**TYPOO**

# A PROJECT REPORT

*Submitted in partial fulfillment of the requirements for the award of the degree of*

## BACHELOR OF TECHNOLOGY

**in**

## CSE with Specialization in

**Cloud Computing & Virtualization**

**Submitted by:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Course** | **Branch** | **SAP ID** |
| Prabal Bansal | B.Tech. CSE | CCVT | 500068861 |
| Raghav Jindal | B.Tech. CSE | CCVT | 500067476 |
| Raman Kumar | B.Tech. CSE | CCVT | 500068477 |
| Sachin Agrawal | B.Tech. CSE | CCVT | 500067770 |

# Under the guidance of

Mr. Abhirup Khanna

Assistant Professor, Department of Systematics



SCHOOL OF COMPUTER SCIENCE UNIVERSITY OF PETROLEUM & ENERGY STUDIES

Bidholi Campus, Energy Acres, Dehradun – 248007

2021-22



**CANDIDATES’ DECLARATION**

I/We hereby certify that the project work entitled **TYPOO** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science And Engineering with Specialization in Cloud Computing and Virtualization Technology, and submitted to the Department of Virtualization at School of Computer Science, University of Petroleum And Energy Studies, Dehradun, is an authentic record of my/our work carried out during the period from **January, 2022** to **May, 2022** under the supervision **of Mr. Abhirup Khanna**, Assistant Professor, Department of Systematics.

The matter presented in this project has not been submitted by me/us for the award of any other degree of this or any other University.

## Prabal Bansal (R110218104)

**Raghav Jindal (R110218114)**

## Raman Kumar (R110218119)

## Sachin Agrawal (R110218129)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Date: 5th May, 2022) **Mr. Abhirup Khanna**

## (Project Guide)

**ACKNOWLEDGEMENT**

We wish to express our deep gratitude to our guide **Mr. Abhirup Khanna**, for all advice, encouragement and constant support he has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

## Prabal Bansal (R110218104)

**Raghav Jindal (R110218114)**

## Raman Kumar (R110218119)

## Sachin Agrawal (R110218129)



School of Computer Science

**University of Petroleum & Energy Studies, Dehradun**

# Major: II

**PROJECT TITLE: TYPOO**

## ABSTRACT

A full-stack blogging web application made with node and express js and backend with MongoDB with all the CRUD operations. Users can register themselves and write their own blog from their dashboard and the admin will be able to feature that blog on the running website. We also added the functionality of listing blog according to certain topic or interest in the home page itself which will add more look and field to the application.  
Users can register them in the application with the social media as well as mobile number. For social media we have added the functionality of sign in with google and Facebook. And for the mobile number functionality we have added the twillio which will send an OTP the registered mobile number.

Users can update their profile data with all the basic details like the name and password, Users can also update their profile photo by selecting a png file from their system.

**Keywords**: Full-Stack , Database, React, Front-end .

# TABLE OF CONTENT

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Topics** | **Page No.** |
| 1 | Introduction | 7 |
| 2 | Problem Statement | 10 |
| 3 | Literature Review | 11 |
| 4 | Objectives | 17 |
| 5 | Software/Hardware Requirements | 18 |
| 6 | Methodology | 19 |
| 7 | Workflow | 22 |
| 8 | TechStack Used | 24 |
| 9 | SWOT Analysis | 30 |
| 10 | Website UI | 34 |
| 11 | Conclusion and Future scope | 41 |
| 12 | Pert Chart | 42 |
| 13 | References | 43 |

**TABLE OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig No.** | **Figures** | **Page No.** |
| 1 | Methodology | 19 |
| 2 | Agile | 21 |
| 3 | Workflow | 22 |
| 4 | UI Dataflow | 23 |
| 5 | MERN | 25 |
| 6 | Mongo | 26 |
| 7 | Express | 27 |
| 8 | React | 28 |
| 9 | NOde | 29 |
| 10 | User data | 32 |
| 11 | Homepage | 34 |
| 12 | Register Page | 35 |
| 13 | Login Page | 32 |
| 14 | Posting comments | 33 |
| 15 | Create Post | 33 |
| 16 | Sample Blog | 35 |
| 17 | MongoDB | 38 |
| 18 | User data in mongoDB | 39 |
| 19 | Categories in MongoDB | 39 |
| 20 | Posts in MongoDB | 39 |
| 21 | Comments in MongoDB | 40 |

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Figures** | **Page No.** |
| 1 | Pert Chart | 43 |

**INTRODUCTION**

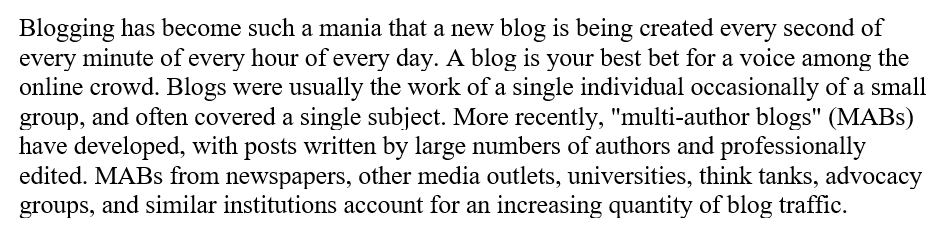
As we know that most of the platforms are build up of combination of multiple component relying on different middleware that are API’s.

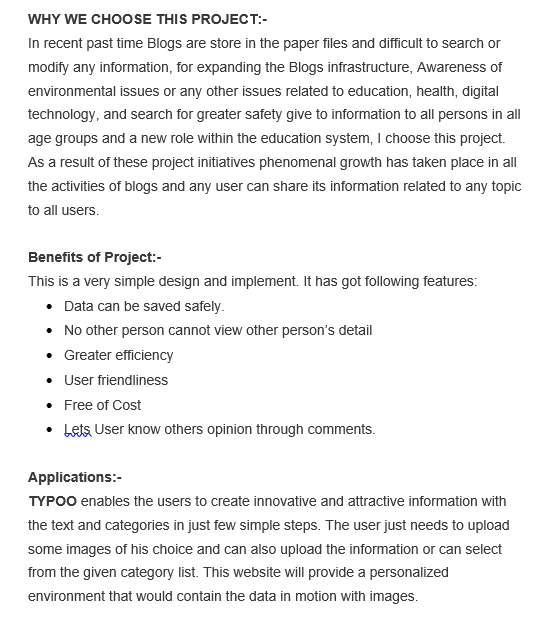
Due to these API’s a lot of work get easier as a lot of complex task are handled by these middleware.

People are able to develop product in short period of time with the help of them and are able to deliver the product more quickly as compared to when they have to create the product from scratch comprising all the features as well as all the different component of a product.

Keeping that in mind and further integrating with components like Node.js which is capable of server side scripting and MongoDB capable of fetching multiple request at the same time and giving a lot of flexibility with the Database to play around , combined these we can create applications that are more reliable and useful at the same time and can be scaled at any moment of time.

