# INTERNSHIP-II REPORT

**on**

# CBIT STACK OVERFLOW

**Submitted in partial fulfilment for the completion of**

**BE-V Semester**

**In**

**INFORMATION TECHNOLOGY**

**By**

**P. MANOJ KUMAR (160120737157)**

**Under the guidance of**

**Mr. G. SRIKANTH**
**Assistant Professor**

**DEPARTMENT OF INFORMATION TECHNOLOGY**
**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**
**(Affiliated to Osmania University; Accredited by NBA(AICTE) and NAAC(UGC), ISO Certified 9001:2015)**
**GANDIPET, HYDERABAD – 500 075**
**Website: www.cbit.ac.in**

**2022-2023**


# CERTIFICATE

This is to certify that the project seminar report entitled "**CBIT STACK OVERFLOW**" submitted to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY,** in partial fulfilment of the requirements for the award of the completion of V semester of B.E in Information Technology, during the academic year 2022-2023, is a record of original work carried out by **P. MANOJ KUMAR (160120737157)** during the period of study in Department of IT, CBIT, Hyderabad.

**Mentor**

**Mr. G. Srikanth**

Asst. Professor, Dept. of IT,

CBIT, Hyderabad.

**Head of the Department**

**Dr. K. Radhika**

Professor, Dept. of IT,

CBIT, Hyderabad.

# DECLARATION

This is to certify that the work reported in the present report titled "**CBIT STACK OVERFLOW**" submitted in partial fulfillment for the completion of B.E., V Semester, in the department of Information Technology, Chaitanya Bharathi Institute of Technology, Hyderabad, is a record of original work.

No part of the report is copied from books / journals / internet and wherever the portion is taken, the same has been duly referred. The reported results are based on the project work done entirely by us and not copied from any other source.

**P. MANOJ KUMAR (160120737157)**

# ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Mr. G. Srikanth**, our Mentor, for his invaluable guidance and constant support, along with his capable instruction and persistent encouragement.

We are grateful to our Head of Department, **Dr. K. Radhika**, for her steady support and for the provision of every resource required for the completion of this project.

We would like to take this opportunity to thank our Principal, **Dr. P. Ravinder Reddy**, as well as the Management of the Institute, for having designed an excellent learning atmosphere.

We would also like to express our heartfelt gratitude to each person who supported us and helped us in the completion of the project.

# ABSTRACT

Question Answers forums are becoming popular as platforms for connecting a sizable user base with one another around shared interests. There are guidelines on how members can communicate on the forum, which uses a Question-and-Answer format. Users are utilizing Stack Overflow not only as a QA forum but also as a learning resource because of the abundance of questions available on the site. Thus, we planned to build a Stack Overflow website for our own college CBIT because we were curious about how users used the SO forum.

The tech stack used in the project is MERN which stands for MongoDB, Express, React, Node. As we enter a new decade, there are tremendous forces converging—cloud computing, big data, AI, ML, and an increasingly diverse group of young coders from around the world. Every day, millions of developers visit Stack Overflow to find information they need as they push these exciting new technologies forward.

But while Stack Overflow has played a massive role in empowering and enabling developers around the world to learn, write code, and build products faster, most of the organizations have not kept up with the evolution of the industry. The key to evolution is the millions of developers from around the world who find the site useful, but who haven’t yet been welcomed into the community. We need to expand the reach and engagement to ensure these developers join the conversation and push their own learning to new heights.

# TABLE OF CONTENTS

# LIST OF FIGURES

# 1. INTRODUCTION

## 1.1 OVERVIEW

The “CBIT Stack Overflow” website is basically designed to help the user ask questions that he is shy asking to other people. The website has the feature where the person can anonymously ask a question if he does not want to reveal his identity to the forum. With this website even the silliest of questions can be asked and there will be people who will be responding to that question. It helps people overcome the problem of not asking and getting their questions clarified. The website is useful for all genre’s of audiences.

## 1.2 APPLICATIONS

The website is useful in solving queries of students who are shy to ask to their respective faculty. It can also be used by students to answer or solve the query raised in the platform that can help their knowledge to grow. Irrespective of whether a student is having doubt or not he can go through the discussion related to a particular question and get better understanding of the topic. Sometimes coding questions are difficult to understand, such type of question can be discussed at a forum like our website. Another application is that questions related to a particular subject are all compiled at one place for better understanding.

