JCN: A Social Network for Job seekers, Recruiters and Employers

Wardatul Akmam
Id: 012171008

A Project

in

The Department

of

Computer Science and Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Science in Computer Science and Engineering

United International University

Dhaka, Bangladesh

December 2019

Approval Certificate

This project titled "**JCN- A Social Network for Job Seekers, Recruiters and Employers**" submitted by **Wardatul Akmam**, Student ID: **012171008**, has been accepted as Satisfactory in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering on December, 2019.

Board of Examiners

1.
_____ Supervisor
D**r. Mohammad Nurul Huda**
Professor & Director-MSCSE
Department of Computer Science & Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

2.
_____ Examiner
Rubaiya Rahtin khan
Assistant Professor
Department of Computer Science & Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

3.
_____ Ex-Officio
D**r. Mohammad Nurul Huda**
Professor & Director-MSCSE
Department of Computer Science & Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

Declaration

This is to certify that the work entitled "**JCN- A Social Network for Job Seekers, Recruiters and Employers**" is the outcome of the project carried out by me under the supervision of Dr. Mohammad Nurul Huda, Professor and Director - MSCSE, United International University (UIU), Dhaka, Bangladesh.

_____

Wardatul Akmam
Department of Computer Science and Engineering
MSCSE Program
Student ID: 012171008
United International University (UIU)
Dhaka-1212, Bangladesh.

In my capacity as supervisor of the candidate's project, I certify that the above statements are true to the best of my knowledge.

_____

D**r. Mohammad Nurul Huda**
Professor & Director-MSCSE
Department of Computer Science & Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

ABSTRACT

The project JCN is a web application which brings the professionals like job seekers employers and recruiters in one platform, easy communication with each other, sharing ideas, application in jobs and everything else. Employees are important resources for any company and selecting good candidates leads nothing but to success. Selection of right candidate is not so easy. Though our country is over populated, finding skilled and right candidate for any position is still very difficult. This platform can be used for publishing job opportunities, Dropping CV, creating and growing network of professionals, messaging, finding different types of companies and also skilled people regarding his criteria. It will bring skilled people and great companies in one network. So, JCN will be a very useful platform for all types of professionals.

Acknowledgement

Firstly I'm Grateful to Almighty Allah for being finished the project successfully.

I want to thank my project supervisor, **Dr. Mohammad Nurul Huda**, Professor and Director-MSCSE, United International University for his continuous support, right direction, motivation and giving me the opportunity to do this project.

Finally, I'm thankful to my parents for their continuous support to me and encouragement they given me during the time period.

Table of Contents

LIST OF FIGURES

Chapter 1

Introduction

Employees are the very important resources of an organization as they directly contribute to its success and growth. In today's business era which is pioneered by constant and rapid changes, knowledge capital of the respective organization must be retained in order for it to be responsive and dynamic to the needs or desire of their investors.

This web application intends to provide a well-established web-based Social Network system between a job seeker and a recruiter. This documents a networking system scope, functionalities, requirements and feasibility. This project aims to develop a networking system or social media site. Where user will be able to make post, react on the post and communicate with each other through communication module.

1.1 Objectives

- To build a source of human resources
- Employee/employer communication
- Research tool for employer and employees
- Transparency of job seeker and recruiter

1.2 Features of Project

- Advertising job and reaching more people
- Building network of professionals
- Source of resources
- Reach a wider resources
- Real time friend requests, messaging
- Follow discussion on various recruitment topics

Chapter 2

System Analysis & Design

System analysis is the process of architecting a system which will be efficient, cost effective and scalable to meet the future need and able to replace the current system and meet the demand of the uses. System analysis is very important part to design a software. We should also keep in mind the current competition of the market. System Analysis is directed with the following objectives in mind:

- The development of a feasibility study, involving determining whether a project is economically, socially, technologically and organizationally feasible.
- Conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system.

2.1 JCN Requirements

- User will be able to sign up with their specific fields ex: Job seeker, employer, and recruiter.
- Email verification when sign up
- Forgot password
- Sending invitation to join network through emails
- Feed where user will be able to see the posts of others, give message and interact with them.
- There will be an option for shortlist a post if it seems important to the user
- There will also be a like option on each post
- There will be a user profile and timeline
- Option for creating different groups
- Search option for searching companies and candidates
- Notification when someone creates new posts or any friend requests
- Connection request option
- Option for remove or block contact

- Message module
- Message in each particular post
- Contacts module: add or remove contacts and group
- Settings module to update user information and privacy

2.2 Design Goals

Designing is the most significant phase of software development. It needs an attentive planning and thinking on the part of the system designer. Designing an application means how different parts of the software will be developed and how they will perform the desired goal. System designer need to design the software in a way so that it will be scalable, able to take increased loads to meet the demand of the user's new requirements.

After analyzing the requirements carefully each step of the software development should be carried out very carefully. Architecting. Designing, development, testing and deployment each and every part is very important to develop a software

The following goals were kept in mind while designing the software:

- Increase the brand value and engagement
- New leads Professionals
- Engaging more people with the brand

2.3 System Diagrams

2.3.1 Data Flow Diagram (DFD)

Level 0 Data Flow Diagram

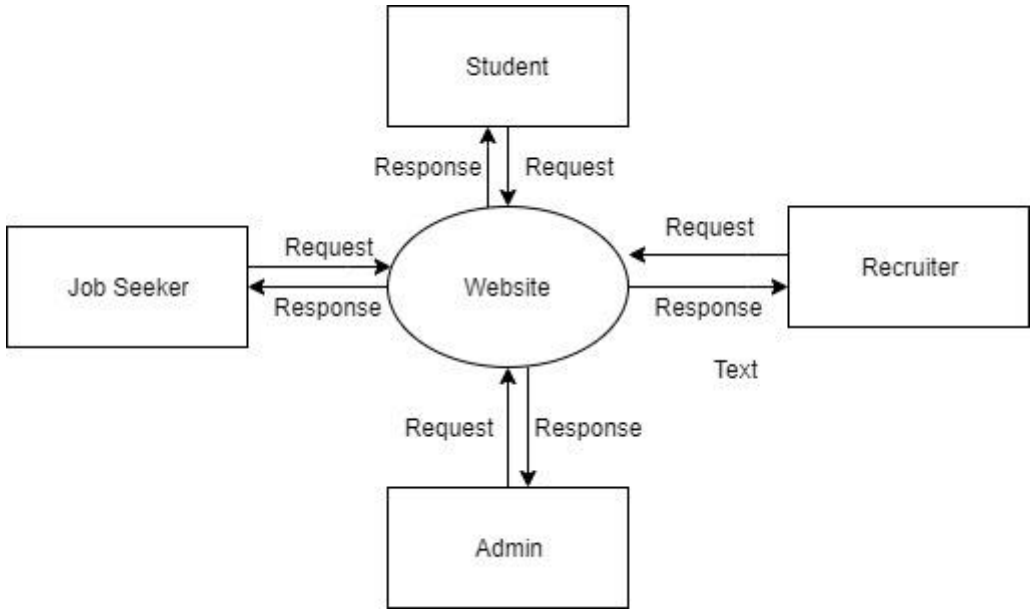

Figure 1: Zero Level Data Flow Diagram

Level 1 Data Flow Diagram:

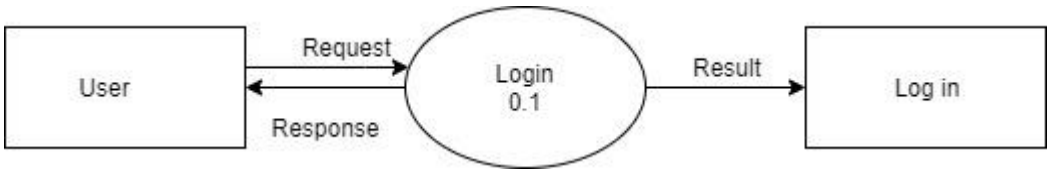

Figure 2: Level-1 Data Flow Diagram (a)

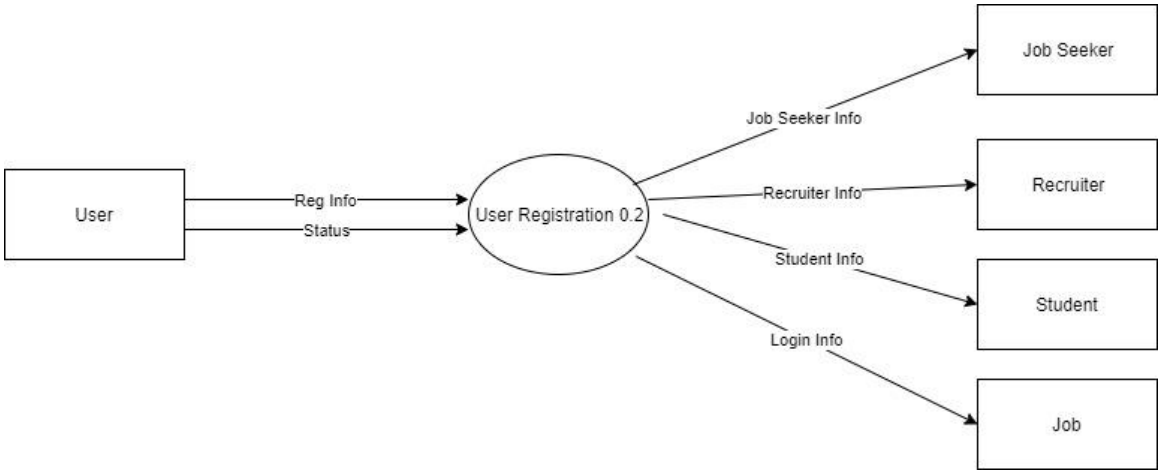

Figure 3: Level-1 Data Flow Diagram (b)

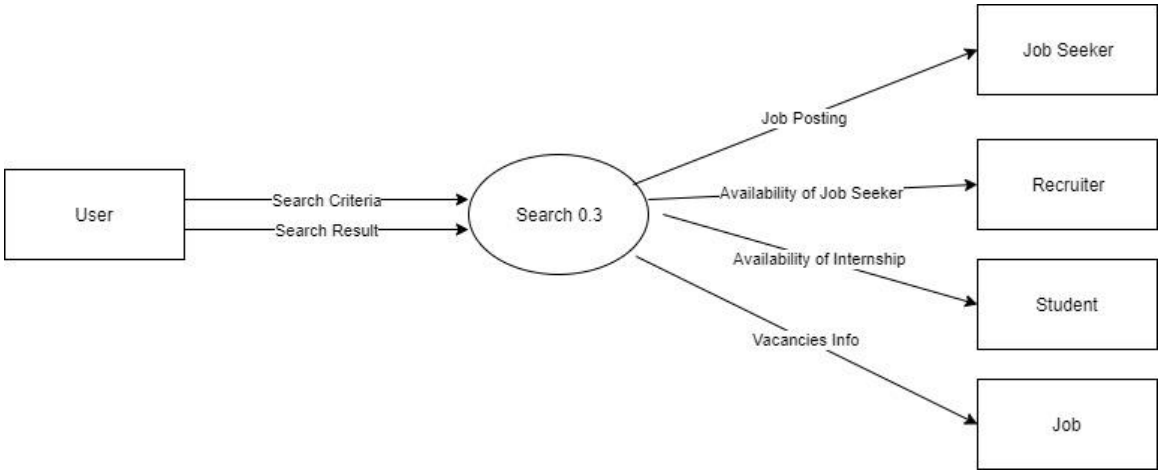

Figure 4: Level-1 Data Flow Diagram (c)

