**Software Application Design Documentation**

Raushawn Mitchell and Aaron Hazzard

Diploma in Software Engineering, University of Trinidad and Tobago

SFEN2005: Software Engineering II

**Table Of Contents**

**ERD Diagram…………………………3**

**Creation of Databases………………..3-11**

**Code Snippets…………………………11-27**

**Client-Side Programming……………27-49**

**Server-Side Programming……………49-51**

**ERD Diagram**

A picture containing text, diagram, plan, technical drawing

Description automatically generated

**Creation of Databases**

**Schemas converted to SQL:**

#Users table

CREATE TABLE Users (

id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

is\_staff BOOLEAN DEFAULT false,

is\_active BOOLEAN DEFAULT false,

date\_joined TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

-- Additional table for the many-to-many relationship between Users and Jobs

user\_jobs INT

);

ALTER TABLE Users

ADD CONSTRAINT FK\_Users\_user\_jobs

FOREIGN KEY (user\_jobs)

REFERENCES Jobs(id);

select \* from users;

#User Profile table

CREATE TABLE UserProfile (

id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL UNIQUE,

profile\_picture VARCHAR(255),

bio VARCHAR(255),

FOREIGN KEY (user\_id) REFERENCES Users(id)

);

#About us table

CREATE TABLE AboutUs (

id INT PRIMARY KEY AUTO\_INCREMENT,

information VARCHAR(255) NOT NULL

);

#Job categories table

CREATE TABLE JobCategories (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(255) NOT NULL

);

#Jobs table

CREATE TABLE Jobs (

id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

title VARCHAR(255) NOT NULL,

description TEXT NOT NULL,

thumbnail VARCHAR(255) NOT NULL,

price DECIMAL(10,2) NOT NULL,

category\_id INT NOT NULL,

FOREIGN KEY (user\_id) REFERENCES Users(id),

FOREIGN KEY (category\_id) REFERENCES JobCategories(id)

);

#Messages table

CREATE TABLE Messages (

id INT PRIMARY KEY AUTO\_INCREMENT,

message\_text VARCHAR(255) NOT NULL,

sender\_id INT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (sender\_id) REFERENCES Users(id)

);

#Ratings table

CREATE TABLE Ratings (

id INT PRIMARY KEY AUTO\_INCREMENT,

job\_id INT NOT NULL,

freelancer\_id INT NOT NULL,

user\_id INT NOT NULL,

rating INT NOT NULL,

feedback VARCHAR(255),

date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (job\_id) REFERENCES Jobs(id),

FOREIGN KEY (freelancer\_id) REFERENCES Users(id),

FOREIGN KEY (user\_id) REFERENCES Users(id)

);

#Services table

CREATE TABLE Services (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(255) NOT NULL UNIQUE,

description TEXT NOT NULL,

thumbnail VARCHAR(255) NOT NULL

);

**Original Schemas:**

const aboutUsSchema = new mongoose.Schema({

information: {

type: String,

required: true

}

})

const jobCategoriesSchema = new mongoose.Schema({

name:{

type: String,

required: true

}

})

const jobSchema = new mongoose.Schema({

user\_id: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

required: true,

},

title: {

type: String,

required: true,

},

description: {

type: String,

required: true,

},

thumbnail: {

type: String,

required: true,

},

price: {

type: Number,

required: true,

},

category: {

type: String,

required: true,

ref: 'JobCategories',

},

});

const MessageSchema = mongoose.Schema(

{

message: {

text: { type: String, required: true },

},

users: Array,

sender: {

type: mongoose.Schema.Types.ObjectId,

ref: "User",

required: true,

},

},

{

timestamps: true,

}

);

const RatingsSchema = mongoose.Schema({

jobID: {

type: mongoose.Schema.Types.ObjectId,

ref: 'Jobs',

required: true

},

freeLancerID: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

required: true

},

userID: {

type: mongoose.Schema.Types.ObjectId,

ref: 'User',

required: true

},

ratings: {

type: Number,

required: true,

min: 1,

max: 5

},

feedback: {

type: String,

maxlength: 255,

required: false

},

date: {

type: Date,

default: Date.now

}

})

const serviceSchema = new mongoose.Schema({

title: {

type: String,

required: true,

unique: true,

},

description: {

type: String,

required: true,

},

thumbnail: {

type: String,

required: true,

},

});

const userSchema = new mongoose.Schema({

username: {

type: String,

required: true,

},

email: {

type: String,

required: true,

unique: true,

},

password: {

type: String,

required: true,

min: 8,

},

is\_staff: {

type: Boolean,

default: false

},

is\_active: {

type: Boolean,

default: false

},

date\_joined: {

type: Date,

default: Date.now

},

jobs: [{

type: mongoose.Schema.Types.ObjectId,

ref: 'Job'

}]

});

const userProfileSchema = new mongoose.Schema({

  user\_id: {

    type: mongoose.Schema.Types.ObjectId,

    ref: 'User',

    required: true,

    unique: true,

  },

  profile\_picture: {

    type: String,

  },

  bio: {

    type: String,

  },

});

**Code Snippets**

**Database Connection String**

***Env file***

MONGO\_URL=”mongodb+srv://868drgnstudio:123drgnstudio@cluster0.lgjbopq.mongodb.net/test”

***Server file***

mongoose

  .connect(process.env.MONGO\_URL, {

    useNewUrlParser: true,

    useUnifiedTopology: true,

  })

  .then(() => {

    console.log("DB Connection Successfull");

  })

  .catch((err) => {

    console.log("ERROR CONNECTION: " +err.message);

  });

**API Calls Used to Interact with Database**

***About Us Related Code:***

module.exports.updateAbout = async (req, res, next) => {

    try {

      const { information } = req.body;

      console.log(`Attempting to update about us content with information: “${information}”`);

      if (!information) {

        return res.status(400).json({ error: ‘Missing required field’ });

      }

      const about = await About.create({ information });

      console.log(‘Information Updated’);

      return res.status(201).json({ message: ‘Information Added Successfully’ });

    } catch (err) {

      console.log(`Error updating about us content: ${err}`);

      return res.status(500).json({ error: ‘Internal Server Error’ });

    }

  };

  module.exports.getAboutInfo = async (req, res, next) => {

    try {

      console.log(‘Fetching about us information’)

      const about = await About.findOne({ \_id: “6449a1f4cfca4b43a156d685” }).select(‘information’).exec();

      if (about) {

        return res.status(200).json({ information: about.information });

      } else {

        console.error(‘No information found’)

        return res.status(400).json({ error: ‘No information found for the given ID’ });

      }

    } catch (err) {

      console.log(`Error retrieving about us content: ${err}`);

      return res.status(500).json({ error: ‘Internal Server Error’ });

    }

  };

***Job/ Services Related Code:***

module.exports.getAllJobs = async (request, response, next) => {

  try {

    const jobs = await Jobs.aggregate([

      {

        $lookup: {

          from: ‘users’,

          localField: ‘user\_id’,

          foreignField: ‘\_id’,

          as: ‘user’,

        },

      },

    ]);

    const job\_list = [];

    const reversedJobList = []

    jobs.forEach(job => {

      const str1 = job.category.replace(“{ name: ‘”, “”)

      const str2 = str1.replace(“’ }”, “”)

      job\_list.push({

        job\_id: job.\_id,

        user\_id: job.user\_id,

        title: job.title,

        description: job.description,

        thumbnail: job.thumbnail,

        price: job.price,

        category: str2,

        username: job.user[0].username

      });