## 1.3 PROBLEM DEFINITION

Waiting till the next class or the next time a teacher is free wastes a lot of time. Sometimes we shy away from clarifying doubts with our peers. Many students may be sharing the same doubt. We can help these students by providing them with a platform where they can post their queries (anonymously based on preference) and their peers, seniors, or faculty can clear their doubts.

## 1.4 AIM OF THE PROJECT

Individuals increasingly rely on their distributed peer communities for information, advice, and expertise. Millions of individuals learn from each other on public discussion forums (e.g., Usenet), community-built encyclopedias (e.g., Wikipedia), social networks (e.g., Aardvark), and online question and answer sites (e.g., Yahoo! Answers). Recently, several large Q&A sites have attracted the attention of researchers . In aggregate, these studies suggest that general-purpose Q&A sites have answer rates between 66% and 90%; often attract non-factual, conversational exchanges of limited archival value; and may be poorly suited to provide high quality technical answers. So we have taken Interest in understanding the usage of SO forum by users, we wanted to implement a query resolving web application for our college and name it under CBIT stack overflow.

# 2. SYSTEM REQUIREMENT SPECIFICATION

## 2.1 SOFTWARE REQUIREMENTS

- Code editor: Visual Studio Code
- Database: MongoDB Atlas
- Technologies: Html, CSS, JavaScript, ReactJS, NodeJS, Express JS
- Operating Systems: Windows 7

## 2.2 HARDWARE REQUIREMENTS

- OS : Windows 7 SP1
- RAM : 2GB
- STORAGE : 5GB
- INPUT DEVICE : Keyboard or touch screen.
- OTHER REQUIREMENTS: Graphical User Interface (GUI) or integrated video card.

## 3. PROPOSED SYSTEM / METHODOLOGY

- This website consists of a dashboard page, login page, registration page, ask question page, post answer page and user's page.
- If one is not the existing user of the website, then he/she should first register for the website by filling in the details.
- Once the user registers for the website, from then he/she can directly log in website Using their given credentials.
- As soon as the user logs into the website, they will be able to see the dashboard page.
- The dashboard page consists of multiple features which include asking a question with characteristics like adding filters and styling it, seeing all the users registered, different filters like newest question asked, oldest question asked. The page also consists of popular subject tags. Now if a person wants to answer a particular question, he needs to click on the question and post his answer below it. The user can either upvote or downvote the questions as well as answer according to their satisfaction.
- If the person wants to log out of the website, he can click on logout button present on the dashboard