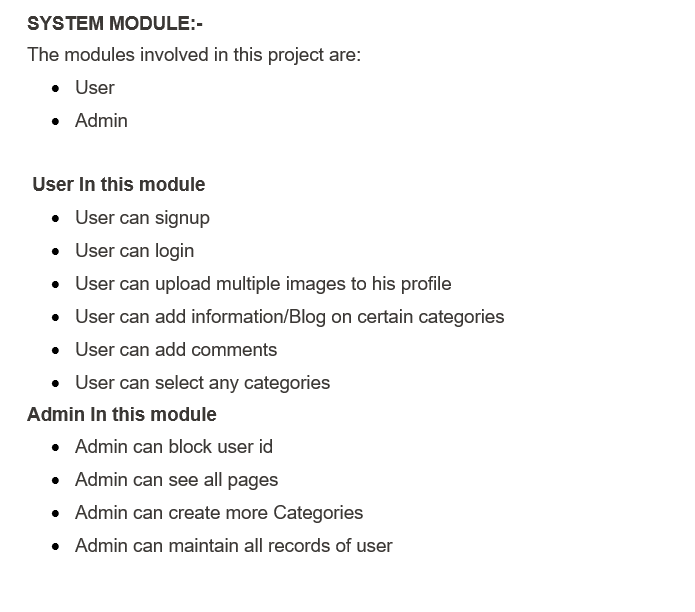
A full-stack blogging web application made with node and express js and backend with MongoDB with all the CRUD operations. Users can register themselves and write their own blog from their dashboard and the admin will be able to feature that blog on the running website. We also added the functionality of listing blog according to certain topic or interest in the home page itself which will add more look and field to the application.



****

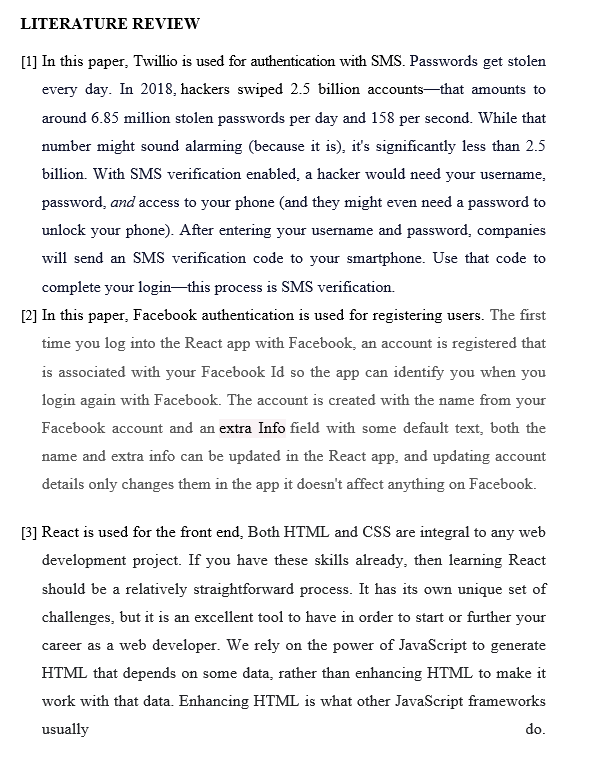
**Scope of the Project:-**

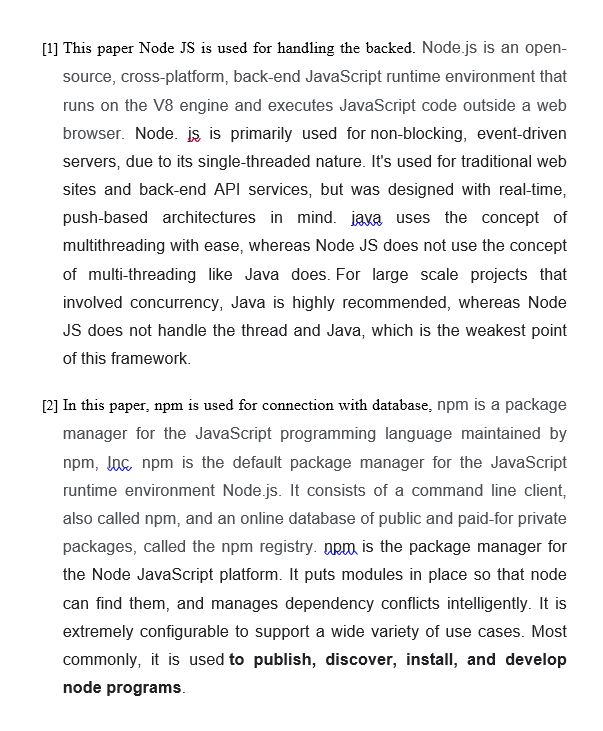
* To Share Technical knowledge
* Contributing to open source
* Share your opinion on internet
* Posting comments.
* To merge users according to certain interest
* Help people to collaborate.

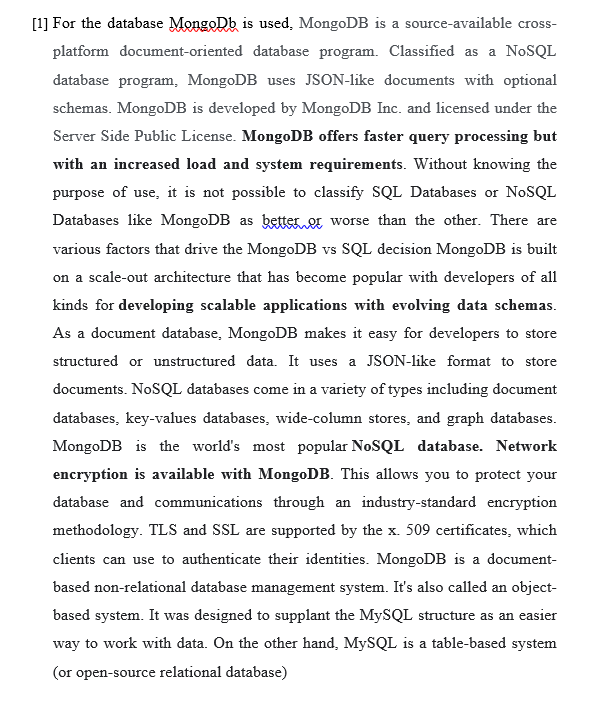
****

# PROBLEM STATEMENT

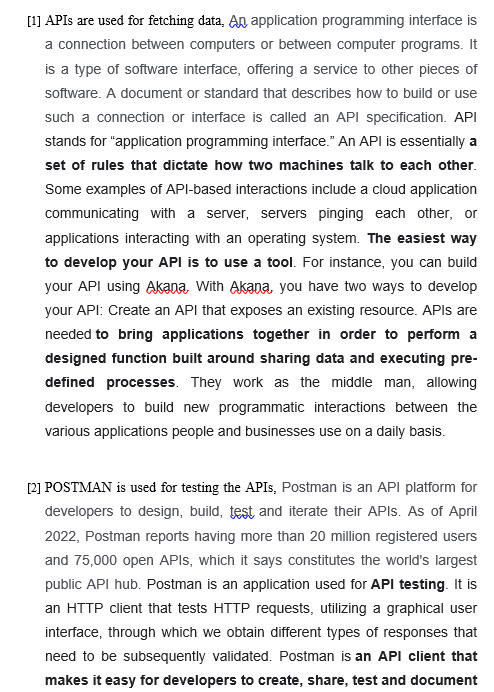
* With this project we are trying to bring together communities with intellect in different fields having same interest of knowledge.
* This project is open to people who wants to share their knowledge in a particular field.
* To share individual thoughts on the internet.

