atlas: MongoDB Atlas

My Contacts

localhost:3000

Apps Gmail YouTube Maps Reading list

My Contacts

Add a contact...

Sathya - 7736194331

Anjana - 7356504321

10. CONCLUSION

Today, using real-time features with desktop, mobile, and web applications has almost become a necessity. In this project, includes a real-time app has been created with the help of using NodeJS and MongoDB Atlas.

11. REFERENCE

1. https://livecodestream.dev/post/a-starter-guide-to-building-real-time-applications-with-nodejs/
2. https://www.mongodb.com/cloud/atlas
3. https://www.bmc.com/blogs/mongodb-atlas