Level 2 Data Flow Diagram:

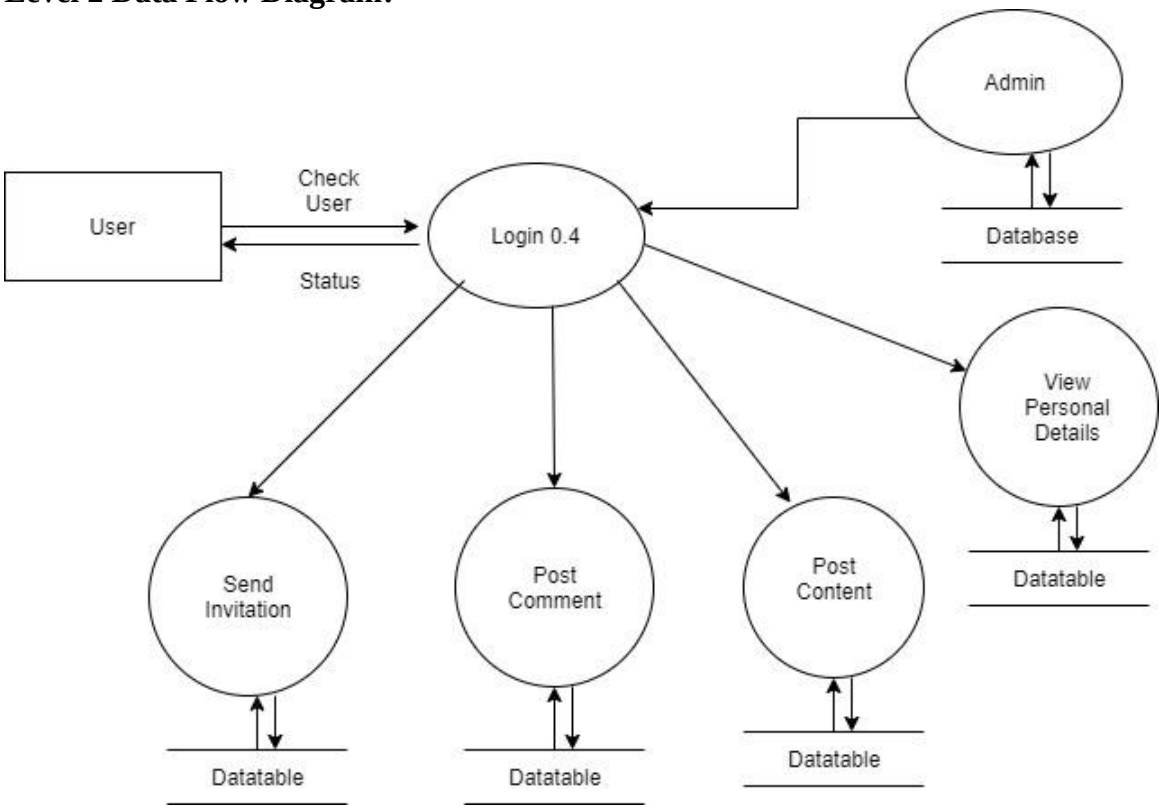

Figure 5: Level-2 Data Flow Diagram

2.3.2 Use Case Diagram

Login:

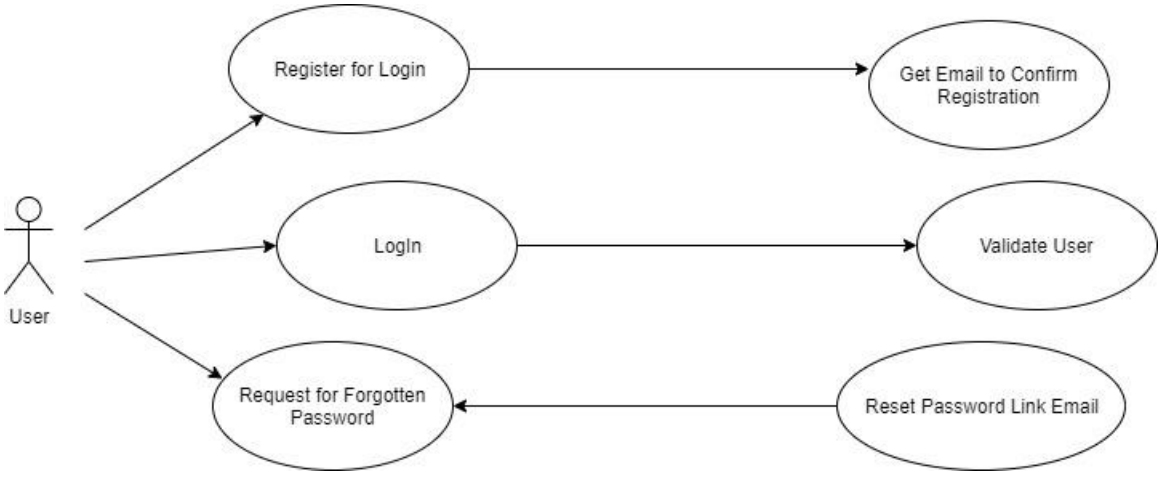

Figure 6: Use Case Diagram of Login System

Other Functionalities:

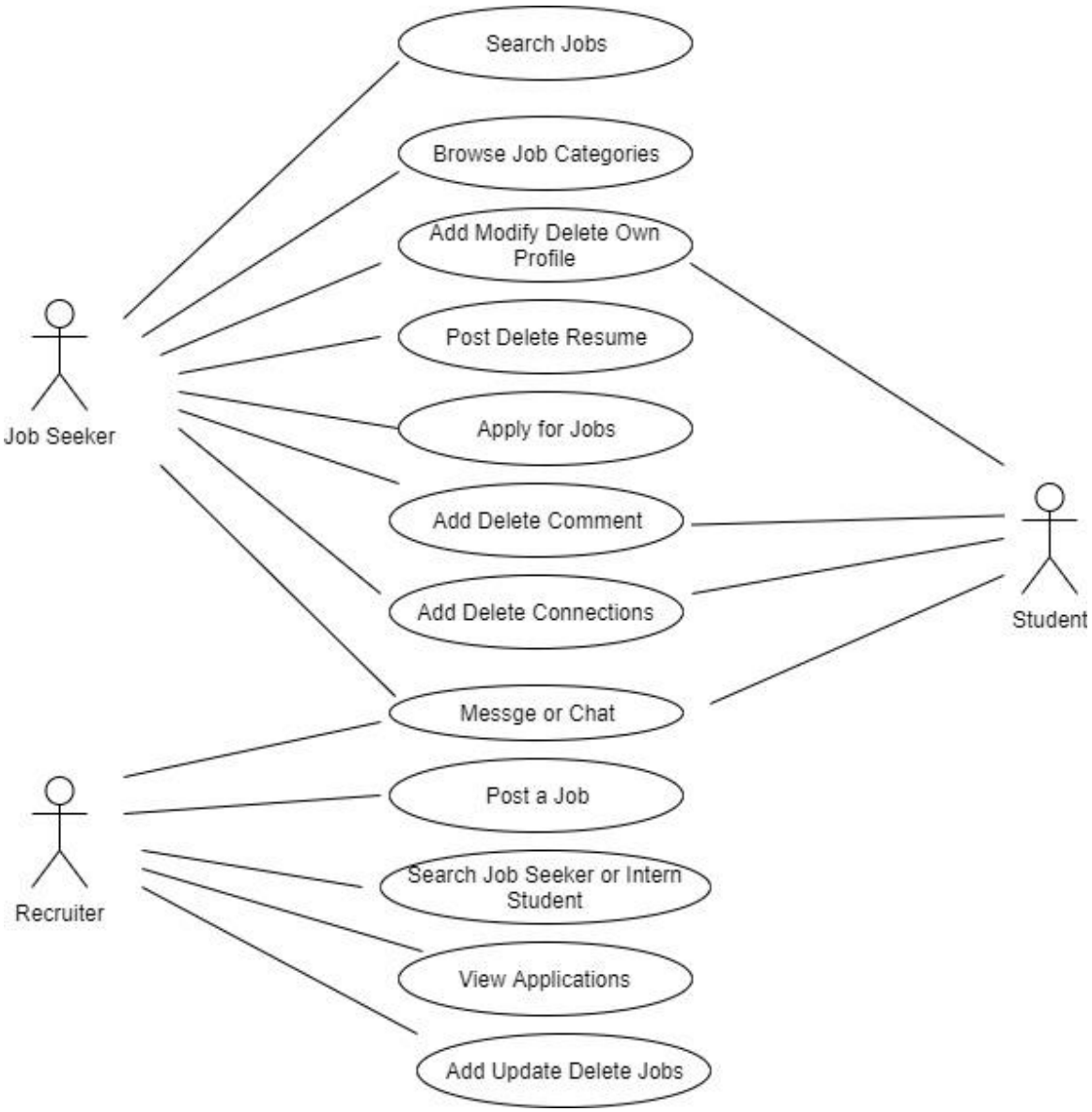

Figure 7: Use Case Diagram of Other Features

2.3.3 Sequence Diagram

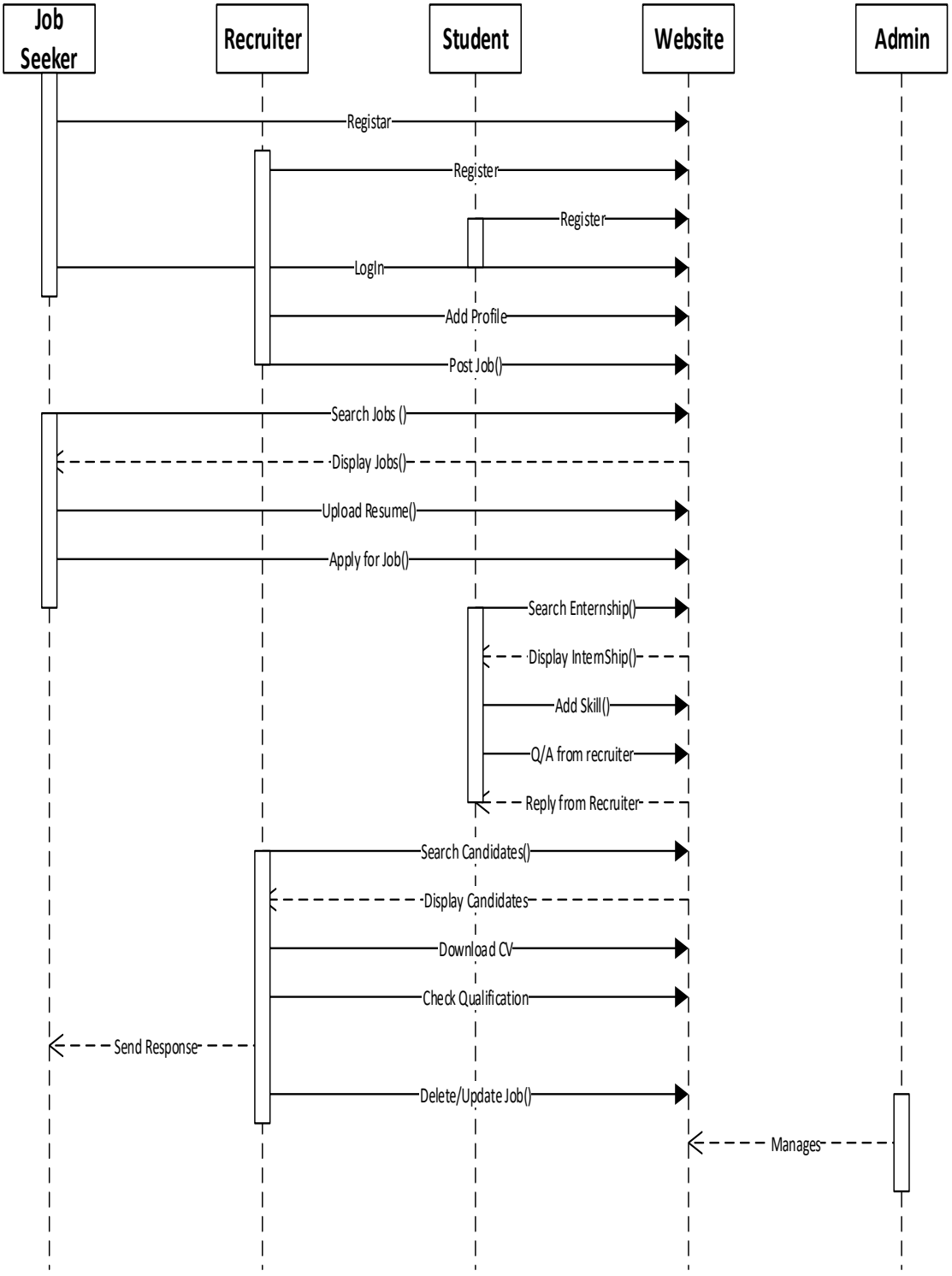

Figure 8: Sequence Diagram

2.3.4 Class Diagram

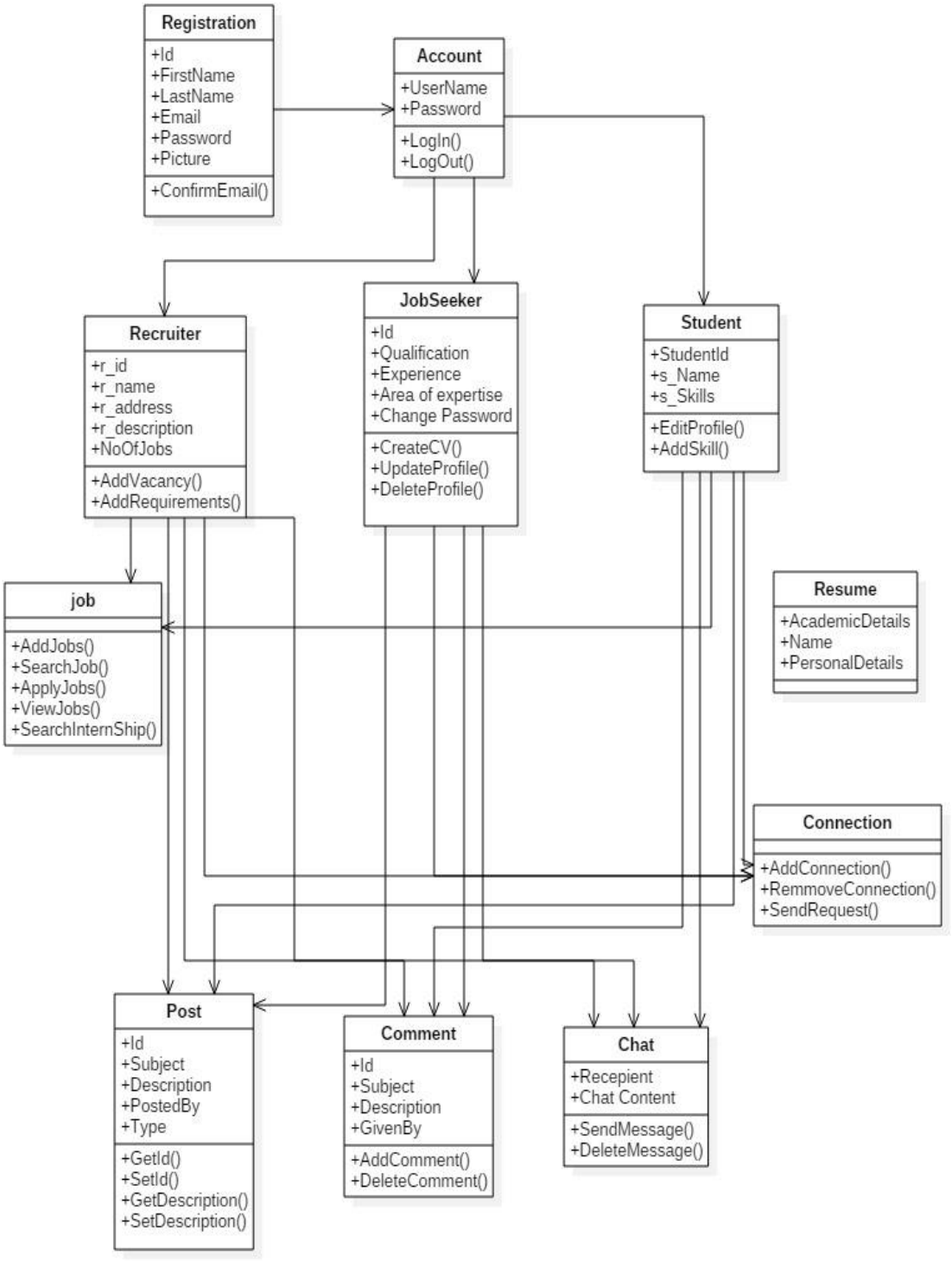

Figure 9: Class Diagram

2.4 Grant Chart

Select a period to highlight at right. A legend describing the charting follows.					**Period Highlight:**	**10/1/2017**
ACTIVITY	**PLAN START**	**PLAN DURATION**	**ACTUAL START**	**ACTUAL DURATION**	**PERCENT COMPLETE**	**PERIODS**
						1-Oct-17
Score Identification	1-Oct-18	27	27-Oct-17	27	**50%**	
Proposal Submission	28-Oct-18	1	28-Oct-17	1	**0%**	
Waiting for Approval	3-Nov-18	4	3-Nov-17	4	**0%**	
Requirement Analysis	15-Oct-18	40			**0%**	
Design & Document	24-Nov-18	25			**0%**	
Development	19-Dec-18	80			**0%**	
Testing	30-Jan-19	45			**0%**	
Deployment	15-Mar-19	5			**0%**	
Documentation	20-Mar-19	30			**0%**	
Project Submission	19-Apr-19	5			**0%**	
Post Project Activity	24-Apr-19	10			**0%**	

Figure 10: Gantt Chart

Chapter 3

Methodology

3.1 Introduction

In this section, discussion on the method used to develop the "JCN- A social network for job seekers, recruiters and employers" System will be discussed. The method used to develop the software will be explained in details.

3.2 Method to Develop JCN Software

For the development of JCN application I have followed the agile methodology. Agile is the best method to develop modern software nowadays. In agile we can develop a software in module by module basis and can develop multiple modules parallel. In agile we can analysis the requirement and can design, develop different module and functionality in parallel manner. This makes the development of any software faster, cleaner and organized. Requirements may change in any step during the development of software. In waterfall model it was very difficult to change any functionality or module once it has been developed and the stakeholders need to wait until the overall process has been finished which include the development of whole software, testing and development. Which is not flexible for the stakeholders. In agile scrum methodology is more favorable to developers than Kanban. In scrum development goal is set to sprint by sprint basis. In each sprint developers are given a goal and after the development testing and deployment also goes on at the same time. So agile is more efficient, faster software development methodology than others.

3.3 Identify the Software Development Tools to be Used

JCN have been tried to develop with modern technologies to make it perform faster with growing user. For database mongodb has been used. In the server side NodeJS has been used which is a javascript server side language and faster than most other language. In the front end angular js has been used. Invision has been used to develop the prototype. Saas and angular material has been used for the design purposes.

3.4 Screenshot of the Graphical User Interface

Sign up form is the two step form and this is the second step. Users can sign up with their specific criteria.

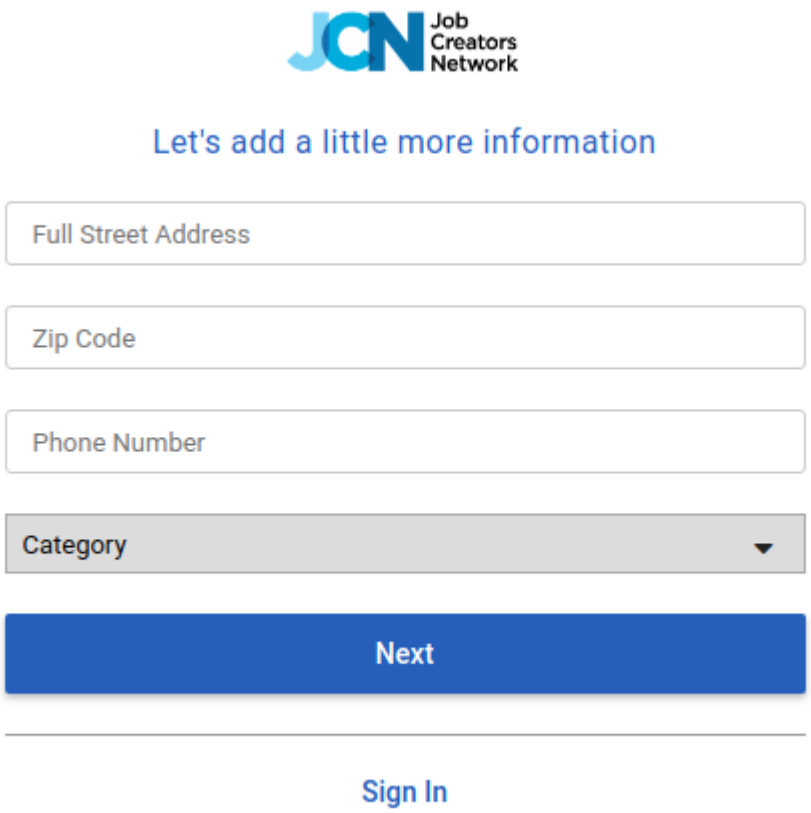

Figure 11: Graphical User Interface - Signup Form

Following is the login form where user need to sign in with their email and passwords. There is also an option for checking robots. Forgot password option is also available in case of any user forgets his password.