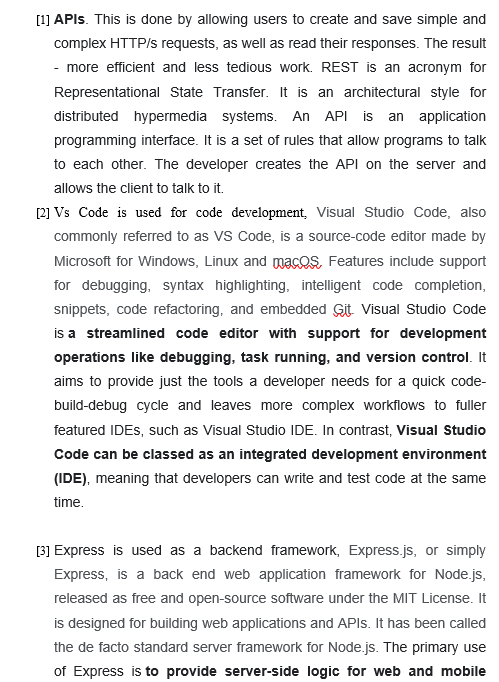
****

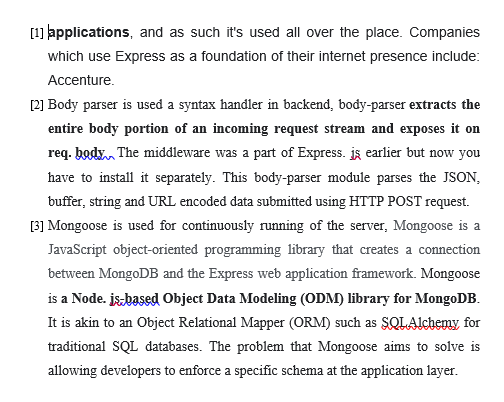










****

# OBJECTIVES

Now the final object here will be to make a dashboard that will let the people do ---

1. Register, login with Email or Phone number.
2. Quick login with Google, Facebook, SMS.
3. Forgot password, reset password and register a new account by Email or SMS verification.
4. Update personal information (name, password and avatar)
5. Create new blog with React quill.
6. Comment real-time with Socket.io
7. Pagination, search with autocomplete Mongodb

# SOFTWARE / HARDWARE REQUIREMENTS

# Software Requirements:-

# Vs Code

# Npm

# nodeJs

# ReactJs

# MongoDb

# MOngoDb Atlas

# GitHub

# Postman

# Hardware Requirements:-

1. **Minimum Hardware Configurations :-**  
   Processor: 800MHz Intel Pentium III or equivalent

Memory: 512 MB

Disk space: 750 MB of free disk space

Screen resolution is 1024x768 pixels

1. **Recommended Hardware Configurations**

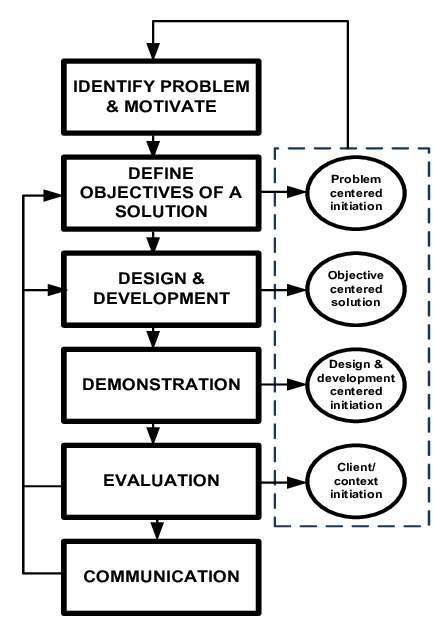
Processor: Intel Core i5 or equivalent.

Memory: 2 GB (32-bit), 4 GB (64-bit)

Disk space: 1.5 GB of free disk space

Screen resolution is 1024x768 pixels

# METHODOLOGY

****

**Fig.1 Methodology**

The entire implementation of this project can be summarized into the following steps:

1. **Agile methodology** of software development will be followed for the proposed project.

The project is divided into **12 sprints** where the sprint 7 and 8 will consist of parallel development by different members of the team.

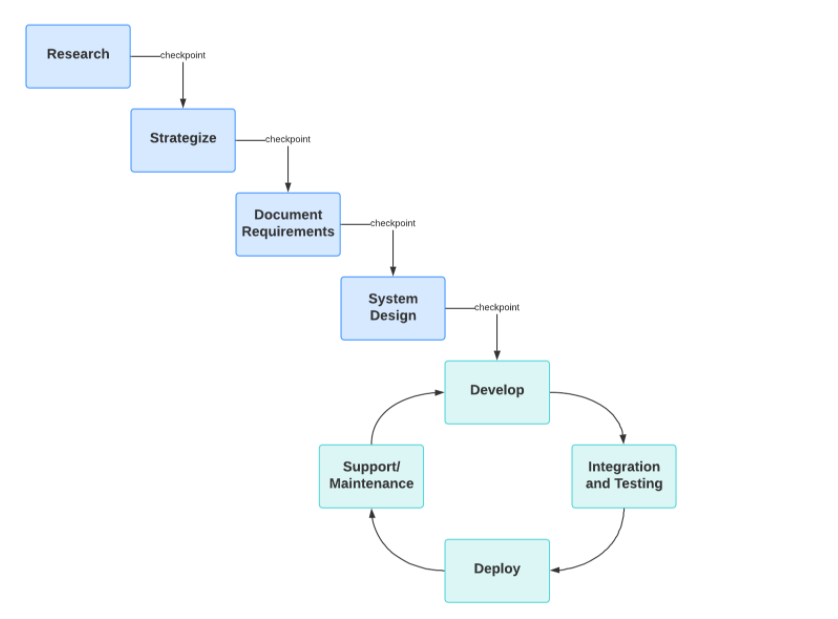
1. Each sprint is provided ample time to complete itself as well as to maintain the product's backlog (if any).
2. The project can **accommodate changes** if required at any stage of the project. The sprints 1, 2 and 3 are specifically for requirement analysis and designing of the project. One sprint is specifically designed for setting the environment like maintaining the **Version Control (Git in our case) .**
3. Each development sprint is followed by **Unit Testing** and an **Integration Testing** at the end. **Sprints** are also designed for the reviewing as well as retrospection part.
4. Overall, the time for the project is dedicated to an approach where the beginning time is dedicated towards the requirement analysis and the documentation part and during the implementation part all the team members are following their dedicated sprints cycles to implement the functionality.
5. After the implementation, testing is to be done for the whole application. Finally, the application is deployed with the documentation.

**Agile methodology** of software development will be followed for the proposed project.

The project is divided into **12 sprints** where the sprint 7 and 8 will consist of parallel development by different members of the team. Each sprint is provided ample time to complete itself as well as to maintain the product's backlog (if any). The project can **accommodate changes** if required at any stage of the project. The sprints 1, 2 and 3 are specifically for requirement analysis and designing of the project. One sprint is specifically designed for setting the environment like maintaining the **Version Control (Git in our case) .**

Each development sprint is followed by **Unit Testing** and an **Integration Testing** at the end. **Sprints** are also designed for the reviewing as well as retrospection part.