    });

    for(let I = job\_list.length – 1; i>= 0; i--){

      reversedJobList.push(job\_list[i])

    }

    if (job\_list.length > 0) return response.status(200).json({ reversedJobList });

    else return response.status(400).json({ error: “There are currently no jobs available ☹“ });

    } catch (err) {

      console.log(`Error retrieving jobs: ${err}`);

      return response.status(500).json({error: ‘Internal Server Error’});

    }

};

module.exports.searchJobs = async (request, response) => {

  const { jobCategory, search } = request.body;

  console.log(“Attempting to Search For Jobs”);

  console.log(“Searching for TITLE: “ + search);

  if (jobCategory) console.log(“At category “ + jobCategory);

  else console.log(“No categories”);

  try {

    // Get the jobs and users

    const jobs = await Jobs.find({

      $or: [

        { title: { $regex: search, $options: “m” } },

        { description: { $regex: search, $options: “m” } },

        { category: { $regex: search, $options: “m” } },

      ],

    });

    if (jobCategory) {

      const jobCategories = await JobCategories.find({

        name: jobCategory,

      });

      if (jobCategories.length > 0) {

        const jobCategoryId = jobCategories[0].\_id;

        const jobsInCategory = await Jobs.find({

          category: jobCategoryId,

        });

        if(jobs) console.log(jobs)

        if(jobsInCategory) console.log(jobsInCategory)

        response.status(200).json({jobCategory});

      } else {

        response.status(400).json({ error: “There are currently no jobs available in this category ☹“ });

      }

    } else {

      response.status(200).json(jobs);

    }

  } catch (err) {

    console.log(err);

    response.status(500).json({error: ‘Internal Server Error’});

  }

};

module.exports.searchResults = async (request, response) => {

  try {

    // Get the query parameters

    const {term, category} = request.body;

    console.log(term + “, “)    // Create a query object

    console.log(category + “, “)    // Create a query object

    const query = {

      $or: [

        { title: { $regex: term, $options: “m” } },

        { description: { $regex: term, $options: “m” } },

        { category: { $regex: term, $options: “m” } },

      ]

    };

    // If a category is specified, add it to the query

    if (category) {

      query.category = category;

    }

    // Find the jobs that match the query

    const jobs = await Jobs.find(query);

    // If no jobs are found, return an error message

    if (jobs.length === 0) return response.status(404).json({error: “No jobs found for “ + term,category});

    const job\_list = []

    // Convert the jobs to a JSON object

    const jobList = jobs.map(job => ({

      job\_id: job.\_id,

      user\_id: job.user\_id,

      title: job.title,

      description: job.description,

      thumbnail: job.thumbnail,

      price: job.price,

      category: job.category,

      username: JobCategories.findOne({\_id: job.category}).username,

    }));

    job\_list.push(jobList)

    console.log(JSON.stringify(job\_list[0]));

    // Return the job list to the client

    response.status(200).json({job\_list});

  } catch (err) {

    console.log(err);

    response.status(500).json({

      error: “An error occurred”,

      err,

    });

  }

};

module.exports.getCategories = async(request, response) => {

  try{

      const categories = await JobCategories.find({}, {

        // select only the name field

        \_id: false,

        name: true

      });

      return response.status(200).json({categories})

  }catch(error){

    console.log(‘An error occurred while fetching categories’, error)

    return response.status(500).json({error: ‘Internal Server Error’});

  }

}

module.exports.jobDetails = async(request, response) => {

  const { jobID } = request.params;

  const job = await Jobs.findOne({ \_id: jobID });

  if (!job) {

    console.log(“No job found with ID: “ + jobID);

    return response.status(404).json({ error: “No job found with ID: “ + jobID });

  }

  const user = await Users.findOne({ \_id: job.user\_id });

  if (!user) {

    console.log(“No user found with ID: “ + job.user\_id);

    return response.status(404).json({ error: “No user found with ID: “ + job.user\_id });

  }

  const jobDetails = []

  const jobDetails\_list = {

    job\_id: job.\_id,

    seller\_id: job.user\_id,

    title: job.title,

    description: job.description,

    thumbnail: job.thumbnail,

    price: job.price,

    category: job.category,

    username: user.username,

    user\_bio: user.bio,

    profile\_picture: user.profile\_picture,

  };

  jobDetails.push(jobDetails\_list)

  return response.status(200).json({jobDetails});

};

module.exports.createJob = async (request, response) => {

  try {

    const {user\_id, title, description, price, category} = request.body

    let thumbnail = ‘’;

    if (request.file) {

      thumbnail = request.file.filename.toString(‘base64’)

      // do something with the file

    } else {

      console.log(‘No file uploaded\n’);

      // handle the case where no file was uploaded

    }

    console.log(‘user id:’, user\_id, ‘\ntitle:’, title, ‘description:’, description, ‘price:’, price, ‘category:’, category, ‘thumbnail:’, thumbnail)

    const job = await Jobs.create({

      user\_id,

      title,

      description,

      thumbnail,

      price,

      category,

    });

    return response.status(200).json({ message: ‘Created job successfully’ });

  } catch (error) {

    console.log(‘An error occurred while creating job’, error);

    return response.status(500).json({ error: ‘Internal Server Error’ });

  }

module.exports.createService = async(request, response) => {

  const title = request.body.title;

  const description = request.body.description;

  let thumbnail=’’;

  if (request.file) {

    thumbnail = request.file.filename

    // do something with the file

  } else {

    console.log(‘No file uploaded’);

    // handle the case where no file was uploaded

  }

  if (!title || !description || !thumbnail) {

    return response.status(400).json({ error: ‘Missing required field’ });

  }

  Service.create({ thumbnail, title, description }, (err, service) => {

    if (err) {

      console.error(err);

      return response.status(500).json({ error: ‘Internal server error’ });

    }

    console.log(`${title} Service Added`);

    return response.status(201).json({ message: `${title} Service Added Successfully` });

  });

};

***User Related Code:***

module.exports.updateUser = async (req, res) => {

  const { userID, username, email, password} = req.body;

  console.log(`User ID: ${userID}, Username: ${username}, Email: ${email}, Password: ${password}`)

  try {

    // Find the user by ID

    const user = await User.findById(userID);

    if (!user) {

      return res.status(404).json({ error: ‘User not found’ });

    }

    // Update the user’s fields

    user.username = username || user.username;

    user.email = email || user.email;

    if (password) {

      user.password = await bcrypt.hash(password, 10);

    }

    // Save the updated user

    await user.save();

    // Return the updated user object

    console.log(‘User account updated’)

    return res.json({ error: ‘Profile updated’, user });

  } catch (ex) {

    console.log(ex);

    return res.status(500).json({ error: ‘Internal Server Error’ });