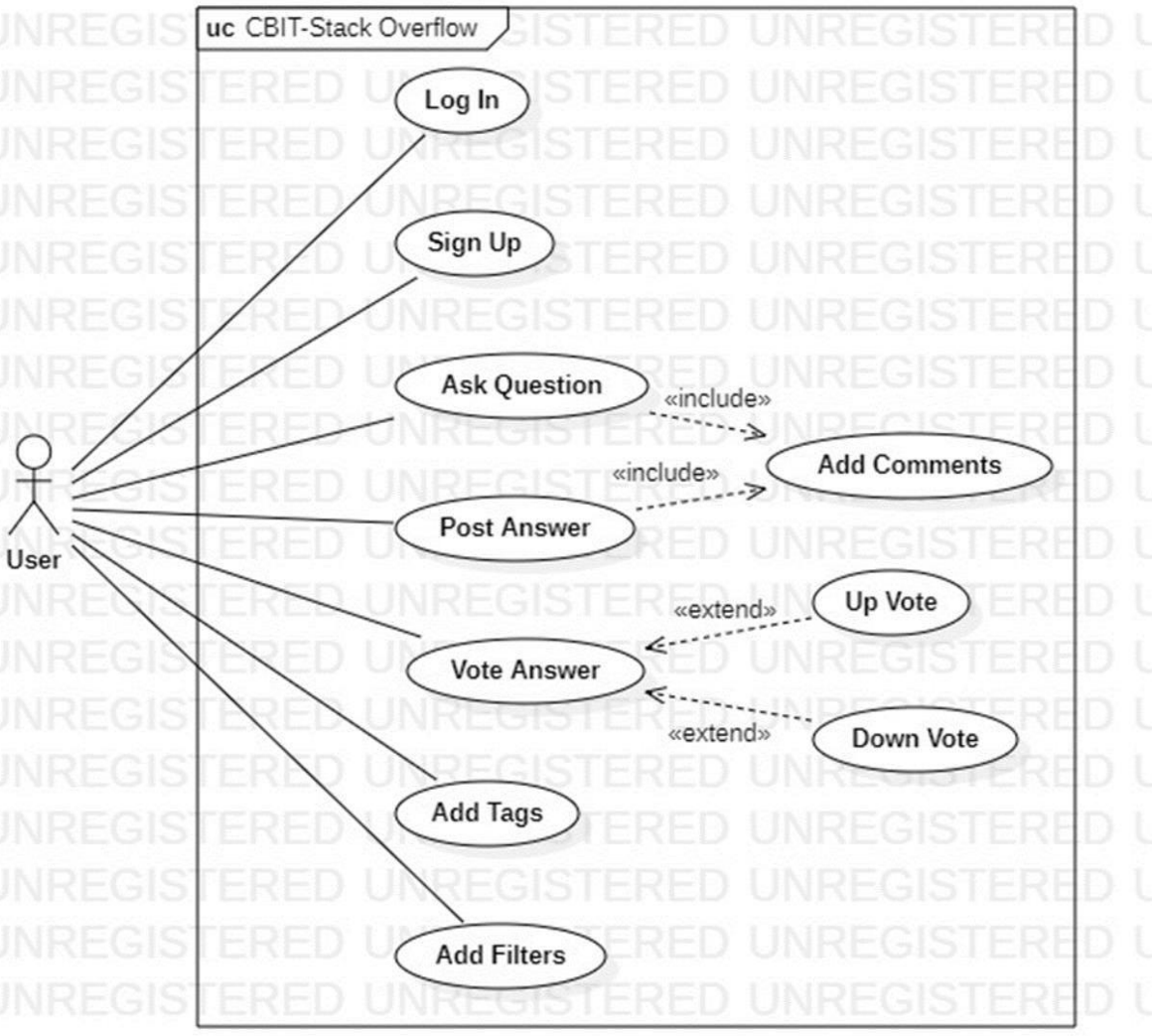



**Fig 3.1 System Design of the Proposed System**

The System Design of the proposed system focuses on looking at the system as a combination of many different components, and how they interact with each other to produce the desired result. First the user has to login into the website. If he\she is a first-time user of the application, they have to get registered for the website by clicking on 'Don't have an account?' in the login page. Both the login and registration page direct the user to the same dashboard page where he can access different features of the website.

# 4. IMPLEMENTATION

There are various functions which are included in the user login section, user registration section, and the dashboard section. These functions enable us to perform different tasks like registering one's self for the website, and later logging in to it, and using the features of stack overflow website.

There are three pages in this application namely login page , registration page and the dashboard page. The users have to register by entering their username and password. This username is unique to all the users and used in login page . This page helps to collect complete and relevant users' information.

The dashboard page has many features but to sum it up in short it includes asking a question and other users responding to it with different characteristics like tags, filters, votes styling etc.

