International Journal of Interdisciplinary Innovative Research & Development (IJIIRD)
ISSN: 2456-236X
Vol. 03 Special Issue | 2019

Scanly Using Node.js

Dr. Vaishali D. Khairnar[1], Mr.Amit Vishwakarma[2], Mr.Aditya Hazarika[3]
[1, 2, 3] *Department of Information Technology, Terna Engineering College, Nerul, Navi Mumbai*

ABSTRACT

This paper contains the information about the project which use to mail automation. In this project the user only need to use the web application and mobile application to scan the visiting card then this application fetch all the data mention in the visiting card. At that time application send mail to card holder email address in the format which is selected by the user. Here users need to select the template for format of the mail and then scan the visiting card. Then the application do the whole work for user in the back end like sending mail, storing all the data of VC holder, etc.

Keywords—Mail Automation, Email address, Visiting card, web application, mobile application, User.

1. INTRODUCTION

This application helps users for creating and maintaining new contacts in professional life. In this modern word the people need to make new contacts with other organization, companies, etc. for growth of their business. This application helps in reducing time and effort of the users and it is quicker than the traditional way.

This application helps the business people in all possible way to create new contacts for their growth. Here application provides the mechanism for scan the Visiting card then fetch all the information mention in it, using that information app send the mail to the respective person without any effort of users. Thus the application helps to create instant contact.

In this paper we mention the design and implementation of Visit Automation using OCR library for scanning the Visiting Card, mailing library for auto-mailing purpose, Gmail Authentication, Local Authentication for authentication and authorization.

1.1 Main Feature of This Paper:

1. Platform independent
2. Instant mailing
3. Faster than traditional way
4. Creating and Maintaining new contacts
5. Create own mail format

We are further going to discuss existing system of this paper

2. EXISTING SYSTEM

First there are many mail automation systems available in market but there is no such application which uses the OCR library or scanning system for fetching the data and figure the email then mail to that email address. Some available systems are not able to provide error-free data for further processes.

The previous systemsare just collect the data from the visiting cards using standard OCR features and some systems provide mechanism for mail automation but there user have to fill the required fields but this system provide the OCR with mail automation facility for instant mailing in nutshell without any extra efforts.

There are many systems available for mail automation but they can't give you the guarantee of the mailing. And some applications have many confusing way to achieve the goal not like this project which is easy to use. This system gives the flexibility to use the application from web browser as well as mobile devices.

Some systems uses the hardware devices for scanning process but this application only used software for scanning. So the users don't need to carry such devices. Users only need to login to their accounts either from web browser or mobile application and perform all the required tasks.

The database, libraries, framework and dependencies which are used in existing system:

1. *MongoDB*
2. *Passport-google-auth*
3. *Passport-local-mongoose*
4. *Nodemailer*
5. *express.js*
6. *mongoose*
7. *passport-jwt*
8. *EJS*
9. *OCR*

1. MongoDB: MongoDB is NoSQL type database. Which gives more flexibility, fault-tolerance, efficiency, agile than RDBMS.
2. Passport-google-auth: It is a library of a node.js which used to provide the functionality of Gmail authentication in this system. Using this library the system gives the privilege to use all the services without creating another local account.
   Here they use passport-strategy for distinguish all the users based on their google profile unique id.
3. Passport-local-mongoose: This library is used to create the local account for the user and create their session and uses all the services. User can use all the services if they don't have the google account, twitter account, facebook account, etc. Here they use local-strategy for distinguish all the users based on their Unique key.
4. Nodemailer: This is the library which used to send mail to any recipient using the user mailing credentials like SMTP credentials or Gmail credentials. In SMTP credentials Nodemailer uses username, password, SMTP server name, port-name and SSH key. In other hand Gmail credentials which used in Nodemailer are refreshToken, accessToken.
5. Express.js: Express.js is framework of node.js. This framework used for creating RESTful APIs. In this framework all the methods we can use like GET, POST, PUT, DELETE methods. This framework use for creating CRUD functions.
6. Mongoose: This is dependency in node.js for creating database models in MongoDB and connects all those models with APIs.
7. Passport-jwt: Passport-jwt is a library in node.js which distinguish all the user based on theirjwt(json web token). Passport-jwt uses users unique id then generate the jwt for every user.
8. EJS: EJS(Embedded JavaScript) is a function of node.js which used to create the view for system. It consist the html, css, jquery, javascript. This type of files have the extension of ".ejs". They are bind with APIs.
9. OCR: This library used in the mobile device for fetching all the data in text form. For example extract all the information on the visiting card like company name, name, designation, email-id, contact no, website and other information by scanning it.