  }

};

module.exports.updateProfile = async (req, res) => {

  const { userID, bio } = req.body;

  let update1 = false;

  let update2 = false;

  let profile\_picture = ‘’;

  if (req.file) {

    profile\_picture = req.file.filename.toString(‘base64’)

    console.log(profile\_picture)

    // do something with the file

  } else {

    console.log(‘No file uploaded\n’);

    // handle the case where no file was uploaded

  }

  console.log(‘Attempting to update profile’);

  if(!userID) {

    console.log(‘You are not authorized to perform this action’)

    return res.status(401).json({error: ‘You are not authorized to perform this action’})

    }

    console.log(`userID: ${userID}`);

  if(bio == ‘undefined’ && profile\_picture == ‘undefined’) {

    console.log(‘You need to upload something’)

    return res.status(403).json({error: ‘You need to upload something’})

  }

  console.log(`bio: ${bio}`);

  console.log(`Profile Picture: ${profile\_picture}`);

  try {

    if (profile\_picture != ‘undefined’ && profile\_picture != ‘’) {

      console.log(‘Updating profile picture’)

      const updatePicture = await userProfile.updateOne(

        { user\_id: userID },

        { $set: { profile\_picture } }

      );

      console.log(‘Updated picture successfully’);

      update1 = true

    }

    if (bio != ‘undefined’ && bio != ‘’) {

      const updateBio = await userProfile.updateOne(

        { user\_id: userID },

        { $set: { bio } }

      );

      console.log(‘Updated bio successfully’);

      update2 = true

    }

    if(update1 == true || update2 == true)

      res.status(200).json({ message: ‘Profile updated successfully’ })

    else

      res.status(500).json({ error: ‘An unexpected error occurred. Try again.’ })

  } catch (err) {

    console.error(`${err}\nUPDATE PROFILE ERROR!!`);

    return res.status(500).json({ error: ‘Internal server error’ });

  }

};

module.exports.getUserProfile = async (req, res, next) => {

  try {

    const userID = req.params.id;

    // Find the profile and user data for the specified user ID.

    Const profile = await userProfile.aggregate([

      { $match: { user\_id: mongoose.Types.ObjectId(userID) } },

      { $lookup: { from: ‘users’, localField: ‘user\_id’, foreignField: ‘\_id’, as: ‘user’ } },

      { $unwind: ‘$user’ },

      { $project: { \_id: ‘$user.\_id’, username: ‘$user.username’, profile\_picture: 1, bio: 1 } },

    ]);

    if (profile.length === 0) {

      return res.status(404).json({ error: ‘Profile not found’ });

    }

    // Format the response data.

    Const user\_profile = {

      id: profile[0].\_id,

      user\_id: userID,

      profile\_picture: profile[0].profile\_picture,

      name: profile[0].username,

      bio: profile[0].bio,

    };

    user\_profile.bio == undefined ? user\_profile.bio = ‘undefined’ : null

    user\_profile.profile\_picture == undefined ? user\_profile.profile\_picture = ‘undefined’ : null

    return res.status(200).json({ message: ‘Profile fetched successfully’, user\_profile });

  } catch (ex) {

    next(ex);

  }

};

module.exports.login = async (req, res, next) => {

  try {

    const { email, password } = req.body;

    console.log(‘Attempting to log in’)

    if (email == ‘’) {

      console.log(“Enter your email”)

      return res.json({error: “Form Violation! Enter your email”})

    }

    if (password == ‘’) {

      console.log(“Enter your password”)

      return res.json({error: “Form Violation! Enter your password”})

    }

    const user = await User.findOne({ email });

    if (!user)

      return res.json({ error: “Incorrect email or password”, status: false });

    const isPasswordValid = await bcrypt.compare(password, user.password);

    if (!isPasswordValid)

      return res.json({ error: “Incorrect email or password”, status: false });

    delete user.password;

    console.log(‘LOGGED IN’)

    return res.json({ status: true, user });

  } catch (ex) {

    next(ex);

  }

};

module.exports.register = async (req, res, next) => {

  try {

    const { username, email, password } = req.body;

    if (email == ‘’) {

      console.log(“Enter your email”)

      return res.json({error: “Enter your email”})

    }

    if (username == ‘’) {

      console.log(“Enter your username”)

      return res.json({error: “Enter your username”})

    }

    if (password == ‘’) {

      console.log(“Enter your password”)

      return res.json({error: “Enter your password”})

    }

    const usernameCheck = await User.findOne({ username });

    if (usernameCheck)

      return res.json({ error: “Username already used”, status: false });

    const emailCheck = await User.findOne({ email });

    if (emailCheck)

      return res.json({ error: “Email already used”, status: false });

    const hashedPassword = await bcrypt.hash(password, 10);

    console.log(hashedPassword)

    const user = await User.create({

      email,

      username,

      password: hashedPassword,

    });

    delete user.password;

    // Insert the user’s ID into the profiles table.

    Const profile = await userProfile.create({

      user\_id: user.id,

      profile\_picture: null

    });

    return res.json({ status: true, user, profile });

  } catch (ex) {

    next(ex);

  }

};

module.exports.getAllUsers = async (req, res, next) => {

  try {

    const users = await User.find({ \_id: { $ne: req.params.id } }).select([

      “email”,

      “username”,

      “profile\_picture”,

      “\_id”,

    ]);

    users.forEach(user => {

      user.profile\_picture == undefined ? users.profile\_picture = ‘undefined’ : null

    })

    console.log(users)

    return res.json({users});

  } catch (ex) {

    next(ex);

  }

};

module.exports.getAllUserInfo = async (req, res, next) => {

  try {

    const userInfo = await User.find();

    res.json(userInfo);

  } catch (ex) {

    next(ex);

  }

};

module.exports.setAvatar = async (req, res, next) => {

  try {

    console.log(‘fetched avatar api’)

    const userId = req.params.id;

    const avatarImage = req.body.image;

    console.log(`USER ID: ${userId}`)

    const userData = await User.findByIdAndUpdate(

      userId,

      {

        isAvatarImageSet: true,

        avatarImage,

      },

      { new: true }

    );

    return res.json({

      isSet: userData.isAvatarImageSet,

      image: userData.avatarImage,

    });

  } catch (ex) {

    next(ex);

  }

};

module.exports.logOut = (req, res, next) => {

  try {

    if (!req.params.id) return res.json({ msg: “User id is required “ });

    onlineUsers.delete(req.params.id);

    return res.status(200).send();

  } catch (ex) {

    next(ex);

  }

};

***Messages Related Code:***

module.exports.getMessages = async (req, res, next) => {

  try {

    const { from, to } = req.body;

    const messages = await Messages.find({

      users: {

        $all: [from, to],

      },

    }).sort({ updatedAt: 1 });

    const projectedMessages = messages.map((msg) => {

      return {

        fromSelf: msg.sender.toString() === from,

        message: msg.message.text,

      };

    });

    res.json(projectedMessages);

  } catch (ex) {

    next(ex);

  }

};

module.exports.addMessage = async (req, res, next) => {

  try {

    const { from, to, message } = req.body;

    const data = await Messages.create({

      message: { text: message },

      users: [from, to],

      sender: from,

    });

    if (data) return res.json({ msg: “Message added successfully.” });

    else return res.json({ msg: “Failed to add message to the database” });