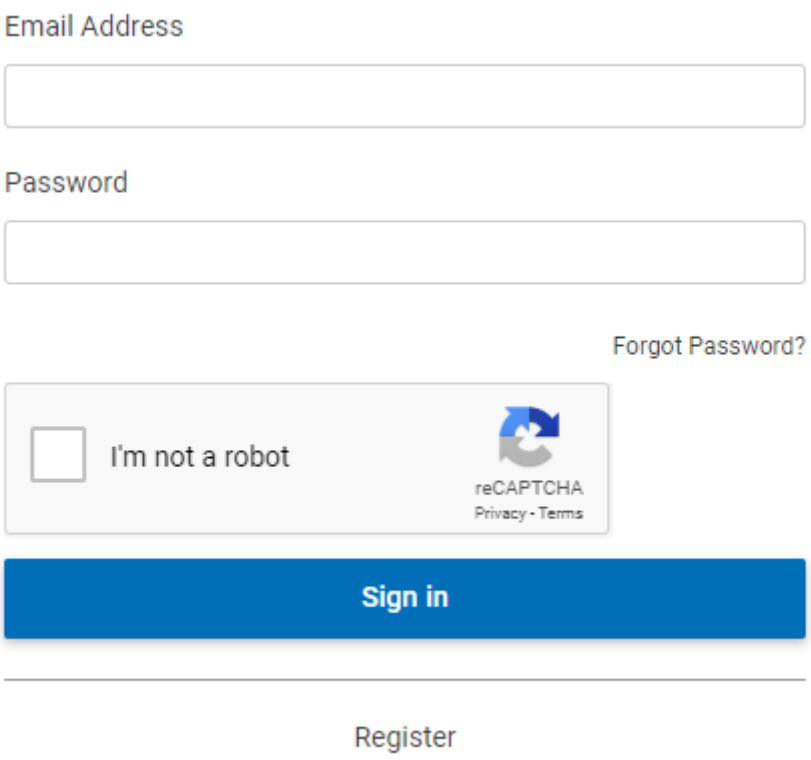

Figure 12: Graphical User Interface - Login Form

After sign up a confirmation email will go to the user email to confirm his or her email.

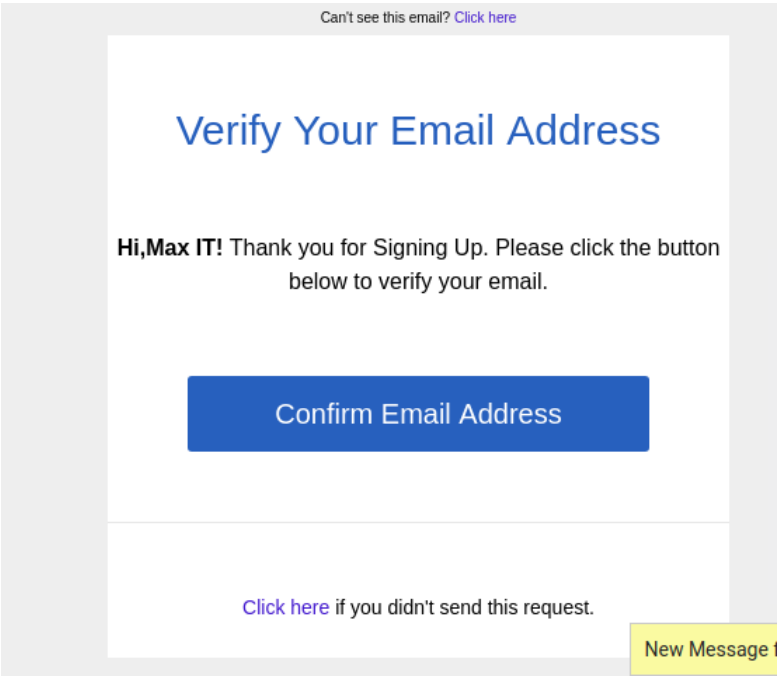

Figure 13: Graphical User Interface - Confirmation Email

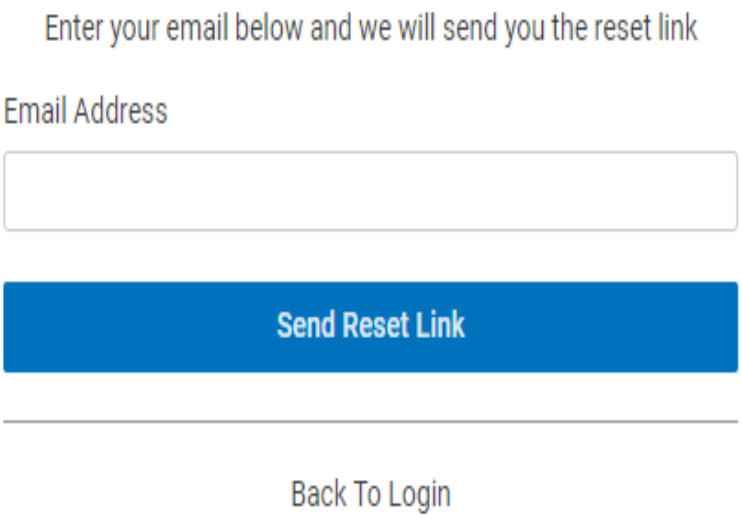

Figure 14: Graphical User Interface - Link for Resetting Password

Invitation cab be send to the friends through email which will help to grow the network.

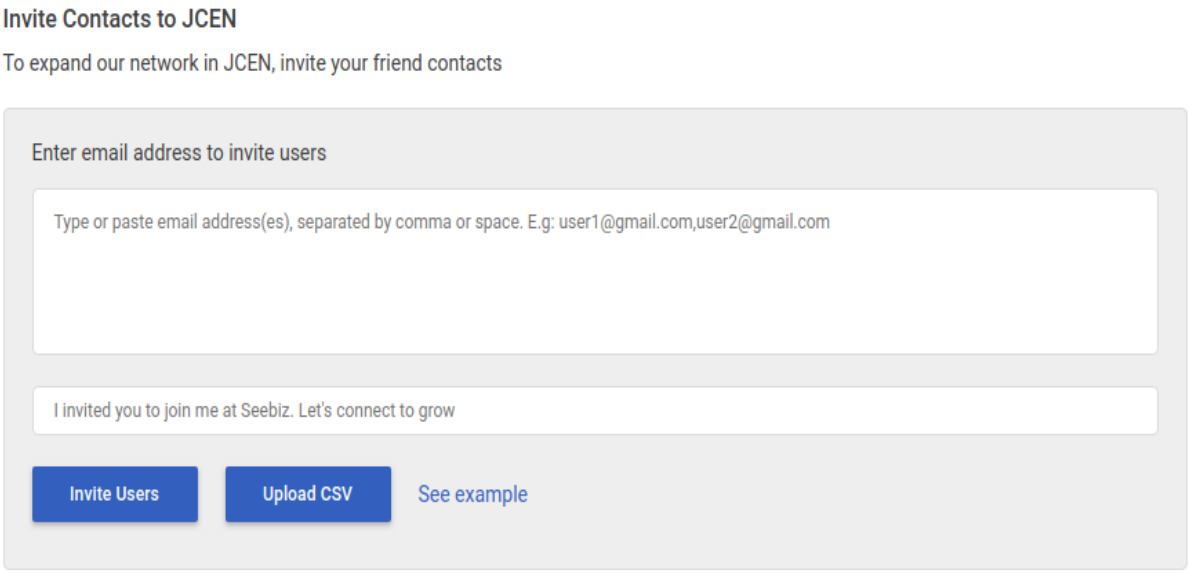

Figure 15: Graphical User Interface - Email Invitation

Following is the post creation form through which anyone can post job related post.

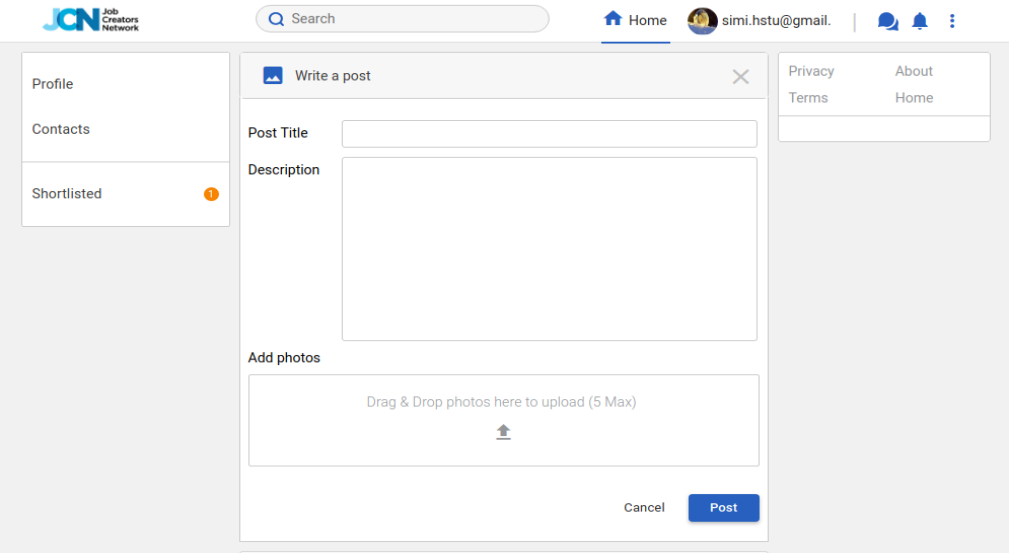

Figure 16: Graphical User Interface -Post Form

Following is the user post where user can see his own post and also the posts of his contacts

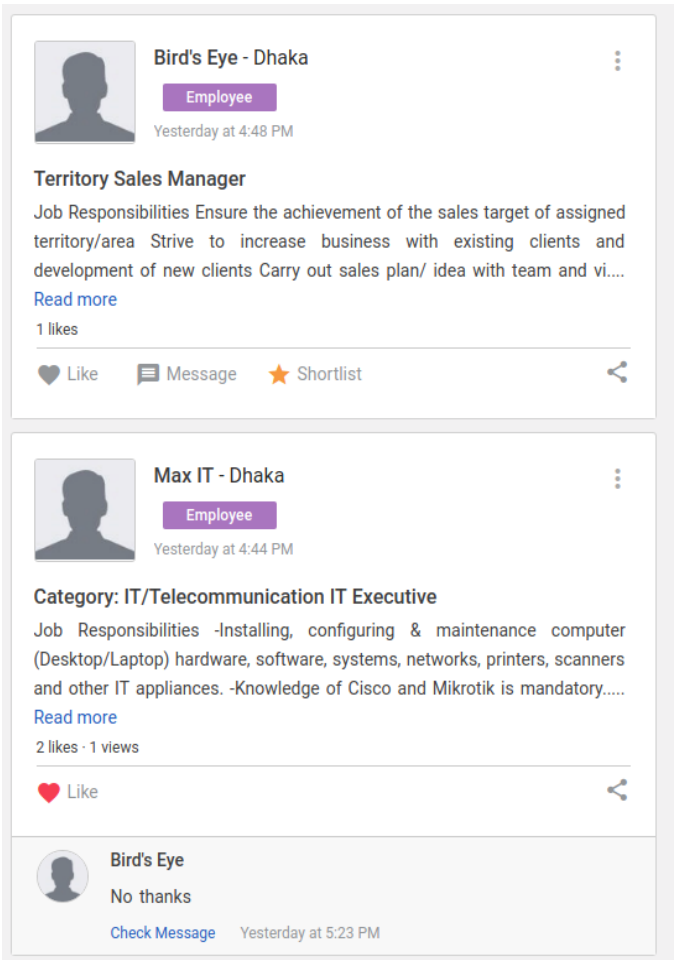

Figure 17: Graphical User Interface - Feed

There is an option for like, message and shortlist in each post

Figure 18: Graphical User Interface - Like, Message and Shortlist

Anyone can give like or can shortlist to his or his contacts post.