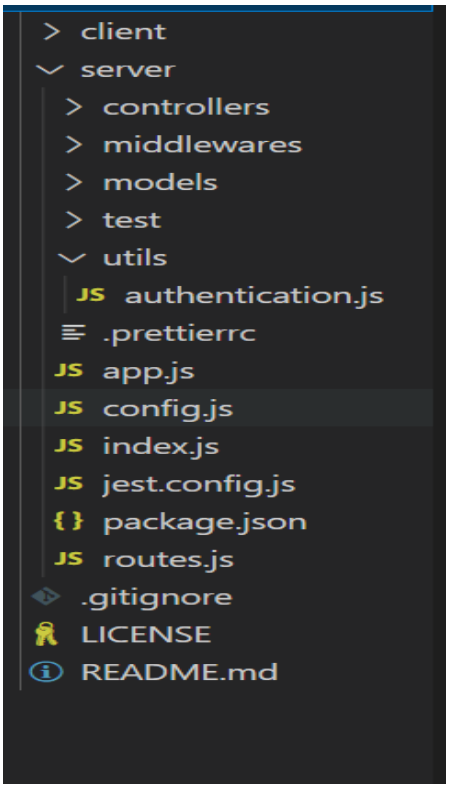



**Fig 4.1 Hierarchy of the project**

```
const LoginForm = () => {
  const { setAuthState } = useContext(AuthContext)
  const { setIsComponentVisible } = useContext(ModalContext)

  const [loading, setLoading] = useState(false)

  return (
    <Formik
      initialValues={{ username: '', password: '' }}
      onSubmit={async (values, { setStatus, resetForm }) => {
        setLoading(true)
        try {
          const { data } = await publicFetch.post('authenticate', values)
          const { token, expiresAt, userInfo } = data
          setAuthState({ token, expiresAt, userInfo })
          resetForm({})
          setIsComponentVisible(false)
        } catch (error) {
          setStatus(error.response.data.message)
        }
        setLoading(false)
      }}
      validationSchema={Yup.object({
        username: Yup.string()
          .required('Required')
          .max(16, 'Must be at most 16 characters long')
          .matches(/^[a-zA-Z0-9_-]+$/, 'Contains invalid characters'),
        password: Yup.string()
          .required('Required')
          .min(6, 'Must be at least 6 characters long')
          .max(50, 'Must be at most 50 characters long')
      })}
    >
```

**Fig 4.2 Login page (index.js)**

```
import React from 'react'
import Link from 'next/link'
import formatDistanceToNowStrict from 'date-fns/formatDistanceToNowStrict'
import slug from 'slug'

import Tag from '../../tag'

import styles from './question-summary.module.css'

const QuestionSummary = ({
  id,
  title,
  tags,
  author,
  createdTime,
  children
}) => {
  return (
    <div className={styles.container}>
      <Link href="/questions/[slug]" as={`/questions/${id}-${slug(title)}`}>
        <a className={styles.link}>{title}</a>
      </Link>
      <div className={styles.excerpt}>{children}</div>
      <div className={styles.footer}>
        <div className={styles.tagContainer}>
          {tags.map((tag) => (
            <Tag key={tag}>{tag}</Tag>
          ))}
        </div>
        <div className={styles.userDetails}>
          <Link href="/users/[user]" as={`/users/${author.username}`}>
            <a>
```