  } catch (ex) {

    next(ex);

  }

};

***Ratings Related Code:***

module.exports.rateFreelancers = async(request, response) => {

    const { jobID, freeLancerID, clientID,  ratings, feedback } = request.body;

    try{

        await Ratings.create({

            jobID,

            freeLancerID,

            userID: clientID,

            ratings,

            feedback

        })

        return response.json({message: “Thank you for your feedback :D”})

    }catch(error){

        console.log(error + “\n\nERROR Rating freelancer”)

        return response.json({error: “Internal Server Error”})

    }

}

module.exports.getFreelancerRatings = async(request, response) => {

  const { jobID, freeLancerID } = request.body

  // console.log(‘jobID’, jobID, “freelancerID”, freeLancerID)

  try{

      const freeLancerRatings = await Ratings.aggregate([

      //stage1

      {

          $match: {

              “jobID”: ObjectId(jobID),

              “freeLancerID”: ObjectId(freeLancerID)

          }

      },

      {

        $group: {

          \_id: null,

          totalRatings: { $sum: “$ratings” }

        }

      }

      ])

      if (freeLancerRatings.length > 0) {

        const sumOfRatings = freeLancerRatings[0].totalRatings;

        return response.json({freeLancerRatings: sumOfRatings})

      } else {

        console.log(“No ratings found for the freelancer”);

        return 0;

      }

  }catch(error){

      console.log(error + “\n\nERROR getting ratings”)

      return response.json({error: “Internal Server Error”})

  }

}

module.exports.getRatings = async(request, response) => {

    const { jobID, freeLancerID, clientID } = request.body

    // console.log(jobID, ‘ is jobID’, freeLancerID, ‘ is freelancerID’, clientID, ‘ is clientID’)

    try{

        const totalRatings = await Ratings.aggregate([

        //stage1

        {

            $match: {

                “jobID”: ObjectId(jobID),

                “freeLancerID”: ObjectId(freeLancerID),

                “userID”: ObjectId(clientID)

            }

        },

        ])

        // console.log(totalRatings.ratings, “\nTotal ratings”)

        return response.json({totalRatings})

    }catch(error){

        console.log(error + “\n\nERROR getting ratings”)

        return response.json({error: “Internal Server Error”})

    }

}

module.exports.getAllRatings = async (request, response) => {

  const { jobID, freeLancerID } = request.body;

  try {

    await Ratings.aggregate([

      {

        $match: {

          “jobID”: ObjectId(jobID),

          “freeLancerID”: ObjectId(freeLancerID),

        }

      },

      {

        $lookup: {

          from: ‘users’, // name of the users collection

          localField: ‘userID’,

          foreignField: ‘\_id’,

          as: ‘users’,

        },

      },

      {

        $unwind: ‘$users’,

      },

      {

        $project: {

          \_id: 0,

          username: ‘$users.username’,

          feedback: 1,

          ratings: 1

        },

      },

    ])

    .then((results) => {

      // results will contain an array of objects with the “username” field

      // const usernames = results.map((result) => result.username);

      // console.log(results);

      return response.json({ allRatings: results });

    })

  } catch (error) {

    console.log(error + ‘\n\nERROR getting ratings’);

    return response.json({ error: ‘Internal Server Error’ });

  }

};

module.exports.updateRatings = async (request, response) => {

  const { jobID, freeLancerID, clientID, ratings, feedback } = request.body;

  console.log(‘Updating Ratings’)

  try {

    const updatedRating = await Ratings.findOneAndUpdate(

      {

        jobID: ObjectId(jobID),

        freeLancerID: ObjectId(freeLancerID),

        userID: ObjectId(clientID),

      },

      {

        $set: {

          ratings,

          feedback,

        },

      },

      { new: true } // Return the updated document

    );

    if (!updatedRating) {

      return response.json({ error: “Rating not found” });

    }

    return response.json({ updatedRating, message: ‘Thank you for your new feedback!’ });

  } catch (error) {

    console.log(error + “\n\nERROR updating ratings”);

    return response.json({ error: “Internal Server Error” });

  }

};

**Client-Side Programming**

**CSS**

***About Us Page***

.about-main-container {

    background-color: linear-gradient(0deg, rgba(237,255,246,1) 76%, rgba(255,255,255,1) 100%);

    height: 100vh;

}

.container {

    display: flex;

    justify-content: space-around;

    max-width: 95%;

    margin: 0 auto;

    align-items: center;

    flex-direction: row;

}

.title-item {

    font-size: 50px;

    text-transform: capitalize;

    margin-left: 6%;

    margin-top: 2%;

}

.title-item h1 {

    color: grey;

}

.image-item {

    width: 40%;

    height: 30rem;

    display: flex;

    justify-content: center;

    align-items: center;

    overflow: hidden;

    /\* border: 2px solid black; \*/

    margin-left: 2%

}

.image-item img {

    width: 100%;

    object-fit: contain;

}

.info-item {

    width: 40%;

    /\* height: 100%; \*/

    /\* border: 2px solid black; \*/

    display: flex;

    margin-right: 15%;

    height: 20rem;

    justify-content: center;

    align-items: center;

}

***Admin Pages***

**Main Page.**

#admin {

    background-color: #f2f2f2;

    margin-top: 20px;

    padding: 20px;

  }

  .admin-header {

    background-color: #fff;

    border-bottom: 1px solid #ccc;

    padding: 10px 20px;

    margin-bottom: 20px;

  }

  .admin-header h1 {

    font-size: 24px;

    margin: 0;

  }

  .admin-header a {

    color: #555;

    text-decoration: none;

  }

  .admin-header a:hover {

    color: #000;

  }

  .welcome-msg {

    background-color: #fff;

    border: 1px solid #ccc;

    padding: 10px 20px;

    margin-bottom: 20px;

  }

  .add-info {

    background-color: #fff;

    border: 1px solid #ccc;

    padding: 20px;

    margin-bottom: 20px;

  }

  .add-info table {

    width: 100%;

    border-collapse: collapse;

  }

  .add-info th {

    text-align: left;

    padding: 8px;

    background-color: #f2f2f2;

    border-bottom: 1px solid #ddd;

  }

  .add-info td {

    padding: 8px;

    border-bottom: 1px solid #ddd;

  }

  .add-info a {

    color: #007bff;

  }

  .add-info a:hover {

    color: #0056b3;

  }

**Admin Page Forms.**

#about-content, .serviceform {

    display: flex;

    flex-direction: column;

    align-items: left;

    background-color: #ffffff;

    padding: 20px;

    border-radius: 8px;

    width: 30%;

    margin: 10% auto;

  }

  #about-content label {

    margin-bottom: 10px;

    font-weight: bold;

  }

  #about-content input[type="text"] {

    width: 100%;

    padding: 8px;

    border: 1px solid #ccc;

    border-radius: 4px;

    margin-bottom: 10px;

  }

  #about-content button, .serviceform button {

    padding: 8px 16px;

    background-color: #3e7258;

    color: #ffffff;

    border: none;

    border-radius: 4px;

    cursor: pointer;

  }

  #about-content button:hover, .serviceform button:hover {

    background-color: #45a049;

  }

  .service-container {

    display: flex;

    flex-direction: column;

    align-items: center;

    margin-top: 20px;