All the above frameworks, dependencies, libraries used in this system.

3. SYSTEM ARCHITECTURE

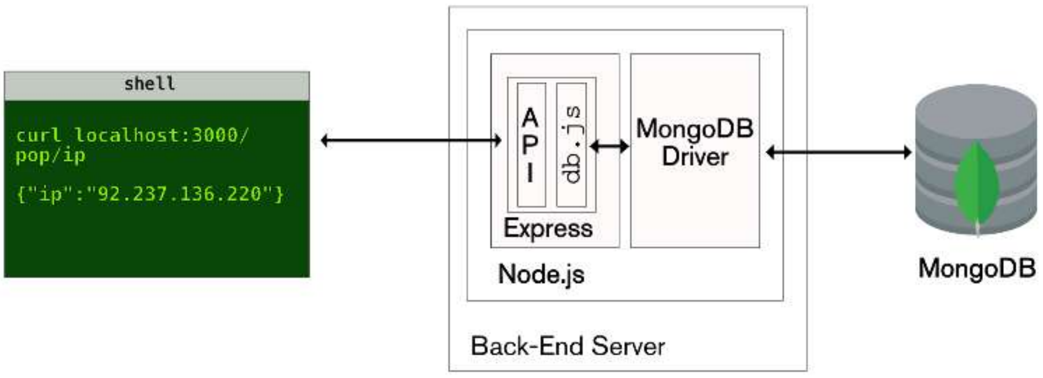
```
curl localhost:3000/
pop/ip
{"ip":"92.237.136.220"}
```

Fig 3.1 . System Architecture

The implemented system consists of NoSQL Database storage MongoDB, a server-side scripting language Node.js, Web framework for Node.js which is "ExpressJs". We are using Representational State Transfer (REST) APIs which provides a set of operations that can be invoked by a remote client which could be another service also over a network, using the HTTP protocol. Shell command is used to run Node.js on specific ports.