Figure 19: Graphical User Interface - Liked and Shortlisted

There is a pop up to see the shortlisted posts

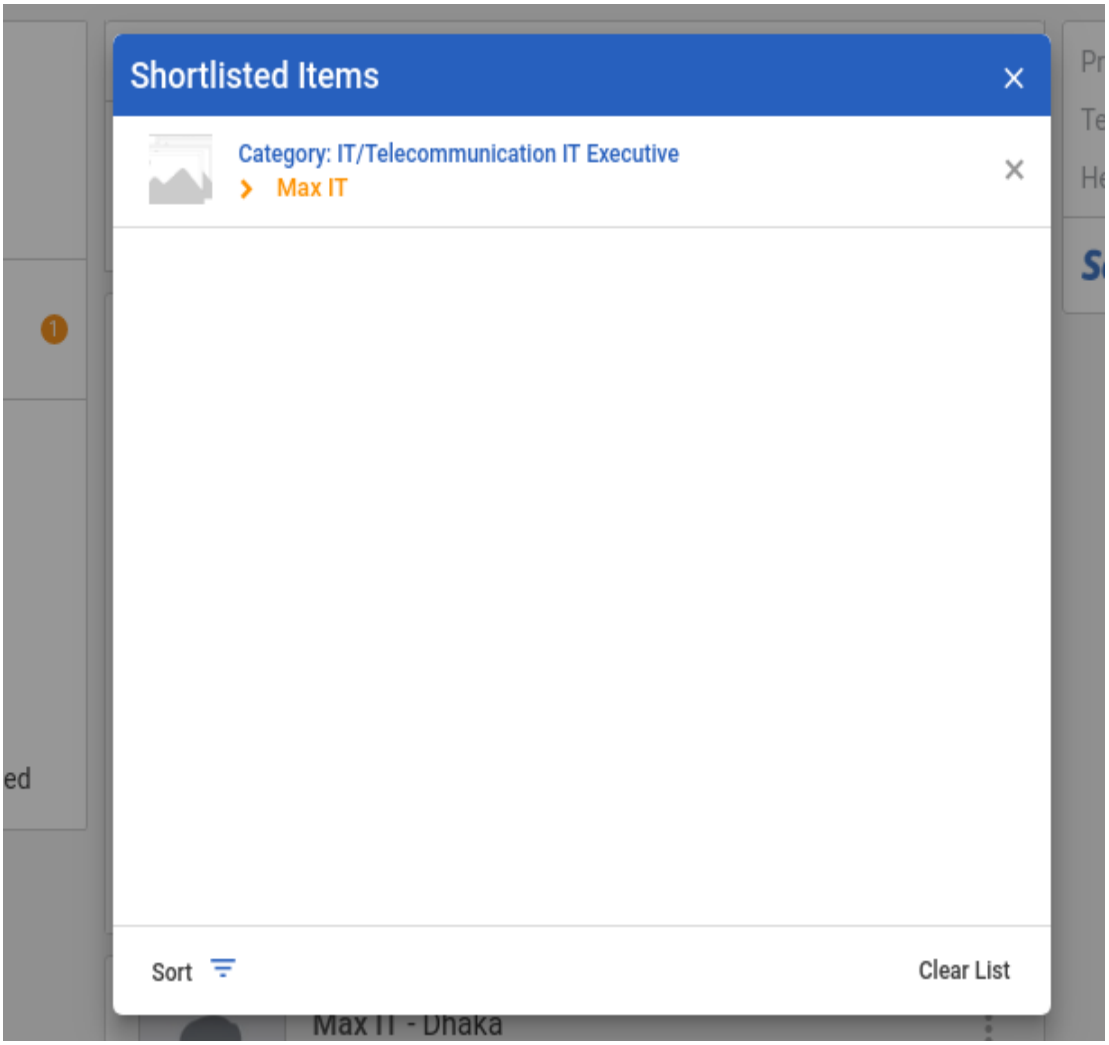

Figure 20: Graphical User Interface -Shortlisted Posts

There is a message option for message under each post. If anyone sends message to the post creator, a channel will open with the title of the post. Where they can negotiate about the post.

Figure 21: Graphical User Interface - Message from Post

In message module contacts can negotiate with each other for particular post, create new message, send attachment and can get notification for the new message.

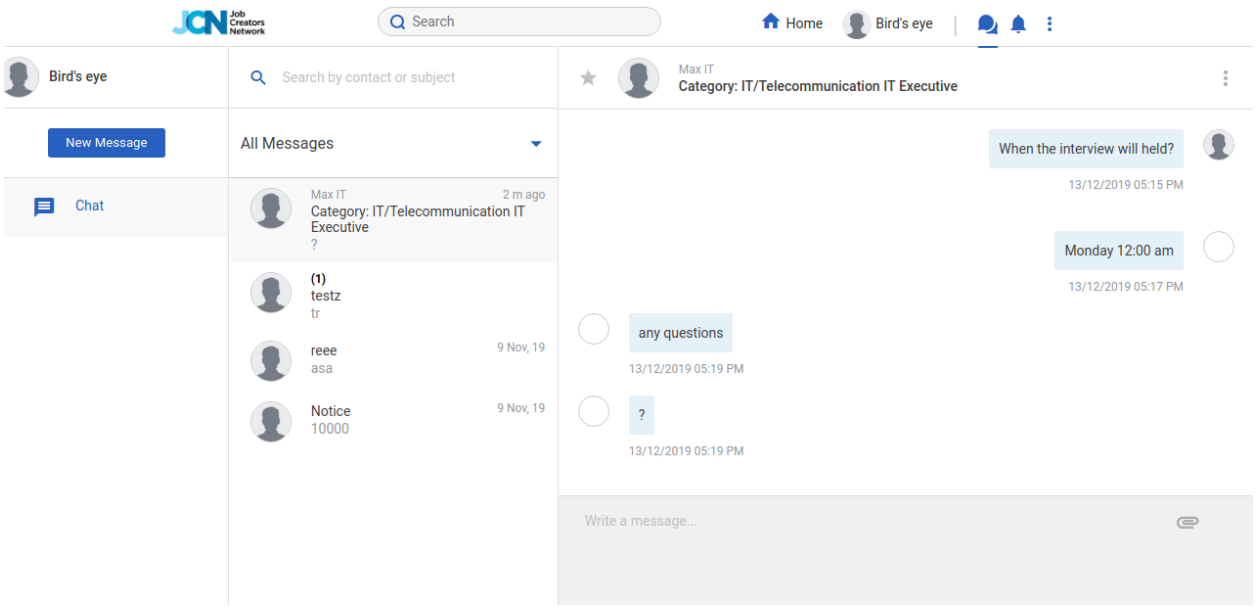

Figure 22: Graphical User Interface - Message Module

Figure 23: Graphical User Interface - Notification

User will notified if anyone sends him friend request, accepts it or give like to his post. He can accept or deny friend request from notification panel.

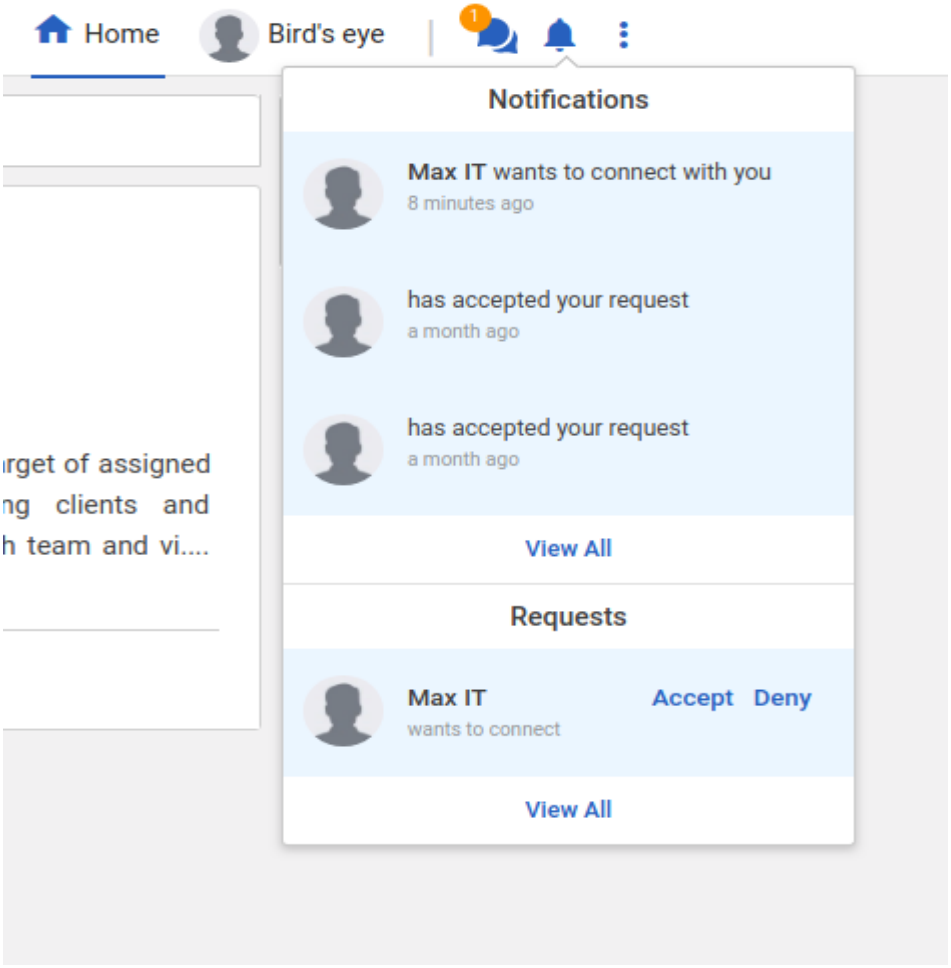

Figure 24: Graphical User Interface - Notification Panel

User can search for contacts and send friend request using the search module.

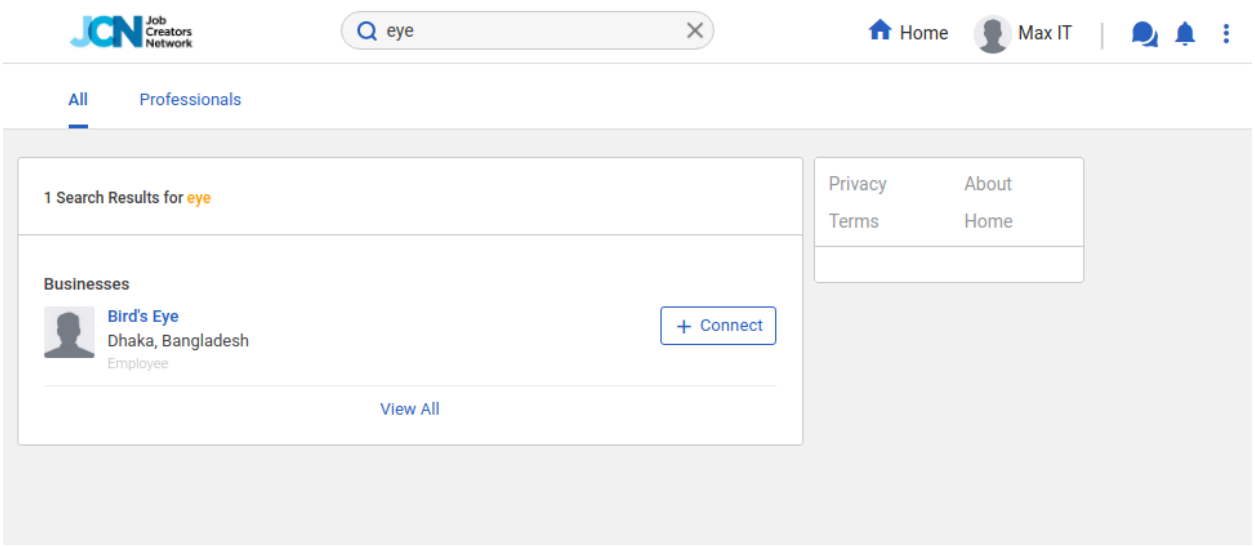

Figure 25: Graphical User Interface - Search Module

From contacts module user can see the list of contacts, add or remove contacts, create contacts group, send message, send invitation to join the network.