  }

  .serviceform label {

    margin-bottom: 10px;

    font-weight: bold;

  }

  .serviceform input[type="text"],

  .serviceform input[type="file"] {

    width: 100%;

    padding: 8px;

    border: 1px solid #ccc;

    border-radius: 4px;

    margin-bottom: 10px;

  }

**User Info Display.**

#user-info {

    width: 100%;

    border-collapse: collapse;

  }

  #user-info th,

  #user-info td {

    padding: 8px;

    text-align: left;

    border-bottom: 1px solid #ddd;

  }

  #user-info th {

    background-color: #f2f2f2;

  }

***Home Page***

@import url("https://fonts.googleapis.com/css2?family=Roboto+Condensed&display=swap");

@import url("https://fonts.googleapis.com/css2?family=Mukta:wght@600&display=swap");

\* {

  padding: 0;

  margin: 0;

}

html {

  scroll-behavior: smooth;

}

body {

  overflow-x: hidden;

  font-family: "Roboto Condensed", sans-serif;

}

.header {

  position: absolute;

  width: 100vw;

  display: flex;

  flex-wrap: no-wrap;

}

.header .logged\_out-navbar {

  display: flex;

  z-index: 1;

  margin-left: 68%;

  margin-top: 1%;

  background-color: transparent;

}

.header .logged\_out-navbar ul {

  display: flex;

  list-style: none;

  margin: -6% 0 0 -8%;

  padding: 0;

}

.header .logged\_out-navbar ul li a {

  text-decoration: none;

  color: white; /\* change color to white \*/

  font-family: "Roboto Condensed", sans-serif; /\* specify the font-family \*/

  font-size: 1.8rem; /\* you can adjust the font size \*/

}

.header .red-gradient {

  position: absolute;

  background: linear-gradient(51deg, rgb(214, 27, 98) 0%, rgb(214, 27, 98) 10%, rgb(133, 186, 56) 100%, rgb(133, 186, 56) 100%);

  width: 100%;

  height: 30vh;

}

.header .red-gradient .logo {

  height: 100%;

  width: 15%;

  -o-object-fit: contain;

     object-fit: contain;

  position: absolute;

  margin-top: -5%;

  margin-left: 6.5%;

  z-index: 3;

}

.header .red-gradient .title {

  font-weight: 600px;

  font-family: Arial;

  font-size: 2rem;

  position: absolute;

  top: 7%;

  left: 7.5%;

  color: white;

}

.header .red-gradient form {

  position: relative;

  z-index: 1;

  width: 100%;

}

.header .red-gradient form .category {

  position: absolute;

  margin-left: -2%;

  margin-top: -2%;

  height: 50px;

  width: 80px;

  background: grey;

  border-radius: 32px;

  border: none;

  padding-left: 2%;

  outline: none;

  font-size: 1.3rem;

}

.header .red-gradient form .tagline {

  font-weight: 600px;

  font-family: Arial;

  font-size: 1.2rem;

  position: absolute;

  top: 44%;

  left: 7%;

  color: white;

}

.header .red-gradient form .search {

  position: absolute;

  left: 90px;

  top: 80px;

  height: 50px;

  width: 300px;

  background: white;

  border-radius: 32px;

  border: none;

  padding-left: 2%;

  outline: none;

  font-size: 1.3rem;

}

.header .red-gradient form .category {

  position: absolute;

  left: 430px;

  top: 110px;

  height: 50px;

  width: 150px;

  background: white;

  border-radius: 1rem;

  border: none;

  padding-left: 0.5%;

  outline: none;

  font-size: 1rem;

}

.header .red-gradient form .search-btn {

  font-size: 1.5rem;

  position: absolute;

  margin-left: 23%;

  margin-top: 5.8%;

  border: none !important;

  background-color: transparent;

  width: 0;

  outline: none;

}

.home-main-container {

  margin-top: 14%;

  position: absolute;

  background: linear-gradient(0deg, rgb(255, 255, 255) 75%, rgb(241, 253, 247) 100%);

}

.home-main-container .flexible-services {

  position: relative;

  width: 100vw;

  padding-left: 4%;

}

.home-main-container .flexible-services h2 {

  font-weight: 600px;

  font-family: "Mukta", sans-serif;

  font-size: 2.5rem;

  margin-top: 2%;

  margin-left: 2.5%;

  height: 2%;

  text-align: left;

  color: rgb(82, 71, 71);

}

.home-main-container .flexible-services .services {

  display: flex;

  flex-wrap: wrap;

  width: 100vw;

}

.home-main-container .flexible-services .services .service, .home-main-container .flexible-services .services .service-img {

  border-radius: 0.5rem;

}

.home-main-container .flexible-services .services .service {

  display: inline-block;

  margin-top: 2%;

  margin-left: 2.5%;

  position: relative;

  width: 20%;

  height: 40vh;

  cursor: pointer;

  overflow: hidden;

  background-color: rgb(98, 98, 98);

}

.home-main-container .flexible-services .services .service .service-img {

  width: 100%;

  height: 100%;

  -o-object-fit: cover;

     object-fit: cover;

  transition: transform 0.3s ease-in-out;

}

.home-main-container .flexible-services .services .service .service-img:hover {

  opacity: 80%;

  transform: scale(1.1);

}

.home-main-container .flexible-services .services .service .service-title, .home-main-container .flexible-services .services .service .service-title-span, .home-main-container .flexible-services .services .service .service-tagline {

  position: absolute;

  z-index: 2;

  font-weight: 600px;

  font-family: Arial;

  margin-left: 5%;

  color: white;

  text-shadow: 2px 1px 2px black;

}

.home-main-container .flexible-services .services .service .service-title {

  margin-top: 65%;

  font-size: 1.3rem;

}

.home-main-container .flexible-services .services .service .service-tagline {

  margin-top: 73%;

  font-size: 1.3rem;

}

.home-main-container .flexible-services .services .service .service-title-span {

  margin-top: 72%;

  font-size: 1.8rem;

  font-weight: bold;

}

.home-main-container .flexible-services .services .service #transportation-title {

  margin-top: 65%;

}

.home-main-container .flexible-services .services .service #transportation-span {

  margin-top: 72%;

}

.home-main-container .reassurance-flex-box {

  display: flex;

  flex-wrap: nowrap;

  background: linear-gradient(0deg, rgb(237, 255, 246) 76%, rgb(255, 255, 255) 100%);

  padding-left: 10%;

  padding-top: 10%;

}

.home-main-container .reassurance-flex-box .text {

  width: 35%;

}

.home-main-container .reassurance-flex-box .text h3 {

  color: #404145;

  font-weight: 700;

  font-size: 32px;

  line-height: 38px;

}

.home-main-container .reassurance-flex-box .text ul {

  list-style-type: none;

}

.home-main-container .reassurance-flex-box .text ul .sub-heading {

  margin-bottom: 2%;

}

.home-main-container .reassurance-flex-box .text ul .bold {

  color: #404145;

  font-weight: 700;

  font-size: 18px;

  line-height: 25px;

  font-style: bold;