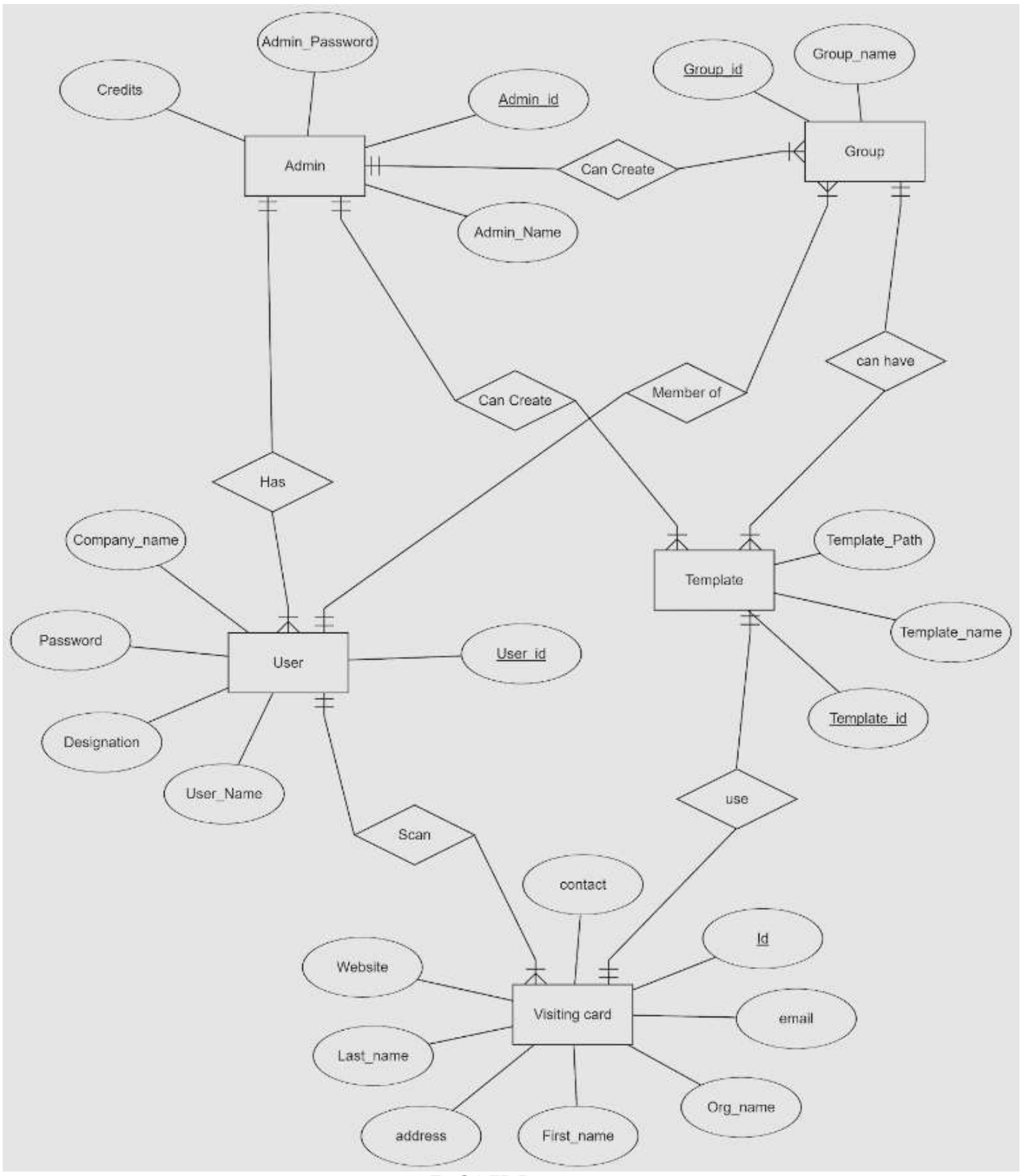

Fig 3.2 ER Diagram

4. PROPOSED SYSTEM

The system will allow user to scan visiting card through OCR which will capture organization's name, email, address, website etc. After scanning of information user can verify scanned data, after verification an email will be sent from backend server to the scanned email-ID. User can create and set his/her own email template. Only Admin has the right to invite users, delete users, add groups and delete Groups