Overall, the time for the project is dedicated to an approach where the beginning time is dedicated towards the requirement analysis and the documentation part and during the implementation part all the team members are following their dedicated sprints cycles to implement the functionality. After the implementation, testing is to be done for the whole application. Finally, the application is deployed with the documentation



**Fig.2 Agile**

# WORKFLOW

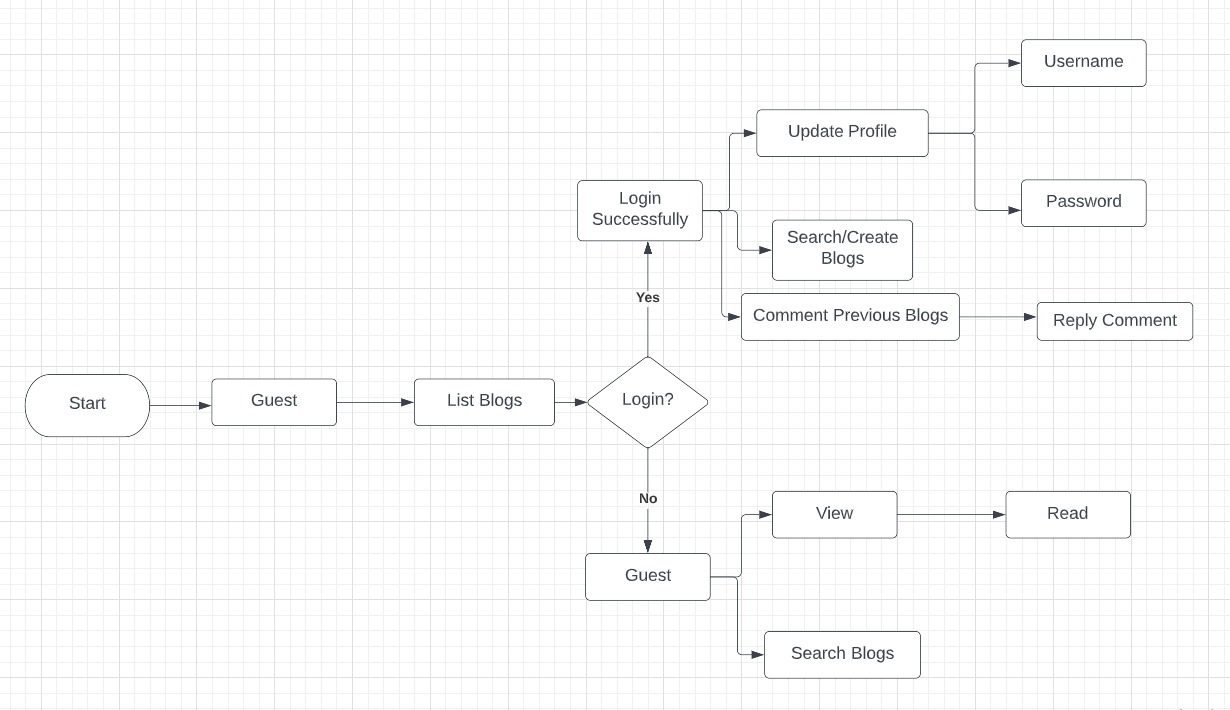
****

Fig.3 Workflow

For Web Application’s User Interface:

* 1. First, User gets to see the home page where all the featured blogs are present according to the category
  2. User can register himself with all the three options available
  3. Users data gets stored in mongodb
  4. User can then change his/her profile settings
  5. User can post blogs according to the different categories
  6. User can post blogs on others posts.

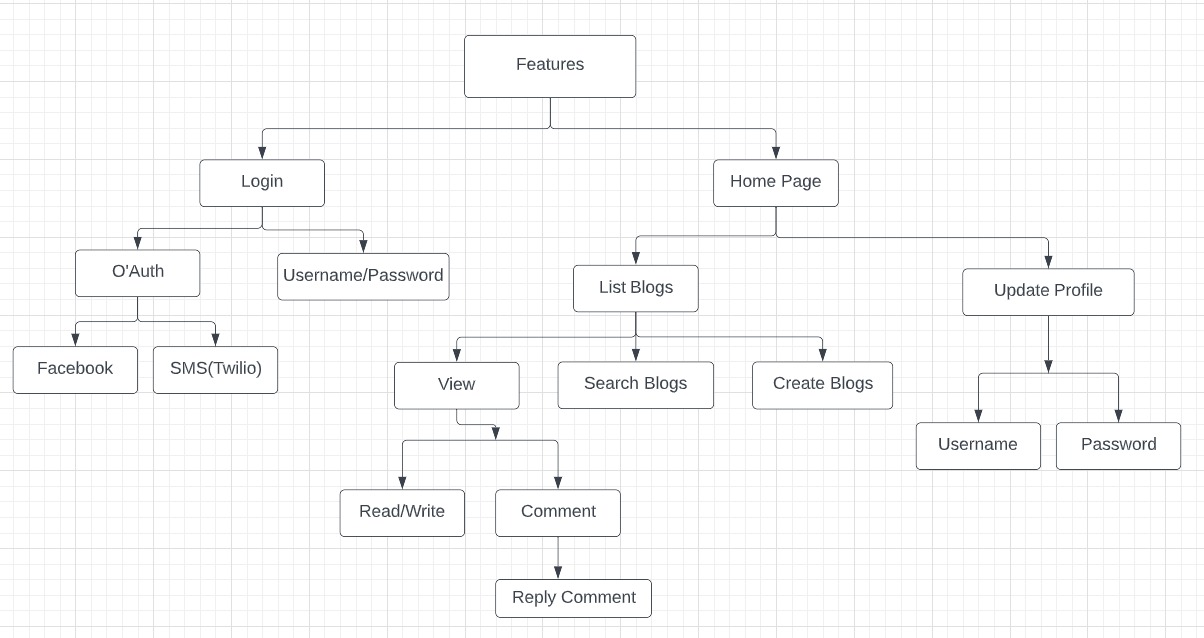
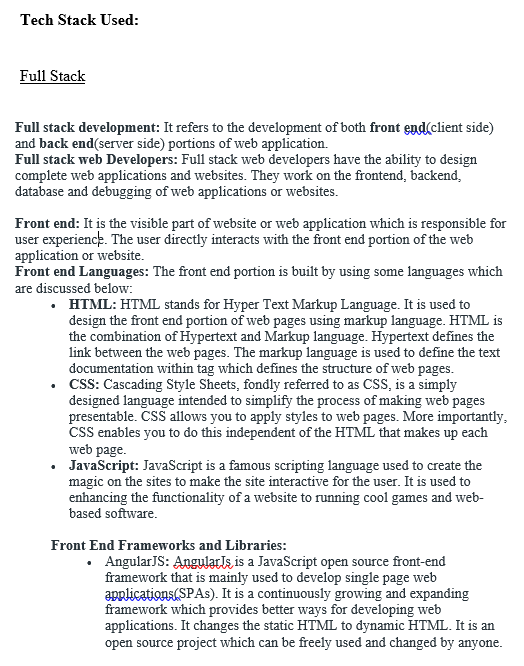
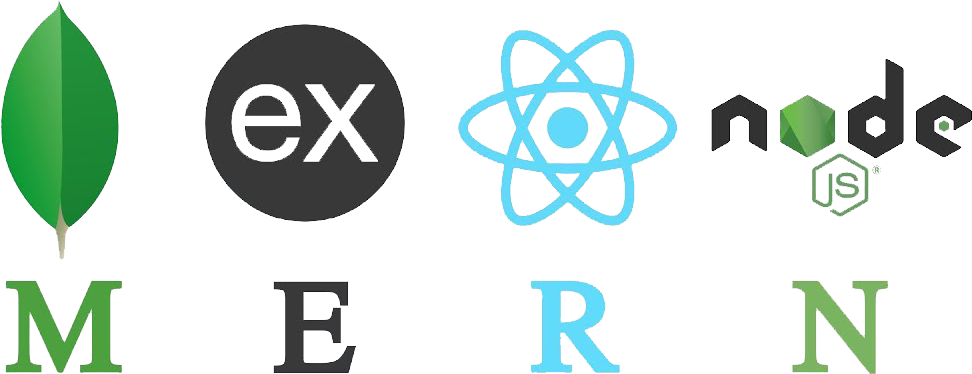
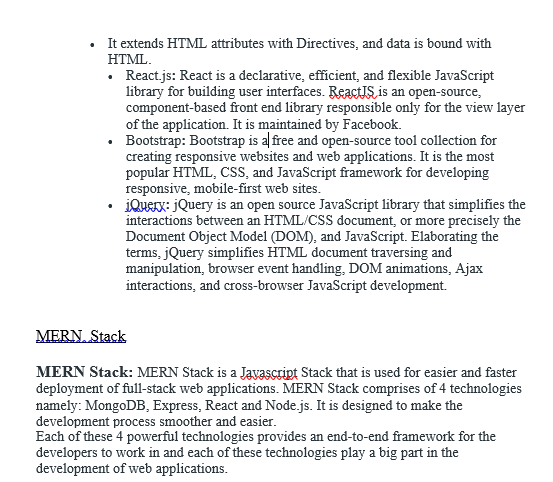