}

.home-main-container .reassurance-flex-box .text ul .info {

  color: #62646a;

  font-weight: 400;

  font-size: 18px;

  line-height: 26px;

  font-style: bold;

}

.home-main-container .reassurance-flex-box .image-container {

  padding-left: 5%;

  padding-top: 2%;

  width: 50%;

}

.home-main-container .reassurance-flex-box .image-container .image {

  width: 100%;

}

.home-main-container .getStarted {

  background-color: #3e7258;

  padding-top: 5%;

  padding-left: 10%;

  margin-left: 10%;

  margin-right: 10%;

  margin-top: 5%;

  display: flex;

  flex-wrap: nowrap;

  height: 55vh;

}

.home-main-container .getStarted h5 {

  font-size: 2rem;

  color: white;

  text-align: left;

}

.home-main-container .getStarted .getStarted-btn {

  padding: 3%;

  border: none;

  border-radius: 0.5rem;

  font-size: 1.1rem;

  color: black;

  background-color: white;

  transition: transform 0.3s ease-in-out;

  cursor: pointer;

}

.home-main-container .getStarted .getStarted-btn a {

  text-decoration: none;

  color: black;

}

.home-main-container .getStarted .getStarted-btn:hover {

  transform: scale(1.1);

}

.home-main-container .getStarted .image {

  margin-top: -5%;

  margin-left: 5%;

  overflow: hidden;

}

.home-main-container .getStarted .overlay {

  position: fixed;

  top: 0;

  left: 0;

  width: 100%;

  height: 100%;

  background-color: rgba(0, 0, 0, 0.5);

  z-index: 1;

}

.home-main-container .getStarted .hidden {

  display: none;

}

.home-main-container .getStarted .login-form, .home-main-container .getStarted .form-hidden {

  background-color: white;

  position: absolute;

  height: 50vh;

  width: 25vw;

  margin-top: -30%;

  margin-left: 15%;

  font-family: "Mukta", sans-serif;

  padding: 2%;

  display: none;

  z-index: 2;

}

.home-main-container .getStarted .login-form .join-new-h2, .home-main-container .getStarted .form-hidden .join-new-h2 {

  text-align: center;

  margin-bottom: 5%;

}

.home-main-container .getStarted .login-form .form-visible, .home-main-container .getStarted .form-hidden .form-visible {

  display: flex;

  flex-direction: column;

  justify-content: center;

  align-items: center;

}

.home-main-container .getStarted .login-form .form-visible .join-with-google-button, .home-main-container .getStarted .form-hidden .form-visible .join-with-google-button {

  background-color: white;

  font-size: 1.2rem;

  border: 1px solid grey;

  border-radius: 0.3rem;

  padding: 3%;

  position: relative;

}

.home-main-container .getStarted .login-form .form-visible .join-with-google-button .google-svg, .home-main-container .getStarted .form-hidden .form-visible .join-with-google-button .google-svg {

  margin-left: -80%;

  margin-top: 2%;

}

.home-main-container .getStarted .login-form .form-visible .join-with-google-button span, .home-main-container .getStarted .form-hidden .form-visible .join-with-google-button span {

  position: absolute;

  margin-left: 15%;

}

.home-main-container .getStarted .login-form .form-visible .or-text, .home-main-container .getStarted .form-hidden .form-visible .or-text {

  width: 80%;

  display: flex;

  align-items: center;

}

.home-main-container .getStarted .login-form .form-visible .or-text .line, .home-main-container .getStarted .form-hidden .form-visible .or-text .line {

  display: inline;

  border: none;

  border-top: 1px solid grey;

  margin: 10px 0;

  width: 50%;

}

.home-main-container .getStarted .login-form .form-visible .OR, .home-main-container .getStarted .form-hidden .form-visible .OR {

  color: grey;

  margin-top: 5%;

  margin-bottom: 5%;

  margin-left: 1%;

  margin-right: 1%;

  font-size: 1.2rem;

}

.home-main-container .getStarted .login-form .form-visible .continue-button, .home-main-container .getStarted .form-hidden .form-visible .continue-button {

  background-color: rgb(29, 191, 115);

  color: white;

  font-size: 1.2rem;

  border: 1px solid rgb(29, 191, 115);

  border-radius: 0.3rem;

  padding: 3%;

  margin-bottom: 2%;

}

.home-main-container .getStarted .login-form .form-visible button, .home-main-container .getStarted .form-hidden .form-visible button {

  width: 80%;

  cursor: pointer;

}

.home-main-container .getStarted .login-form .form-visible input[type=email], .home-main-container .getStarted .form-hidden .form-visible input[type=email] {

  width: 80%;

  padding-top: 2%;

  padding-bottom: 2%;

  padding-left: 2%;

  font-size: 1rem;

  margin-bottom: 5%;

  border: 1px solid grey;

  border-radius: 0.3rem;

}

.home-main-container .getStarted .form-hidden {

  display: flex;

  flex-direction: column;

  justify-content: center;

  align-items: center;

  margin-top: -92%;

  margin-left: -4.5%;

  height: 50vh;

  width: 26.5vw;

  border-bottom: 1px solid grey;

}

.home-main-container .getStarted .form-hidden .join-new-h2 {

  margin-top: -10%;

}

.home-main-container .getStarted .form-hidden button {

  width: 80%;

  cursor: pointer;

}

.home-main-container .getStarted .form-hidden input {

  width: 80%;

  padding-top: 2%;

  padding-bottom: 2%;

  padding-left: 2%;

  font-size: 1rem;

  margin-bottom: 5%;

  border: 1px solid grey;

  border-radius: 0.3rem;

}

.home-main-container .getStarted .form-hidden .username {

  margin-top: 0%;

}

.home-main-container .getStarted .form-hidden .password-validation {

  margin-top: -5%;

  margin-bottom: 5%;

  margin-left: -10%;

  font-size: 0.88rem;

  color: grey;

  width: 70%;

}

.home-main-container .getStarted .form-hidden .join-button {

  background-color: rgb(29, 191, 115);

  color: white;

  font-size: 1.2rem;

  border: 1px solid rgb(29, 191, 115);

  border-radius: 0.3rem;

  padding: 3%;

  margin-bottom: 2%;

}/\*# sourceMappingURL=index.css.map \*/

***Login/ Signup Page***

.login-main-container .loginForm, .login-main-container .signupForm, .signup-main-container .loginForm, .signup-main-container .signupForm {

  width: 30%;

  margin: 10% auto;

}

.login-main-container .loginForm .inputs, .login-main-container .signupForm .inputs, .signup-main-container .loginForm .inputs, .signup-main-container .signupForm .inputs {

  background-color: white;

  color: black;

}

.login-main-container .loginForm .link, .login-main-container .signupForm .link, .signup-main-container .loginForm .link, .signup-main-container .signupForm .link {

  font-size: 1rem;

}

.login-main-container .submit-btn, .signup-main-container .submit-btn {

  width: 100%;

  background-color: #3e7258;

}/\*# sourceMappingURL=members.css.map \*/

**JavaScript**

***About Us Page***

  return(

    <>

      <GlobalNavbar/>

      <main className="about-main-container">

        <h1 className='title-item'>About NE1-FREELANCE</h1>

          <div className ="container">

            <div className ="image-item">

              <img src = {Person\_Img} alt='Person'/>

            </div>

            <div className ="info-item">

              {information && information.length >= 1 ? (

                <p>{information}</p>)