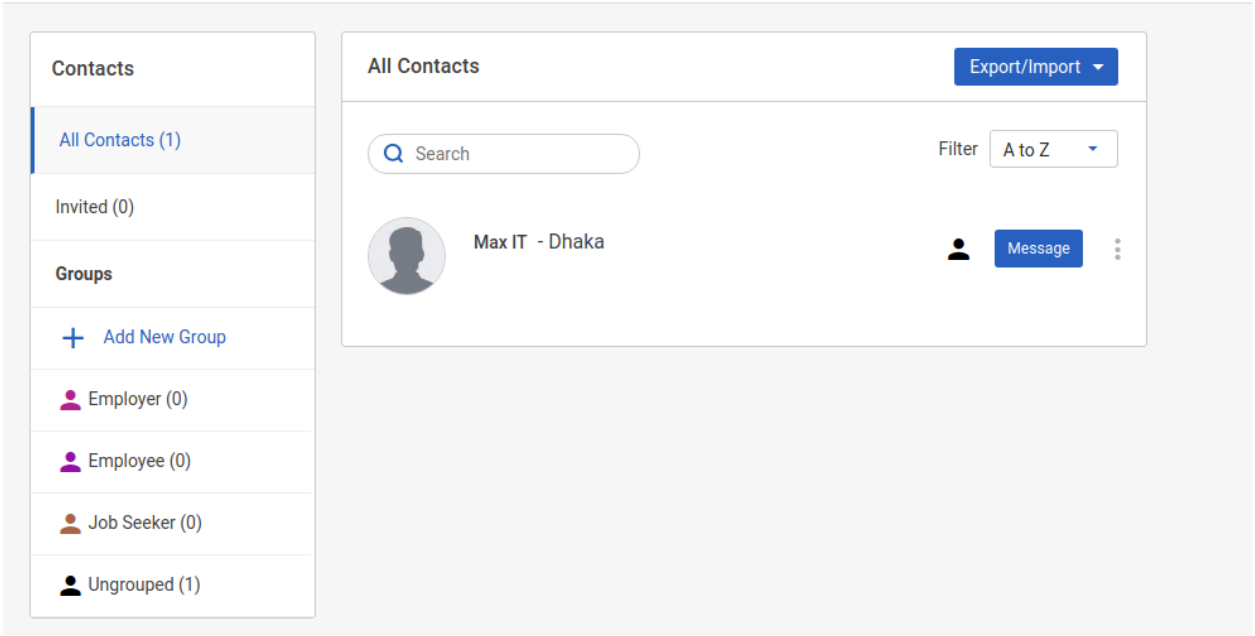

Figure 26: Graphical User Interface - Contacts Module

Following is the contact group form with which users can create different groups.

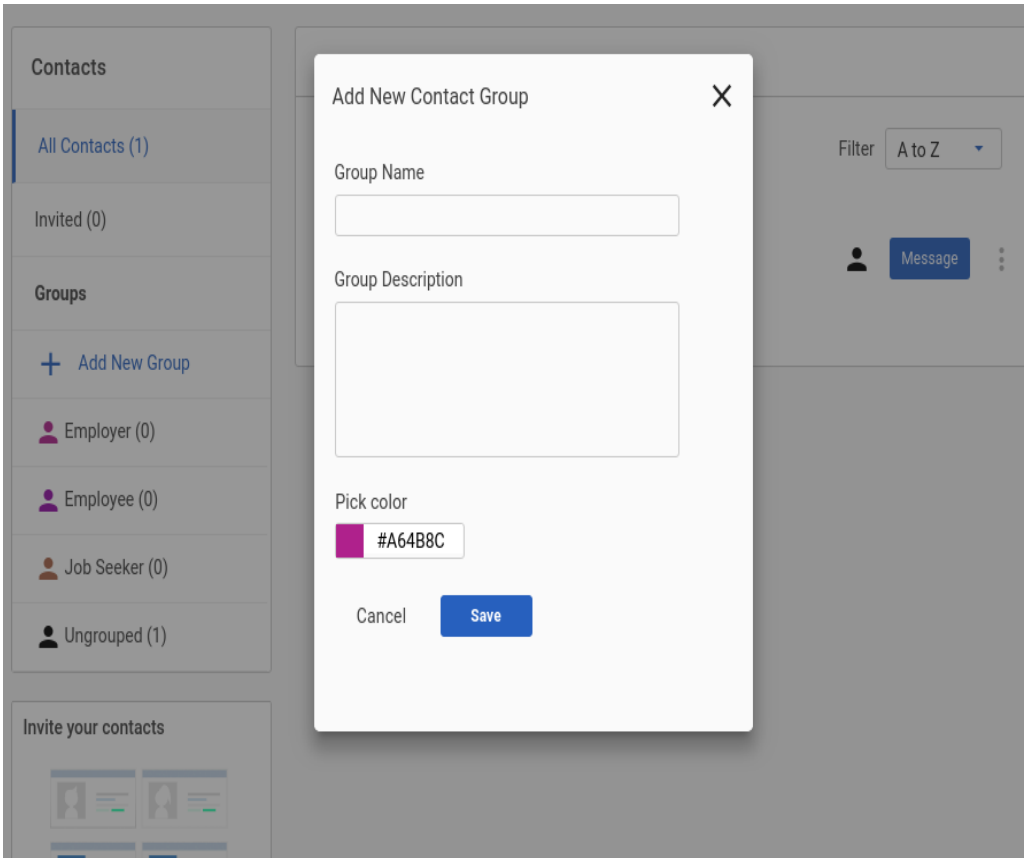

Figure 27: Graphical User Interface - Contact Group Form

Following is the option for removing or deleting contact.

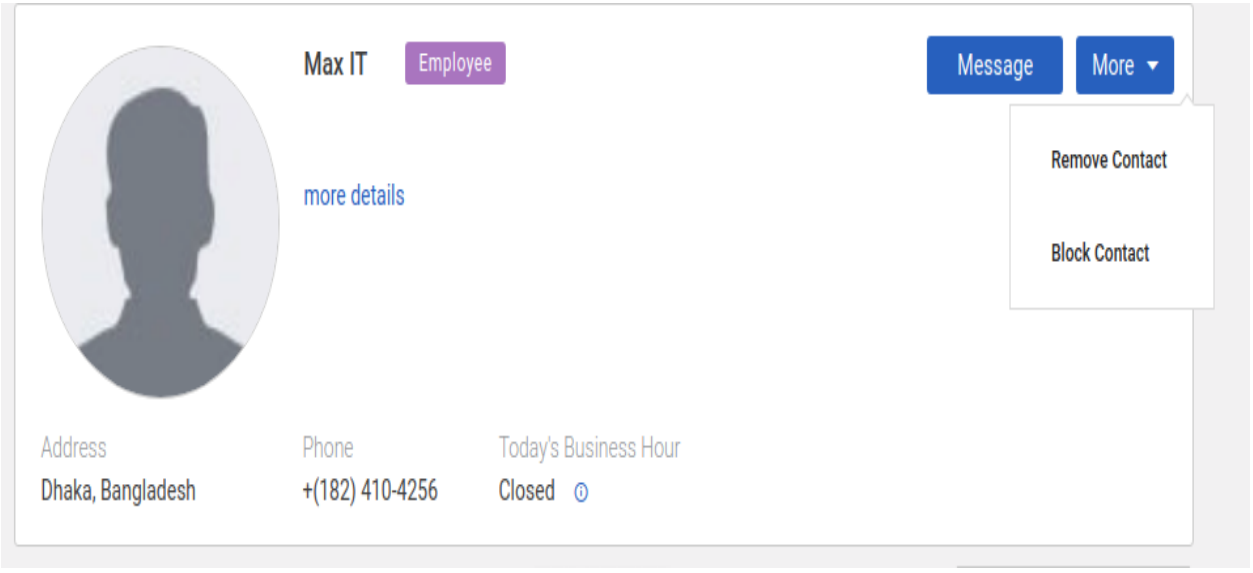

Figure 28: Graphical User Interface - Block or Remove Contact Options

Following is the settings module for updating user information.