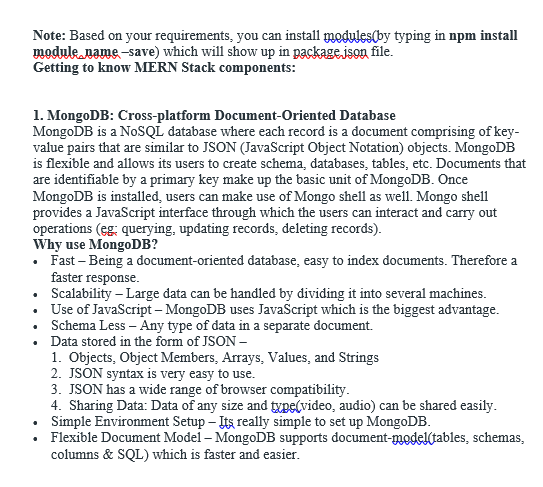
Fig.4 UI Dataflow

* + 



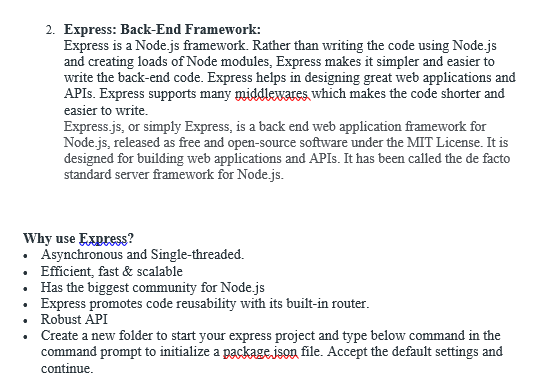
**Fig.5 MERN**

s

****

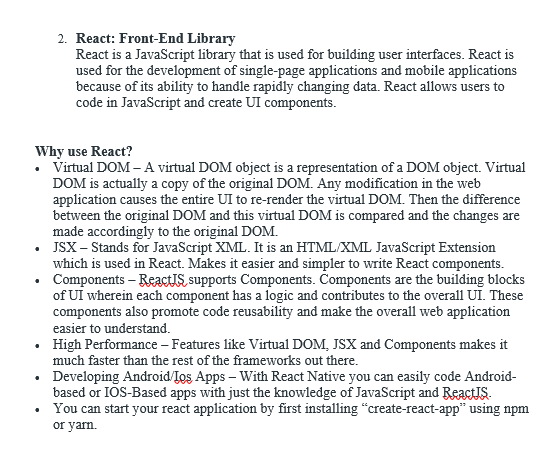


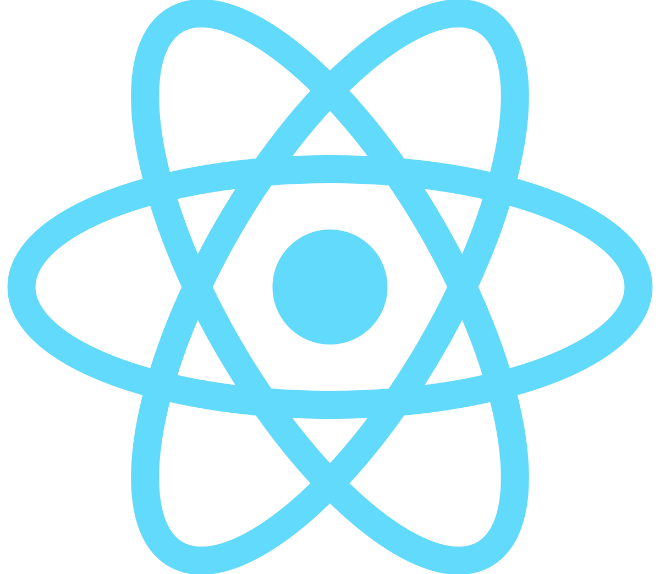
**Fig.6 mongo**

****

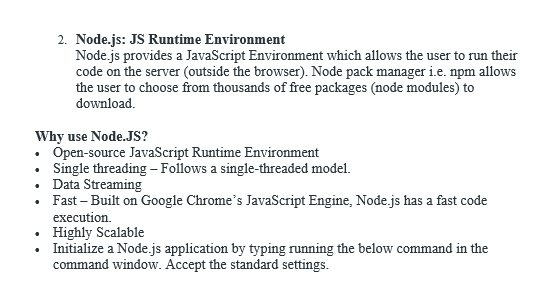


**Fig.7 express**

****

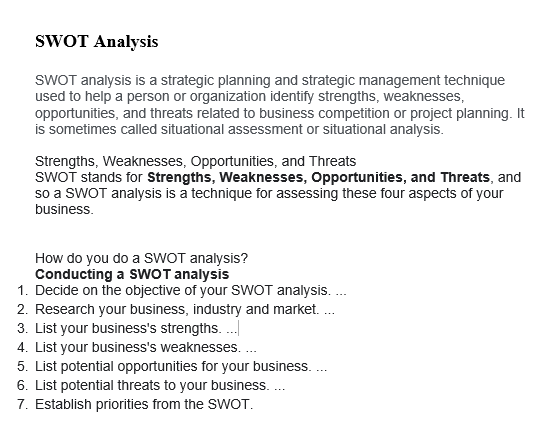


**Fig.8 react**

****



**Fig.10 nodejs**

****

**Strengths:**

* Merging users with similar interests.
* Quick login through Facebook and SMS.
* Working with non-relational database i.e MONGODB which is fast and reliable.
* Working with newest tech stack like MERN which has a very good industrial scope.
* Implemented industry best practice while coding.
* Version controlling through GitHub in our local environment.
* Worked with react which is a powerful frontend framework widely used.
* Created different database for different users which helps us to remove any proxy contents
* Stored passwords in encrypted form.