                :(

                  <p>No Information Available</p>

              )}

            </div>

          </div>

      </main>

      <GlobalFooter/>

    </>

  )

***Home Page***

**Home Page.**

return(

        <div className="Home">

            {/\* Banner used to search for jobs  \*/}

            <Header/>

            <Main />

        </div>

    )

**Header.**

return (

        <header className="header">

          <Navbar />

          {/\* This is the red portion of the curve blending into green \*/}

          <div className="red-gradient">

              {/\* Logo of the website \*/}

              <img className = "logo" src={Logo} alt="logo" />

              {/\* Title of the website  \*/}

              {/\* <h1 className="title"><b>Freelance</b></h1> \*/}

              <form className="search\_form" onSubmit={handleSearchSubmit}>

                  {/\* {% csrf\_token %} \*/}

                  <input className="search" type="text" placeholder="Search for job..." name="search" required

                    onChange={(event)=> setSearch(event.target.value)}

                  />

                  <select name="category" className="category" defaultValue="All Categories" onChange={(e) => setJobCategory(e.target.value)}>

                      <option value="">All categories</option>

                      {jobCategories.map(category => {

                        return(

                             <option value={category.name} key={category.name}>{category.name}</option>

                        )

                      })}

                  </select>

                  <button className="search-btn" type="submit" title="search button">

                      <i className="fa fa-search"></i> {/\*Search icon\*/}

                  </button>

              </form>

          </div>

      </header>

    )

**Main.**

return(

                // Main container to hold everything

        <main className="home-main-container">

        {/\* flexbox container that displays 4 SERVICES \*/}

        <ServicesBanner/>

            {/\* text ABOUT THE SERVICES \*/}

            <Reassurance />

                {/\* Login/Sign up banner \*/}

                <GetStarted />

        <br/><br/>

        </main>

    )

**Reassurance.**

return(

        <div className="reassurance-flex-box">

                <div className="text">

                    <h3>An abundance of freelance skill readily accessible at your disposal</h3>

                    <br/>

                    <ul>

                        <li className="sub-heading">

                            <p className="bold">

                                {/\* Checkbox svg image \*/}

                                <svg width="16" height="16" viewBox="0 0 16 16" xmlns="http://www.w3.org/2000/svg">

                                    <path d="M8 1.75C4.54822 1.75 1.75 4.54822 1.75 8C1.75 11.4518 4.54822 14.25 8 14.25C11.4518 14.25 14.25 11.4518 14.25 8C14.25 4.54822 11.4518 1.75 8 1.75ZM0.25 8C0.25 3.71979 3.71979 0.25 8 0.25C12.2802 0.25 15.75 3.71979 15.75 8C15.75 12.2802 12.2802 15.75 8 15.75C3.71979 15.75 0.25 12.2802 0.25 8Z"></path>

                                    <path d="M11.5303 5.46967C11.8232 5.76256 11.8232 6.23744 11.5303 6.53033L7.53033 10.5303C7.23744 10.8232 6.76256 10.8232 6.46967 10.5303L4.46967 8.53033C4.17678 8.23744 4.17678 7.76256 4.46967 7.46967C4.76256 7.17678 5.23744 7.17678 5.53033 7.46967L7 8.93934L10.4697 5.46967C10.7626 5.17678 11.2374 5.17678 11.5303 5.46967Z"></path>

                                </svg> The finest for every financial plan</p>

                            <p className="info">Obtain superior services at every budget level. No hourly charges, solely project-based pricing.</p>

                        </li>

                        <li className="sub-heading">

                            <p className="bold">

                                {/\* Checkbox svg image \*/}

                                <svg width="16" height="16" viewBox="0 0 16 16" xmlns="http://www.w3.org/2000/svg">

                                    <path d="M8 1.75C4.54822 1.75 1.75 4.54822 1.75 8C1.75 11.4518 4.54822 14.25 8 14.25C11.4518 14.25 14.25 11.4518 14.25 8C14.25 4.54822 11.4518 1.75 8 1.75ZM0.25 8C0.25 3.71979 3.71979 0.25 8 0.25C12.2802 0.25 15.75 3.71979 15.75 8C15.75 12.2802 12.2802 15.75 8 15.75C3.71979 15.75 0.25 12.2802 0.25 8Z"></path>

                                    <path d="M11.5303 5.46967C11.8232 5.76256 11.8232 6.23744 11.5303 6.53033L7.53033 10.5303C7.23744 10.8232 6.76256 10.8232 6.46967 10.5303L4.46967 8.53033C4.17678 8.23744 4.17678 7.76256 4.46967 7.46967C4.76256 7.17678 5.23744 7.17678 5.53033 7.46967L7 8.93934L10.4697 5.46967C10.7626 5.17678 11.2374 5.17678 11.5303 5.46967Z"></path>

                                </svg> Exemplary work executed expeditiously</p>

                            <p className="info">Identify the appropriate freelancer to commence work on your venture within minutes.</p>

                        </li>

                        <li className="sub-heading">

                            <p className="bold">

                                {/\* Checkbox svg image \*/}

                                <svg width="16" height="16" viewBox="0 0 16 16" xmlns="http://www.w3.org/2000/svg">

                                    <path d="M8 1.75C4.54822 1.75 1.75 4.54822 1.75 8C1.75 11.4518 4.54822 14.25 8 14.25C11.4518 14.25 14.25 11.4518 14.25 8C14.25 4.54822 11.4518 1.75 8 1.75ZM0.25 8C0.25 3.71979 3.71979 0.25 8 0.25C12.2802 0.25 15.75 3.71979 15.75 8C15.75 12.2802 12.2802 15.75 8 15.75C3.71979 15.75 0.25 12.2802 0.25 8Z"></path>

                                    <path d="M11.5303 5.46967C11.8232 5.76256 11.8232 6.23744 11.5303 6.53033L7.53033 10.5303C7.23744 10.8232 6.76256 10.8232 6.46967 10.5303L4.46967 8.53033C4.17678 8.23744 4.17678 7.76256 4.46967 7.46967C4.76256 7.17678 5.23744 7.17678 5.53033 7.46967L7 8.93934L10.4697 5.46967C10.7626 5.17678 11.2374 5.17678 11.5303 5.46967Z"></path>

                                </svg> Safe and secure payments, every time</p>

                            <p className="info">Have peace of mind knowing the amount you will pay upfront. Payments are not released until you have given your approval of the work.</p>

                        </li>

                        <li className="sub-heading">

                            <p className="bold">

                                {/\* Checkbox svg image \*/}

                                <svg width="16" height="16" viewBox="0 0 16 16" xmlns="http://www.w3.org/2000/svg">

                                    <path d="M8 1.75C4.54822 1.75 1.75 4.54822 1.75 8C1.75 11.4518 4.54822 14.25 8 14.25C11.4518 14.25 14.25 11.4518 14.25 8C14.25 4.54822 11.4518 1.75 8 1.75ZM0.25 8C0.25 3.71979 3.71979 0.25 8 0.25C12.2802 0.25 15.75 3.71979 15.75 8C15.75 12.2802 12.2802 15.75 8 15.75C3.71979 15.75 0.25 12.2802 0.25 8Z"></path>