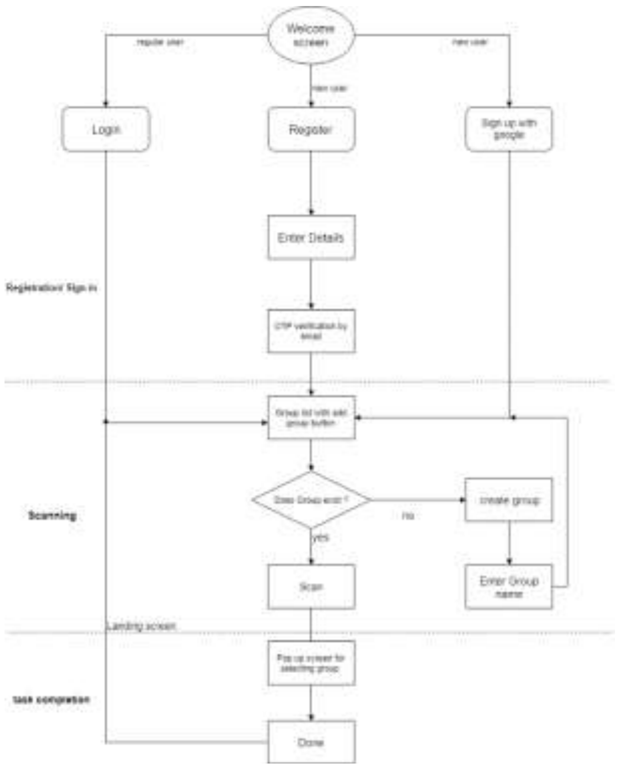

Fig 4.1 ADMIN'S FLOW

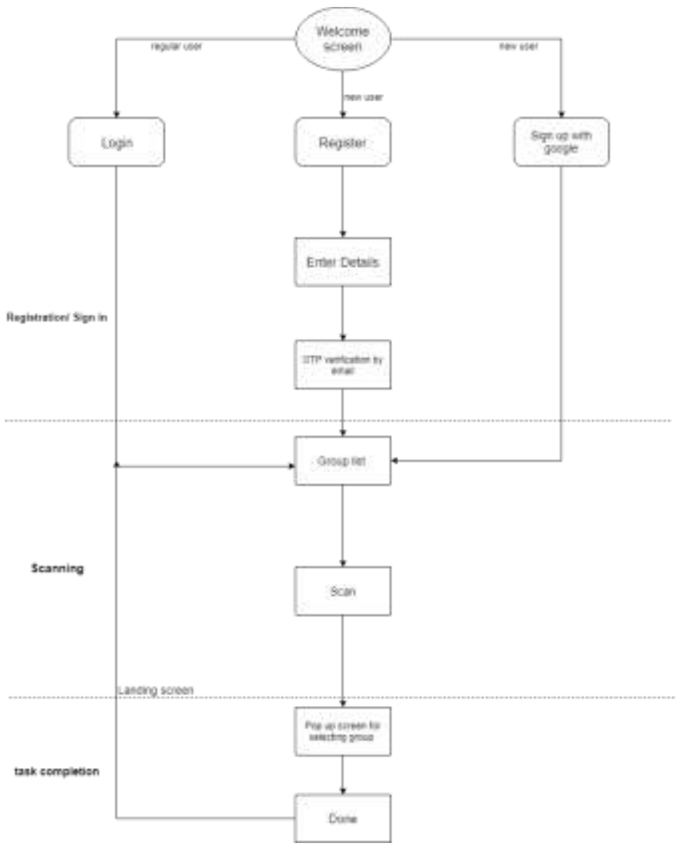

Fig 4.2 USERS FLOW

5. CONCLUSION

This system is geared to enhance the data entry and mailing process for the company where a user can efficiently send multiple emails in short time span just by scanning visiting cards with ready email template.

6. REFERENCES

[1] Web Development with Node Js and Express- Ethan Brown

[2] http://nodeguide.com/Node.js, Express.js, Mongoose.js and Passport.js Authentication

[3] http://net.tutsplus.com/tutorials/javascript-ajax/node-js-for-beginners/Eslam Maged

[4] Mukesh Chapagain, "Node.js, MongoDB & Express: Simple Add, Edit, Delete, View (CRUD)"(2017).

[5] Manoj Singh Negi, "Sending an email using NodeMailer & Gmail"(2018).

[6] https://hackr.io/tutorials/learn-node-jshttps://www.npmjs.com/package/mongoose

[12]http://www.passportjs.org/docs/downloads/html/.
[13] http://www.passportjs.org/packages/passport-jwt/
[14]https://docs.mongodb.com/manual/crud/
[15]https://expressjs.com/en/guide/routing.html.
[16]https://ejs.co/#docs
[17]https://nodemailer.com/about/
[18]https://ieeexplore.ieee.org/document/6836618
[19]https://ieeexplore.ieee.org/document/7960064
[20]https://ieeexplore.ieee.org/document/5617064