# Weakness:-

# Group similar users will affect in our website traffic

# Categories can only be created by the admin which enables the user to post only specific categories

# Complexity and folder structure gets complicated.

# Dependency may become old after some time, so we may need to update the files after some time.

# Mondo doesn’t work good when there’s relational data.

# Need to depend on third party apps for some functionality of our project.

# Website may take to load sometimes.

# Opportunities:

# Get More Clients to Your Existing Business

# Get Immediate Feedback

## Become a Better Writer

## Become a Published Author

## Become a Published Author

## Launch a Blog And Be An Influencer (Create a better world)

# Threats:

# Injection Attack

# Broken authentication

# Cross site scripting

# Insecure direct object reference

# Security misconfiguration.

# Algorithm used for Password storage:

# Bcrypt: Bcrypt operates in a very similar manner to more traditional schemes based on algorithms like PBKDF2. The main difference is its use of a derived key to encrypt known plain text; other schemes (reasonably) assume the key derivation function is irreversible, and store the derived key directly.

# The takeaway is this: bcrypt is a secure algorithm but remember that it caps passwords at 72 bytes. You can either check if the passwords are the proper size, or opt to switch to argon2, where you'll have to set a password size limit.

# What is bcrypt ? bcrypt was designed by Niels Provos and David Mazières based on the Blowfish cipher>): b for Blowfish and crypt for the name of the hashing function used by the UNIX password system. crypt is a great example of failure to adapt to technology changes

# 

**Fig.11 user data**

# UI (USER INTERFACE)

Technologies Used:

* HTML
* CSS
* Bootstrap
* Flask

Logo created using:

* Adobe Photoshop
* Adobe Illustrator

**Website UI (Screenshots)**

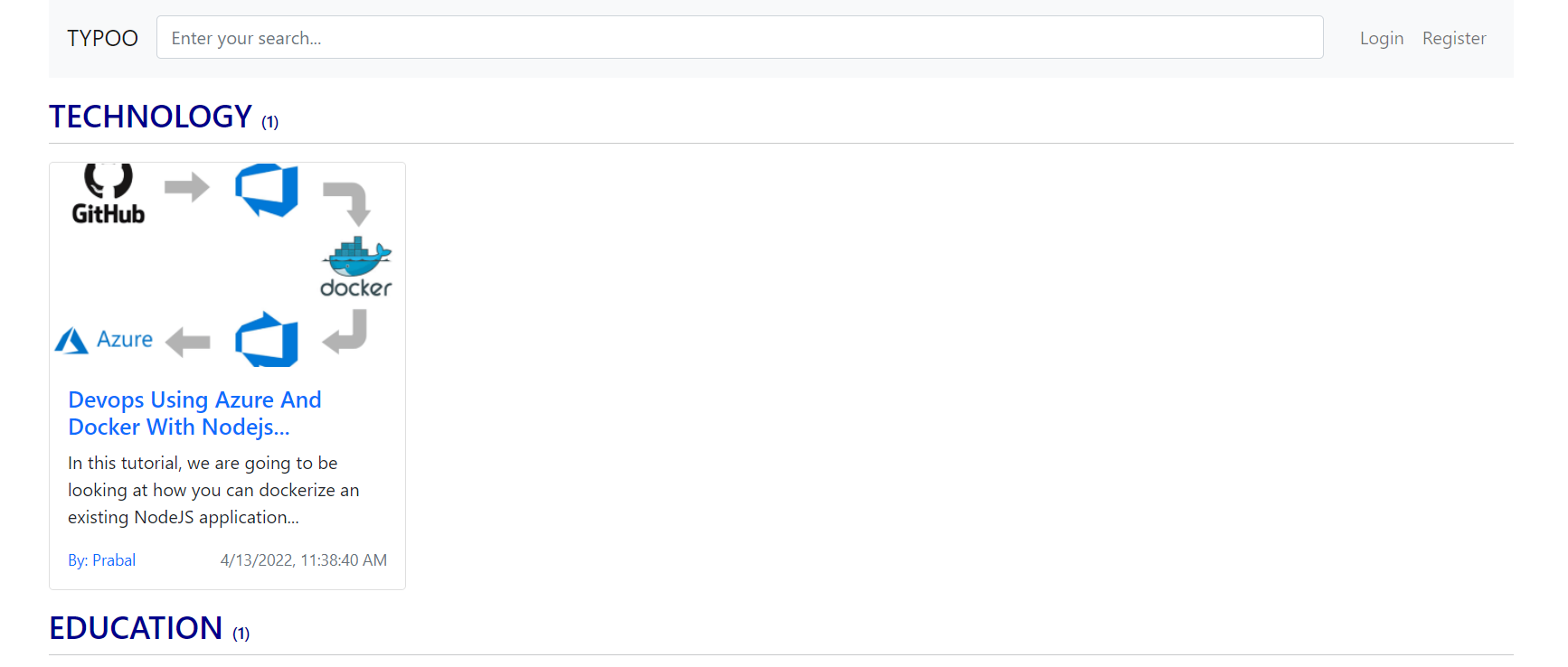


Fig.12 UI Homepage

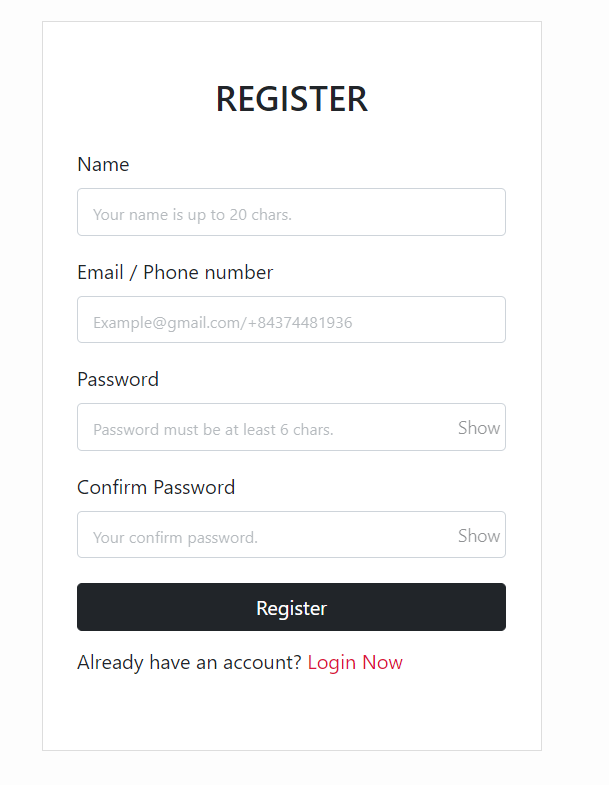


Fig. 13 Register Page

# 

Fig.14 UI Login Page

# 

Fig.15 UI Create Post Page

# 

Fig.16 UI Posting Comments

# 

Fig.17 UI Sample Blog

# MongoDB database

# User Data-Model

# 

# Categories Data-Model

# 

# Comments Data-Model

# 

# Users Data-Model

# 

# CONCLUSION

As we know that most of the platforms are build up of combination of multiple component relying on different middleware that are API’s.