                                    <path d="M11.5303 5.46967C11.8232 5.76256 11.8232 6.23744 11.5303 6.53033L7.53033 10.5303C7.23744 10.8232 6.76256 10.8232 6.46967 10.5303L4.46967 8.53033C4.17678 8.23744 4.17678 7.76256 4.46967 7.46967C4.76256 7.17678 5.23744 7.17678 5.53033 7.46967L7 8.93934L10.4697 5.46967C10.7626 5.17678 11.2374 5.17678 11.5303 5.46967Z"></path>

                                </svg> 24/7 assistance

                            </p>

                            <p className="info">Have queries? Our round-the-clock support team is on hand to assist you at any time, from any location."</p>

                        </li>

                    </ul>

                </div>

                <div className="image-container">

                    <img className="image" src="https://fiverr-res.cloudinary.com/q\_auto,f\_auto,w\_700,dpr\_1.0/v1/attachments/generic\_asset/asset/089e3bb9352f90802ad07ad9f6a4a450-1599517407052/selling-proposition-still-1400-x1.png" alt="" />

                </div>

        </div>

    )

**GetStarted.**

 return (

        <div className="getStarted">

        <div>

            <h5>Unlock your business's <i>potential</i> <br/>

                with the right talent!</h5>

                <br/>

            <button className="getStarted-btn" title="get started button"

                onClick={()=> navigate("/members/signup")}>Get Started

            </button>

        </div>

        <div className="image">

            <img src={Man} alt="Black guy in white jersey excited"/>

        </div>

</div>

    )

***Login/ Signup Pages***

**Login Page.**

 return(

      <main className = "login-main-container">

        <GlobalNavbar/>

          <form className="loginForm" onSubmit={(event) => handleSubmit(event)}>

          {message ? (

            <div className="alert alert-warning alert-dismissible fade show" role="alert">

                  <strong>

                      <i className="fa-solid fa-triangle-exclamation"></i>

                      {message}

                  </strong>

                  <button type="button" className="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>

              </div>

          ) : null

          }

          <h1>Login</h1>

          {/\* {% csrf\_token %} \*/}

          <div className="mb-3">

            <label htmlFor="email" className="form-label">Email:</label>

            <input type="email" className="form-control inputs" id="email" name = "email" placeholder="example@example.com"

             onChange={e => handleChange(e)} required

            />

          </div>

          <div className="mb-3">

            <label htmlFor="password" className="form-label">Password</label>

            <input type="password" className="form-control inputs" id="password" name = "password" placeholder="\*\*\*\*\*\*\*\*"

             onChange={e => handleChange(e)} required

            />

          </div>

          <div className="col-12">

            <p className="link">Need an Account?

              <Link to='/members/signup'>Signup</Link>

            </p>

          </div>

          <div className="col-12">

            <button className="btn btn-primary submit-btn" type="submit">Login</button>

          </div>

        </form> {/\* End Form \*/}

    </main> //  End Login Container

)}  // End stateless function component

**Signup Page.**

 return (

    <main className="signup-main-container">

      <GlobalNavbar />

      <form className="signupForm" onSubmit={handleSubmit}>

      {message ? (

          <div className="alert alert-warning alert-dismissible fade show" role="alert">

                  <strong><i className="fa-solid fa-triangle-exclamation"></i>{message}</strong>

                  <button type="button" className="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>

              </div>

        ) : null

      }

        <h1 className="signup-header">Sign up</h1>

        <div className="mb-3">

          <label htmlFor="username" className="form-label">Username:</label>

          <input type="text" className="form-control inputs" id="username" name="username" placeholder="John Doe"

            onChange={(e) => handleChange(e)} />

        </div>

        <div className="mb-3">

          <label htmlFor="email" className="form-label">Email:</label>

          <input type="email" className="form-control inputs" id="email" name="email" placeholder="example@example.com"

            onChange={(e) => handleChange(e)} />

        </div>

        <div className="mb-3">

          <label htmlFor="password1" className="form-label">Password</label>

          <input type="password" className="form-control inputs" id="password1" name="password" placeholder="\*\*\*\*\*\*\*\*"

             onChange={(e) => handleChange(e)} />

        </div>

        <div className="mb-3">

          <label htmlFor="password2" className="form-label">Confirm Password</label>

          <input type="password" className="form-control inputs" id="password2" name="confirmPassword" placeholder="\*\*\*\*\*\*\*\*"

             onChange={(e) => handleChange(e)} />

        </div>

        <div className="col-12">

          <p className="link">Already a member?

            <Link to='/members/login'>Login</Link>

          </p>

        </div>

        <div className="col-12">

          <button className="btn btn-primary submit-btn" type="submit">Sign up</button>

        </div>

      </form>

    </main>

  )

**Server-Side Programming**

const express = require("express");

const cors = require("cors");

const mongoose = require("mongoose");

mongoose.set('strictQuery', false);

const authRoutes = require("./routes/auth");

const messageRoutes = require("./routes/messages");

const app = express();

const socket = require("socket.io");

require("dotenv").config();

const stripe = require("stripe")('sk\_test\_4eC39HqLyjWDarjtT1zdp7dc');

// app.listen(4242, () => console.log("Node server listening on port 4242!"));

app.use(express.json());

app.use(cors({

  origin: process.env.NE1\_FREELANCE\_URL,

  credentials: true

}));

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

app.use(express.json());

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

mongoose

  .connect(process.env.MONGO\_URL, {

    useNewUrlParser: true,

    useUnifiedTopology: true,

  })

  .then(() => {

    console.log("DB Connection Successfull");

  })

  .catch((err) => {

    console.log("ERROR CONNECTION: " +err.message);

  });

app.use("/api/auth", authRoutes);

app.use("/api/messages", messageRoutes);

const server = app.listen(process.env.MONGO\_PORT, () =>

  console.log(`Mongoose Server started on ${process.env.MONGO\_PORT}`)

);

const io = socket(server, {

  cors: {

    origin: "http://localhost:3000",

    credentials: true,

  },

});

global.onlineUsers = new Map();

io.on("connection", (socket) => {

  global.chatSocket = socket;

  socket.on("add-user", (userId) => {

    onlineUsers.set(userId, socket.id);

  });

  socket.on("send-msg", (data) => {

    const sendUserSocket = onlineUsers.get(data.to);

    if (sendUserSocket) {

      socket.to(sendUserSocket).emit("msg-recieve", data.msg);

    }

  });

});

const calculateOrderAmount = (items) => {

  // Replace this constant with a calculation of the order's amount

  // Calculate the order total on the server to prevent

  // people from directly manipulating the amount on the client

  return 1400;

};

app.post("/create-payment-intent", async (req, res) => {

  const { items } = req.body;

  // Create a PaymentIntent with the order amount and currency

  const paymentIntent = await stripe.paymentIntents.create({

    amount: calculateOrderAmount(items),

    currency: "usd",

    automatic\_payment\_methods: {

      enabled: true,

    },

  });

  res.send({

    clientSecret: paymentIntent.client\_secret,

  });

});