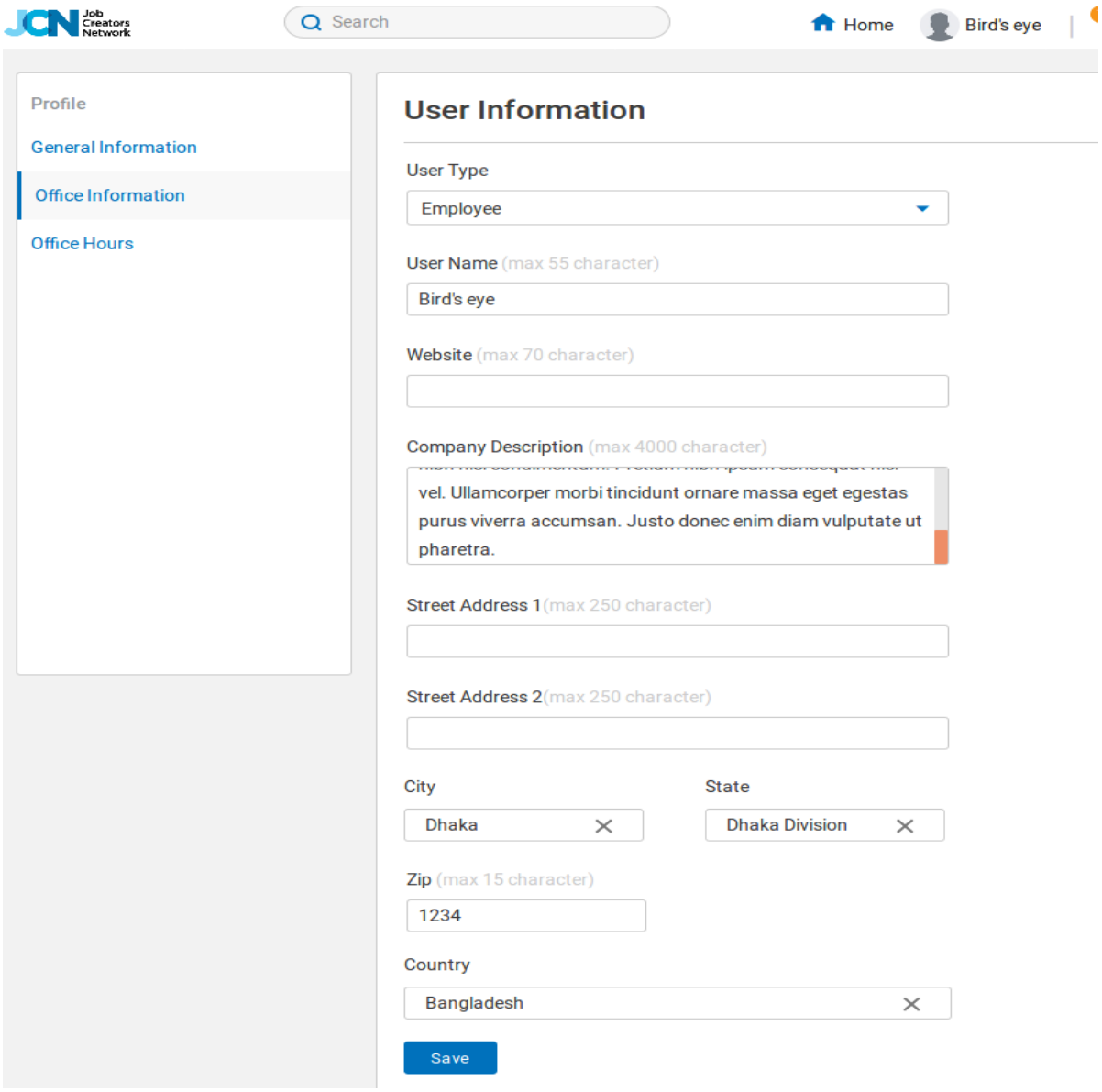

Figure 29: Graphical User Interface - Settings Module

Chapter 4

Results

The platform makes easier communication between the job seeker, recruiter and employers. It helps listing jobs easily, helps to grow network of contacts and connect with a wide range of professionals. It will also save time and money both for job seekers and employer's and. Hiring skilled and talented people will be easier than ever. It will also create new and better opportunities for the job seekers. This platform will also help professionals to engage and follow up any conversation.

Chapter 5

Conclusion

In conclusion we can say that social media has very strong effect in our present generation. Today is the era of networking and communication is the key to success. The more people and company join in this platform, the more engagement will occur which will lead to create more opportunities. As a result hiring right people for any position will be easier. However this platform not only made the job listing easier but also made communication with job seekers and employers very easy. This platform will also establish employer branding tool and employees review for them will also be recorded. So the professionals should ensure the best use of this platform to find great opportunities and great talents.

References

[1] Abel, S. (2011). The role of social networking sites in recruitment: Results of a quantitative study among German companies., (May), 1–65. Retrieved from http://essay.utwente.nl/61154/1/MSc_S_Abel.pdf

[2] Broughton, A., Foley, B., Ledermaier, S., & Cox, A. (2013). The use of social media in the recruitment process. Acas. https://doi.org/10.1080/00219266.2013.788541

[3] Kroeze, R. (2015). Recruitment via social media sites : A critical review and research agenda. 5th IBA Bachelor Thesis Conference.

[4] Ramasamy, V., & Raman, A. (2014). Recruitment in the Social Media Age : An Exploratory Study, (June), 1–13.

[5] Wood, S. P. (2016). The Role of Social Media in Disasters. Disaster Medicine, (April), 283–284. https://doi.org/http://dx.doi.org/10.1016/B978-0-323-28665-7.00044-3