Due to these API’s a lot of work get easier as a lot of complex task are handled by these middleware.

People are able to develop product in short period of time with the help of them and are able to deliver the product more quickly as compared to when they have to create the product from scratch comprising all the features as well as all the different component of a product.

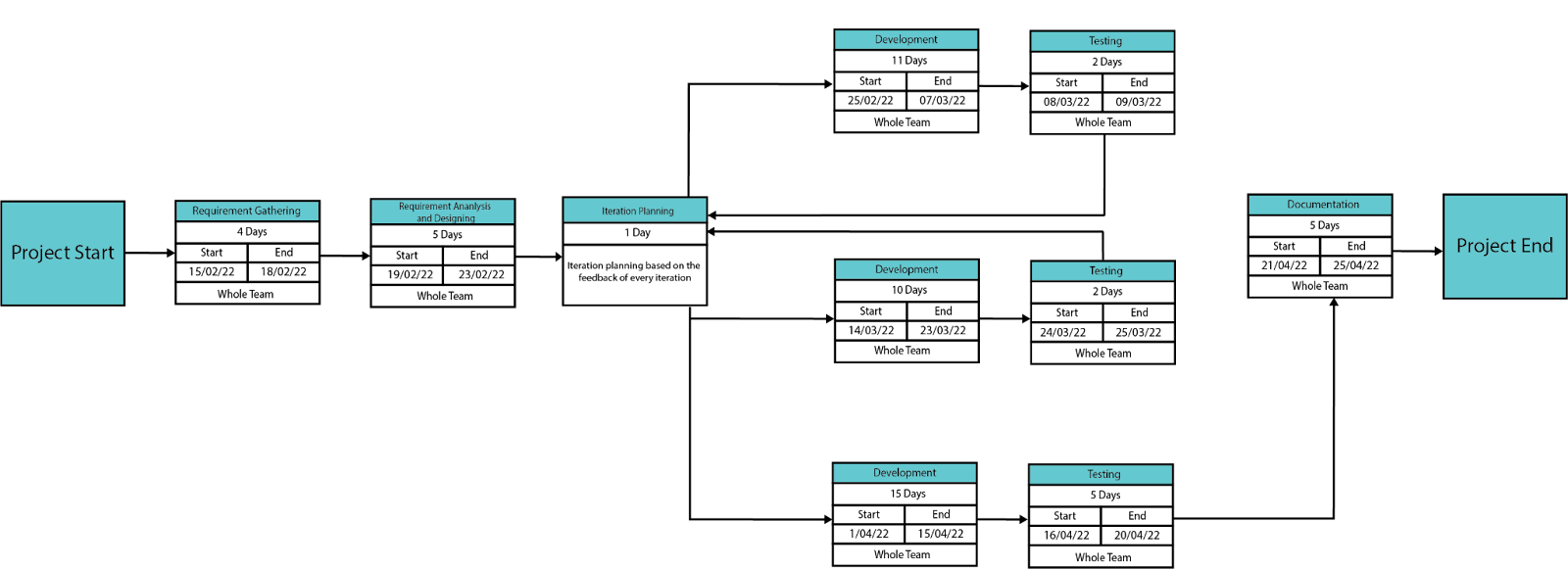
Keeping that in mind and further integrating with components like Node.js which is capable of server side scripting and MongoDB capable of fetching multiple request at the same time and giving a lot of flexibility with the Database to play around , combined these we can create applications that are more reliable and useful at the same time and can be scaled at any moment of time.

# FUTURE SCOPE

We will enhance the functionality of all these operations we implemented in this project like -- Register, login with Email or Phone number. Quick login with Google, Facebook, SMS. Forgot password, reset password and register a new account by Email or SMS verification. Update personal information (name, password and avatar).Create new blog with React quill. Comment real-time with Socket.io. Pagination, search with autocomplete Mongodb.

Keeping that in mind and further integrating with components like Node.js which is capable of server side scripting and MongoDB capable of fetching multiple request at the same time and giving a lot of flexibility with the Database to play around , combined these we can create applications that are more reliable and useful at the same time and can be scaled at any moment of time.

# PERT CHART



**Fig.21 Pert Chart**

# REFERENCES

[1] Office for National Statistics, Internet users in the UK: 2016. Retrieved September 26, 2017, from https:// www.ons.gov.uk/ businessindustryandtrade/itandinternetindustry/bulletins/internetus ers/ 2016.

[2] Liang, L., Zhu, L., Shang, W., Feng, D., Xiao, Z. (2017). Express supervision system based on NodeJS and MongoDB. International Journal of Engineering Research & Technology (IJERT) http://www.ijert.org ISSN: 2278-0181 IJERTV10IS060239 (This work is licensed under a Creative Commons Attribution 4.0 International License.) Published by : www.ijert.org Vol. 10 Issue 06, June-2021 714

[3] M. R. Solanki, A. Dongaonkar, A Journey of human comfort: web1.0 to web 4.0, International Journal of Research and Scientific Innovation (IJRSI), Volume III, Issue IX, pp. 75-78, 2016

[4] Javeed, A. (2019). Performance Optimization Techniques for ReactJS. 2019

[5] J. M. Spool, Content and design are inseparable work partners, 2014. Retrieved September 29, 2017, fromhttps://articles.uie.com/ content and design

[6] Bozikovic, H., Stula, M. (2018). Web design Past, present and future. 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).

[7] Carter, B. (2014). HTML Architecture, a Novel Development System (HANDS): An Approach for Web Development. 2014

[8] Sterling, A. (2019). NodeJS and Angular Tools for JSON-LD. 2019 IEEE 13th

[9] Laksono, D. (2018). Testing Spatial Data Deliverance in SQL and NoSQL Database Using NodeJS Fullstack Web App. 2018

[10] Patil, M. M., Hanni, A., Tejeshwar, C. H., Patil, P. (2017). A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web/android application to explore load balancing Sharding in MongoDB and its advantages

[11] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, Image analogies, in Proc. 28th Annu. Conf. Comput. Graph. Interact. Tech., 2001, pp. 327-340.

[12] ”Tags used in HTML”. World Wide Web Consortium. November 3, 1992. Retrieved November 16, 2008.

[13] R. Irony, D. Cohen-Or, and D. Lischinski, Colorization by example, in Proc. Eurograph. Symp. Rendering, vol. 2. 2005, pp. 201-210.

[14] First mention of HTML Tags on the www-talk mailing list”. World Wide Web Consortium. October 29, 1991. Retrieved April 8, 2007.

[15] JavaScript specification. Retrieved from http://www.w3.org/standards/ webdesign/script, November 1, 2014. International Journal

[16] Bass, B.M. (1990), “From transactional to transformational leadership: learning to share the vision”, Organizational Dynamics, Vol. 18 No. 3, pp. 19-31.

[17] Bin, B.S. and Park, J.K. (2002), “An empirical study on the success factors of a small business startingup”, The Asia Pacific journal of Small Business, Vol. 24 No. 3, pp. 135-158

[18] Nodejs.org. 2020. Nodejs. Accessed on 28 April 2020 [online] Available at: <https://nodejs.org/>

[19] Jyoti Shetty, Deepika Dash, Akshaya Kumar Joish, Guruprasad C “Review Paper on Web Frameworks, Databases and Web Stacks” 2020 IRJET-V7141078

[20] Pro MERN Stack Full Stack Web App Development with Mongo, Express, React, and Node by Vasan Subramaniam

[21] Getting MEAN with Mongo, Express, Angular, and Node Book by Simon Holmes

[22] Angular.io,’AngularJS Documentation’ [Online]. Available: https://angular.io. [Accessed: May- 2021]

[23] MongoDB.com,’MongoDB Official’ [Online]. Available: https://www.mongodb.com/. [Accessed: May- 2021]

[24] Expressjs.com,’ExpressJS Official’ [Online]. Available: http://expressjs.com. [Accessed: May- 2021]

[25] ReactJS.org,’ReactJS official’ [Online]. Available: http://reactjs.org. [Accessed: May- 2021]

[26] Nodejs.com,’Nodejs Official’ [Online]. Available: https://nodejs.org. [Accessed: May- 2021]

[27] K. Arnold. Programmers are people, too. Queue, 3 (5): 54-59, 2005.

[28] I. Balaban, F. Tip, and R. Fuhrer. Refactoring support for class library migration. SIGPLAN Not., 40 (10): 265-279, 2005.

[29] T. Biggerstaff. The library scaling problem and the limits of concrete component reuse. Software Reuse: Advances in Software Reusability, 1994. Proceedings., Third International Conference on, pages 102-109, 1-4 Nov 1994.

[30] J. Bloch. How to design a good API and why it matters. In OOPSLA '06: Companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, pages 506-507, New York, NY, USA, 2006. ACM.

[31] C. Bore and S. Bore. Profiling software API usability for consumer electronics. Consumer Electronics, 2005. ICCE. 2005 Digest of Technical Papers. International Conference on, pages 155-156, 8-12 Jan. 2005.

[32] S. Clarke and C. Becker. Using the cognitive dimensions framework to evaluate the usability of a class library. In Joint Conf. EASE & PPIG, Petre & D. Budgen (Eds), pages 359-366, 2003.

[33] D. Conway. Ten essential development practices. 14 July 2004.

[34] S. Demeyer, T. D. Meijler, O. Nierstrasz, and P. Steyaert. Design guidelines for tailorable frameworks. Commun. ACM, 40 (10): 60-64, 1997.

[35] B. Ellis, J. Stylos, and B. Myers. The factory pattern in API design: A usability evaluation. In ICSE '07: Proceedings of the 29th International Conference on Software Engineering, pages 302-312, Washington, DC, USA, 2007. IEEE Computer Society.

[36] M. Fayad and D. C. Schmidt. Object-oriented application frameworks. Commun. ACM, 40 (10): 32-38, 1997.

[37] D. Garlan, R. Allen, and J. Ockerbloom. Architectural mismatch or why it's hard to build systems out of existing parts. In ICSE '95: Proceedings of the 17th international conference on Software engineering, pages 179-185, New York, NY, USA, 1995. ACM.

[38] E. R. Harold. XOM design principles. Extreme Markup Languages, 2-6 August 2004.

[39] J. Henkel and A. Diwan. CatchUp!: capturing and replaying refactorings to support API evolution. In ICSE '05: Proceedings of the 27th international conference on Software engineering, pages 274-283, New York, NY, USA, 2005. ACM.

[40] M. Henning. API design matters. Queue, 5 (4): 24-36, 2007.

[41] S. Henninger. Retrieving software objects in an examplebased programming environment. In SIGIR'91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval, pages 251-260, New York, NY, USA, 1991. ACM.

[42] R. Holmes, R. Walker, and G. Murphy. Approximate structural context matching: An approach to recommend relevant examples. Software Engineering, IEEE Transactions on, 32 (12): 952-970, Dec. 2006.

[43] D. Mandelin, L. Xu, R. Bodík, and D. Kimelman. Jungloid mining: helping to navigate the API jungle. In PLDI '05: Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation, pages 48{61, New York, NY, USA, 2005. ACM.

[44] S. McLellan, A. Roesler, J. Tempest, and C. Spinuzzi. Building more usable APIs. Software, IEEE, 15 (3): 78- 86, May/Jun 1998.

[45] Microsoft Developer Network. Designing .NET class libraries: Designing progressive APIs. MSDN Online Chat, 02 March 2005.

[46] L. R. Neal. A system for example-based programming. In CHI '89: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 63-68, New York, NY, USA, 1989. ACM.

[47] L. R. Neal. A system for example-based programming. In CHI '89: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 63-68, New York, NY, USA, 1989. ACM.

[48] D. L. Parnas. On the criteria to be used in decomposing systems into modules. Commun. ACM, 15 (12): 1053- 1058, 1972.

[49] D. L. Parnas and D. P. Siewiorek. Use of the concept of transparency in the design of hierarchically structured systems. Commun. ACM, 18 (7): 401-408, 1975.

[50] B. Pryor. Simple concurrency guidelines. Ben Pryor's Blog, March 2008.

[51] Y. Smaragdakis and D. Batory. Mixin layers: an objectoriented implementation technique for refinements and collaboration-based designs. ACM Trans. Softw. Eng. Methodol., 11 (2): 215-255, 2002

[52] J. Spolsky. How Microsoft lost the API war. In Business of Software 2008, a JOEL ON SOFTWARE Conference, Boston, MA, United States, 13 June 2004.

[53] J. Stylos and S. Clarke. Usability implications of requiring parameters in objects' constructors. In ICSE'07: Proceedings of the 29th International Conference on Software Engineering, pages 529-539, Washington, DC, USA, 2007. IEEE Computer Society.

[54] J. Stylos and B. Myers. Mapping the space of API design decisions. Visual Languages and Human-Centric Computing, 2007. VL/HCC 2007. IEEE Symposium on, pages 50-60, 23-27 Sept. 2007.

[55] J. van Gurp and J. Bosch. Design, implementation and evolution of object oriented frameworks: concepts and guidelines. Softw. Pract. Exper., 31 (3): 277-300, 2001.

[56] T. Xie and J. Pei. MAPO: mining API usages from open source repositories. In MSR '06: Proceedings of the 2006 international workshop on Mining software repositories, pages 54-57, New York, NY, USA, 2006. ACM.

[57] Z. Xing and E. Stroulia. API-evolution support with DiffCatchUp. Software Engineering, IEEE Transactions on, 33 (12): 818-836, Dec. 2007.