Appendix

Few Line of Source Codes:

```
exports.create = function(req, res) {
  if (!req.body.postName) {
    res.json({
      success: false,
      message: 'Please enter post name.'
    });
  } else {
    var postSlug = createSlug(req.body.postName);

    var newPost = new Post({
      post_name: req.body.postName,
      slug: `${postSlug}-${shortid.generate()}`,
      category: req.body.category,
      post_class: req.body.postClass,
      merchant_id: req.user,
      description: req.body.description,
      invisible: req.body.invisible,
      published_date: req.body.publishDate
    });

    if (req.body.postClass === 'photo') {
      newPost.more_images = req.body.images;
      newPost.optimizedImages = req.body.optimizedImages;
    } else if (req.body.postClass === 'link') {
      newPost.show_image = req.body.showImage;
    }

    let today = new Date().toISOString();
    let myDate = new Date(today);
    let pt = new Date(req.body.publishDate);
    if (pt.getTime() > myDate.getTime()) {
      newPost.schedule_status = true;
    } else {
      newPost.schedule_status = false;
    }
    newPost.save(function(err, result) {
      if (err) {
        return res.json({
          success: false,
          message: 'That post already exists.'
        });
      } else {
        if (result) {
          Post.findOne({
```

```
          _id: result._id,
          merchant_id: result.merchant_id
        })
          .populate({
            path: 'merchant_id',
            select:
              'profile_pic optimizedProfilePic username unique_url business_name
first_name last_name business_info shortlisted likes',
            populate: {
              path:
                'business_info.businessType business_info.businessCategory',
              select: 'bussinessTypeName bussinessCategoryName'
            }
          })
          .exec(function(error, fullInfos) {
            if (error) {
              return res.json({
                success: false,
                message: 'That post already exists.'
              });
            } else {
              if (fullInfos) {
                if (fullInfos.schedule_status) {
                  res.send({
                    success: true,
                    data: fullInfos,
                    username: fullInfos.merchant_id.username,
                    message:
                      'Your scheduled post has been created successfully!'
                  });
                } else {
                  socket.socketFollowingUsers(
                    result.merchant_id,
                    'post-received',
                    fullInfos,
                    req,
                    res
                  );
                  Newsletter.sendPostUpdateContacts(fullInfos, res);
                  res.json({
                    success: true,
                    data: fullInfos,
                    username: fullInfos.merchant_id.username,
                    message: 'Your post has been created successfully!'
                  });
                }
              }
            }
          });
      }
```

```
      }
    });
  }
};

exports.getUserTimeline = function(req, res) {
  const io = req.app.get('socketio');
  let page = req.params.page;
  let qSort, qLimit; // defining default variable;
  let per_page = 20;
  let username = req.params.username;
  qSort = {
    created_at: -1
  };
  qSkip = (page - 1) * per_page; // Place the pagination count here
  qLimit = per_page; // Place the limit count here
  if (!req.user && !req.guest) {
    res.json({
      success: false,
      message: 'Request is missing from required data.'
    });
  } else {
    User.findOne({
      username: username
    }).exec(function(err, user) {
      if (!err) {
        if (user) {
          let pubUserId = user._id;
          let today = new Date().toISOString();
          var query = null;
          if (req.user) {
            query = {
              merchant_id: pubUserId,
              published_date: {
                $lte: today
              },
              schedule_status: false,
              invisible: {
                $nin: [req.user]
              },
              isTrashed: false
            };
          } else if (req.guest) {
            query = {
              merchant_id: pubUserId,
              published_date: {
                $lte: today
              },
              schedule_status: false,
              invisible: {
```

```
            $size: 0
           },
           isTrashed: false
          };
         }
         Post.find(query)
          .sort(qSort)
          .skip(qSkip)
          .limit(qLimit)
          .populate(
           'merchant_id',
           'profile_pic optimizedProfilePic username unique_url business_name first_name
last_name business_info shortlisted likes edit_history'
          )
          .exec(function(error, result) {
           if (!error) {
            // Checking if own post or other
            let data = result.map((d, indx) => {
             if (d.merchant_id._id == req.user) {
              let obj = {
               _id: d._id,
               category: d.category,
               created_at: d.created_at,
               description: d.description,
               shortlisted: d.shortlisted,
               likes: d.likes,
               invisible: d.invisible,
               isTrashed: d.isTrashed,
               merchant_id: d.merchant_id,
               catalog_id: d.catalog_id,
               product_id: d.product_id,
               more_images: d.more_images,
               optimizedImages: d.optimizedImages,
               post_name: d.post_name,
               published_date: d.published_date,
               share_with: d.share_with,
               slug: d.slug,
               status: d.status,
               post_class: d.post_class,
               show_image: d.show_image,
               ownPost: true,
               viewBy: d.viewBy,
               totalViews: d.totalViews,
               price: d.price,
               edit_history: d.edit_history
              };
              return obj;
             } else {
              let obj = {
               _id: d._id,
```

```
                category: d.category,
                created_at: d.created_at,
                description: d.description,
                shortlisted: d.shortlisted,
                likes: d.likes,
                invisible: d.invisible,
                isTrashed: d.isTrashed,
                merchant_id: d.merchant_id,
                catalog_id: d.catalog_id,
                product_id: d.product_id,
                more_images: d.more_images,
                optimizedImages: d.optimizedImages,
                post_name: d.post_name,
                published_date: d.published_date,
                share_with: d.share_with,
                slug: d.slug,
                status: d.status,
                post_class: d.post_class,
                show_image: d.show_image,
                ownPost: false,
                viewBy: d.viewBy,
                totalViews: d.totalViews,
                price: d.price,
                edit_history: d.edit_history
              };
              return obj;
            }
           });
           res.send({
            data: data
           });
          } else {
           res.send(error).status();
          }
         });
      } else {
       res.status(400).send('user not found');
      }
     }
   });
  }
};

exports.getPostInfo = function(req, res) {
  var postId = req.params.id;
  Post.findOne({
    _id: postId,
    merchant_id: req.user,
    post_class: 'product'
  })
```

```
    .select(
      '_id post_name merchant_id description more_images optimizedImages catalog_id
product_id price'
    )
    .exec(function(err, result) {
      if (err) {
        res.status(400).send(err);
      } else {
        res.send(result);
      }
    });
};

exports.getTotalPostViews = function(req, res) {
  var postId = req.params.id;
  Post.findOne({
    _id: postId
  })
    .select('viewBy totalViews')
    .populate({
      path: 'viewBy.userId',
      options: {
        select: '_id username profile_pic business_info'
      }
    })
    .exec(function(err, result) {
      if (err) {
        res.status(400).send(err);
      } else {
        res.send(result);
      }
    });

  var postId = req.query.key;
  Post.findOne({
    _id: postId,
    merchant_id: req.user
  }).exec(function(err, result) {
    if (err) {
      res.status(400).send(err);
    } else {
      if (result) {
        result.isTrashed = true;
        result.save(function(error, done) {
          if (error) {
            res.status(400).send(error);
          } else {
            res.send({
              success: true,
              message: 'successfully trashed!'
```

```
          });
        }
      });
    } else {
      res.send({
        success: false,
        message: 'not_own_post'
      });
    }
  }
 });
};
exports.countView = function(req, res) {
 let postId = req.body.postId;
 Post.findOne({ _id: postId })
  .select('_id viewBy totalViews')
  .exec(function(err, result) {
    if (err) {
      res.status(400).send(err);
    }
    if (result.viewBy.length > 0) {
      let check = result.viewBy.filter(e => {
        return e.userId.toString() === req.user.toString();
      });
      if (check.length > 0) {
        result.totalViews = result.totalViews + 1;
        result.save((err, data) => {
          if (err) {
            res.status(400).send(err);
          } else {
            res.send(data);
          }
        });
      } else {
        result.totalViews = result.totalViews + 1;
        result.viewBy.push({ userId: req.user });
        result.save((err, data) => {
          if (err) {
            res.status(400).send(err);
          } else {
            res.send(data);
          }
        });
      }
    } else {
      result.totalViews = result.totalViews + 1;
      result.viewBy.push({ userId: req.user });
      result.save((err, data) => {
        if (err) {
          res.status(400).send(err);
```

```
        } else {
          res.send(data);
        }
      });
    }
  });
};

elastic.client.search({
  index: 'products',
  ignore: [404],
  size: size,
  from: from,
  //q: `${req.query.q}`
  body: {
    query: {
      bool: {
        must: [
          {
            multi_match: {
              query: req.query.q,
              type: 'phrase_prefix',
              fields: ['title', 'description']
            }
          }
        ],
        filter: filter
      }
    },
    sort: [{'price': {'order': 'desc'}}]
  }
}).then((result) => {
  res.send({
    results: result.hits
  });
}).catch((err) => {
  res.send({
    success: false,
    message: 'Something went wrong!'
  });
});
};

exports.createShortlist = function(req, res) {
  var userId = req.user;
  var postId = req.params.id;
  Post.findOne({
    _id: postId,
  })
    .populate('merchant_id', 'username _id')
```

```
.exec(function(err, post) {
  if (!err) {
    if(post){
      if (
        post.shortlisted.filter(function(e) {
          return e.userId == userId;
        }).length > 0
      ) {
        Post.findOneAndUpdate(
          { _id: postId },
          { $pull: { shortlisted: { userId: userId } } },
          { multi: false },
          function(err, success) {
            if (err) {
              console.log(err);
            } else {
              Post.findOne({_id: postId}).exec(function (err, data) {
                if(!err){
                  if(data){
                    res.send({
                      success: true,
                      message: 'removed from shortlist successfully!',
                      action: 'remove',
                      data: data
                    });
                  }
                }
                else {
                  res.status(400).send({
                    message: err.message
                  })
                }
              });
            }
          }
        );
        // Making unshortlist is ended
      } else {
        var shortlist = {
          userId: req.user,
        };
        post.shortlisted.push(shortlist);
        post.save(function(err, data, num) {
          if(data){
            res.send({
              success: true,
              message: 'shortlisted successfully!',
              action: 'create',
              data: data
            });
```

```
            }
           });
         }
        }
        else{
          res.status(400).send('Post not found');
        }
      }
    });
};

exports.getTotalShortlistsCount = function(req, res) {
  Post.find({
    shortlisted: {
      $elemMatch: {
        userId: req.user,
      },
    },
    published_date: {
      $lte: new Date().toISOString(),
    },
    isTrashed: false,
  })
    .select('_id')
    .exec(function(err, result) {
      if (err) {
        res.sendStatus(400).send(err);
      } else {
        res.send(result);
      }
    });
};
exports.getTotalShortlists = function(req, res) {
  var skipData = req.body.skip || 0;
  var limit = 8;
  var sortToDate = req.query.Sort;

  var dateRange;

  //  Extending date object with prototype to add more days
  Date.prototype.addDays = function(days) {
    this.setDate(this.getDate() + parseInt(days));
    return this;
  };
  var currentDate = new Date();

  // var today = new Date(currentDate.addDays(0)).toISOString();

  // var maxDays = new Date(currentDate.addDays(-10)).toISOString();
```

```
if (sortToDate == 'today') {
  dateRange = {
    $lte: new Date(currentDate.addDays(0)).toISOString(),
    $gte: new Date(currentDate.addDays(-1)).toISOString(),
  };
} else if (sortToDate == 'yesterday') {
  dateRange = {
    $lte: new Date(currentDate.addDays(0)).toISOString(),
    $gte: new Date(currentDate.addDays(-2)).toISOString(),
  };
} else if (sortToDate == '7days') {
  dateRange = {
    $lte: new Date(currentDate.addDays(0)).toISOString(),
    $gte: new Date(currentDate.addDays(-7)).toISOString(),
  };
} else if (sortToDate == '30days') {
  dateRange = {
    $lte: new Date(currentDate.addDays(0)).toISOString(),
    $gte: new Date(currentDate.addDays(-30)).toISOString(),
  };
} else {
  dateRange = {
    $lte: new Date(currentDate.addDays(0)).toISOString(),
  };
}
Post.find({
  shortlisted: {
    $elemMatch: {
      userId: req.user,
      shortlist_at: dateRange,
    },
  },
  published_date: {
    $lte: new Date().toISOString(),
  },
  isTrashed: false,
})
  .sort({ 'shortlisted.shortlist_at': -1 })
  .skip(skipData)
  .limit(limit)
  .select('_id post_name slug merchant_id description more_images price')
  .populate({
    path: 'merchant_id',
    select:
      'username unique_url business_name first_name last_name business_info',
  })
  .exec(function(err, result) {
    if (err) {
      res.sendStatus(400).send(err);
    } else {
```

```
        res.send(result);
      }
    });
};
exports.removeShortlist = function(req, res) {
  const userId = req.user;
  const postId = req.params.postId;
  Post.findOneAndUpdate(
    { _id: postId },
    { $pull: { shortlisted: { userId: userId } } },
    { multi: true },
    function(err, success) {
      if (err) {
        console.log(err);
      } else {
        res.json(success);
      }
    }
  );
};

/**
 * Clear User Shortlist List
 */
exports.clearShortlistList = function(req, res) {
  Post.update(
    {},
    { $pull: { shortlisted: { userId: req.user } } },
    { multi: true },
    function(err, success) {
      if (err) {
        console.log(err);
      } else {
        res.json(success);
      }
    }
  );
};


exports.send = function (from, to, message, link, callback) {
  /*
  @@ Sending notification
  */
  var Notify = new Notification({
    sender: from,
    receiver: to,
    message: message,
    link: link
  });
```

```
  Notify.save(function (err, result, num) {
    callback(err, result, num);
  });
  //Ends notification
};
/*
@@ Getting all notification by pagination
*/
exports.getAll = function (req, res) {
  // pagining information
  var skip = '';
  if (req.query.skip > 0) {
    skip = parseInt(req.query.skip);
  } else {
    skip = 0;
  }
  Notification.find({
    "receiver": {
      "$in": [req.user]
    }
  }).sort({
    'created_at': -1
  }).limit(5).skip(skip).populate('sender', 'profile_pic first_name last_name username
business_info').exec(function (err, result) {
    if (!err) res.send(result);
  });
};

/**
 * Getting new notifications
 */
exports.new = function (req, res) {
  Notification.find({
    "receiver": {
      "$in": [req.user]
    },
    "read_by.readerId": {
      "$nin": [req.user]
    }
  }).sort({ 'created_at': -1 })
    .populate('sender', 'profile_pic first_name last_name username business_info')
    .exec(function (err, result) {
      res.send(result);
    });
};

/**
 * Reading the notification
 */
```

```
exports.read = function (req, res) {
  var notifyId = req.params.id;
  var userId = req.user;
  Notification.findOne({
    '_id': notifyId
  }, function (err, noti) {
    if (!err) {
      if (noti.read_by.filter(function (e) {
        return e.readerId == userId;
      }).length > 0) {
        res.send({
          success: true,
          message: 'Already viewed.'
        });
      } else {
        noti.read_by.push({
          readerId: userId
        });
        noti.save(function (err, data, num) {
          res.send({
            success: true,
            message: 'Viewed successfully!!'
          });
        });
      }
    }
  });
};

exports.readAll = function (req, res) {
  if (req.user) {
    Notification.find({
      'receiver': req.user
    }, function (err, noti) {
      if (!err) {
        for (var i = 0; i < noti.length; i++) {
          if (noti[i].read_by.filter(function (e) {
            return e.readerId == req.user;
          }).length > 0) {
            // console.log('Notification already viewed');
          } else {
            noti[i].read_by.push({
              readerId: req.user
            });
            noti[i].save(function (err, data, num) {
              console.log('Viewed');
            });
          }
        }
        res.send({
```

```
          success: true,
          message: 'Viewed successfully!!'
        });
      }
    });
  } else {
    req.status(401).send({
      'message': 'unauthorised'
    });
  }
};

/**
 * Getting un viewed notifications
 */
exports.getUnviewed = function (req, res) {
  Notification.find({
    "receiver": {
      "$in": [req.user]
    },
    "viewed_by.viewerId": {
      "$nin": [req.user]
    }
  }).sort({ 'created_at': -1 })
    .populate('sender', 'profile_pic first_name last_name username business_info')
    .exec(function (err, result) {
      res.send(result);
    });
};

exports.viewAll = function (req, res) {
  if (req.user) {
    Notification.find({
      'receiver': req.user
    }, function (err, noti) {
      if (!err) {
        for (var i = 0; i < noti.length; i++) {
          if (noti[i].viewed_by.filter(function (e) {
            return e.viewerId == req.user;
          }).length > 0) {
            // console.log('Notification already viewed');
          } else {
            noti[i].viewed_by.push({
              viewerId: req.user
            });
            noti[i].save(function (err, data, num) {
              console.log('Viewed');
            });
          }
        }
```

```
        res.send({
          success: true,
          message: 'Viewed successfully!!'
        });
      }
    });
  } else {
    req.status(401).send({
      'message': 'unauthorised'
    });
  }
};
exports.likePost = function(req, res) {
  var userId = req.user;
  var postId = req.params.id;
  Post.findOne({
    _id: postId,
  })
    .populate('merchant_id', 'username _id')
    .exec(function(err, post) {
      if (!err) {
        if(post){
          if (
            post.likes.filter(function(e) {
              return e.userId == userId;
            }).length > 0
          ) {
            Post.findOneAndUpdate(
              { _id: postId },
              { $pull: { likes: { userId: userId } } },
              { multi: false },
              function(err, success) {
                if (err) {
                  console.log(err);
                } else {
                  Post.findOne({_id: postId}).then(function (data) {
                    socket.updateLikeCount(req, data._id);
                    res.send({
                      success: true,
                      message: 'removed like successfully!',
                      action: 'remove',
                      data: data
                    });
                  })
                  .catch(function (error) {
                    res.status(400).send({
                      message: error.message
                    })
                  })
                }
```

```
    }
  );
  // Making unshortlist is ended
 } else {
  var like = {
   userId: req.user,
  };
  post.likes.push(like);
  post.save(function(err, data, num) {
   if(data){
    if(userId !== post.merchant_id.id){
     let sms = 'liked ' + post.post_name;
     let link =
      config[env].clientHost +
      '/user/' +
      post.merchant_id.username +
      '/post/' +
      post.slug +
      '?i=' +
      post._id;

     Notify.send(userId, data.merchant_id, sms, link, function(
      err,
      result,
      num
     ) {
      if (!err) {
       socket.increaseNotification(
        data.merchant_id._id,
        'notification',
        req,
        res
       );
       socket.updateLikeCount(req, data._id);
       res.send({
        success: true,
        message: 'notification sent successfully!',
        action: 'create',
        data: data
       });
      }
     });
    }else {
     socket.updateLikeCount(req, data._id);
     res.send({
      success: true,
      message: 'liked successfully!',
      action: 'create',
      data: data
     });
```

```
                }
              }
            });
          }
        }
        else{
          res.status(400).send('Post not found');
        }
      }
    });
};
```