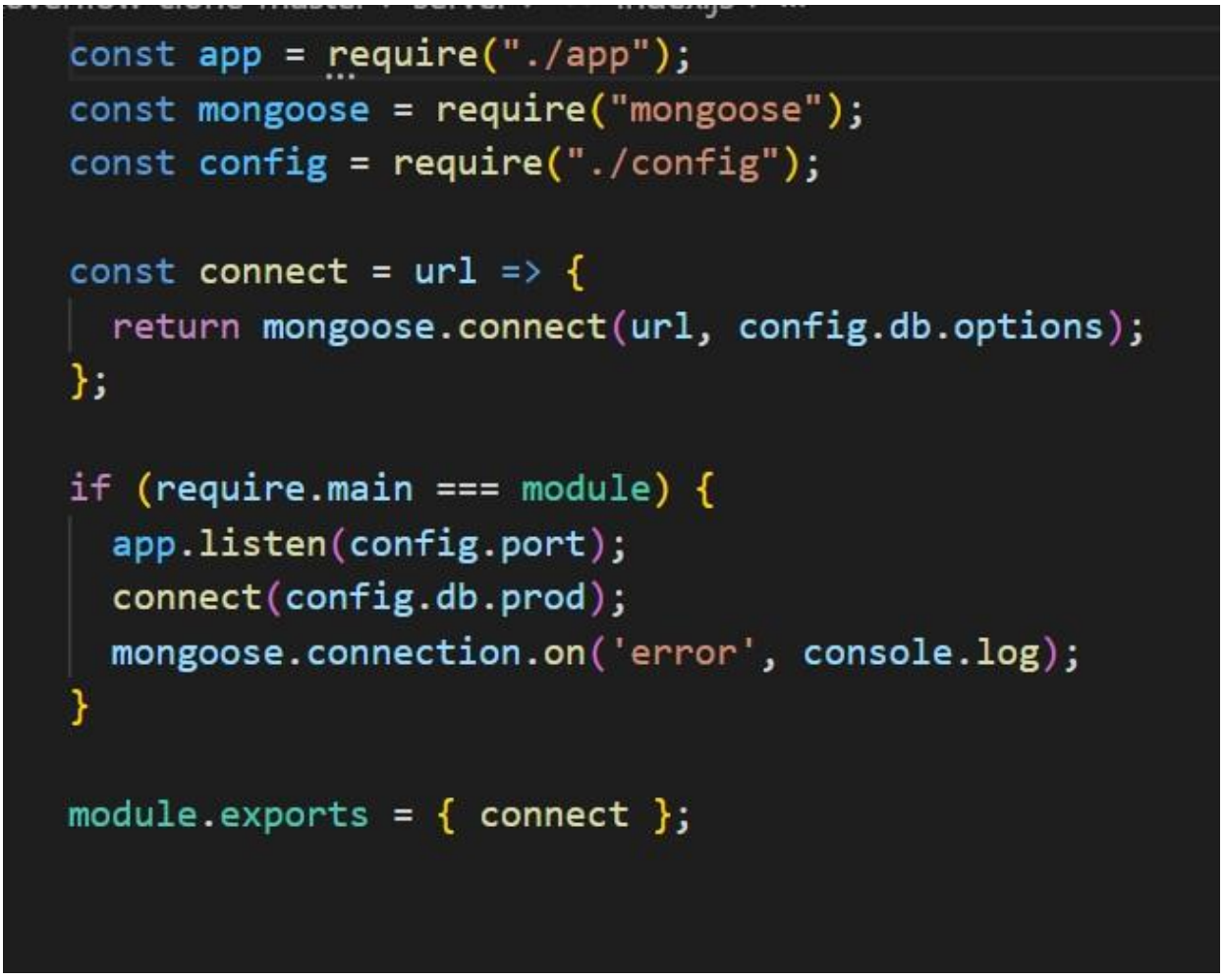
**Fig 4.3 Question Summary(index.js)**

```
exports.removeAnswer = async (req, res, next) => {
  try {
    const { answer } = req.params;
    const question = await req.question.removeAnswer(answer);
    res.json(question);
  } catch (error) {
    next(error);
  }
};

exports.answerValidate = [
  body('text')
    .exists()
    .trim()
    .withMessage('is required')

    .notEmpty()
    .withMessage('cannot be blank')

    .isLength({ min: 30 })
    .withMessage('must be at least 30 characters long')

    .isLength({ max: 30000 })
    .withMessage('must be at most 30000 characters long')
];
```

**Fig 4.4 Answer feature(index.js)**

```
} = require('./controllers/questions');
const {
  loadAnswers,
  answerValidate,
  createAnswer,
  removeAnswer
} = require('./controllers/answers');
const { listPopulerTags, searchTags, listTags } = require('./controllers/tags');
const { upvote, downvote, unvote } = require('./controllers/votes');
const { loadComments, validate, createComment, removeComment } = require('./controllers/comments');

const requireAuth = require('./middlewares/requireAuth');
const questionAuth = require('./middlewares/questionAuth');
const commentAuth = require('./middlewares/commentAuth');
const answerAuth = require('./middlewares/answerAuth');

const router = require('express').Router();

//authentication
router.post('/signup', validateUser, signup);
router.post('/authenticate', validateUser, authenticate);

//users
router.get('/users', listUsers);
router.get('/users/:search', search);
router.get('/user/:username', find);

//questions
router.param('question', loadQuestions);
router.post('/questions', [requireAuth, questionValidate], createQuestion);
router.get('/question/:question', show);
```

**Fig 4.5 Middleware (routes,js)**



```
const app = require("./app");
const mongoose = require("mongoose");
const config = require("./config");

const connect = url => {
  return mongoose.connect(url, config.db.options);
};

if (require.main === module) {
  app.listen(config.port);
  connect(config.db.prod);
  mongoose.connection.on('error', console.log);
}

module.exports = { connect };
```

**Fig 4.6 Database Connection(index.js)**

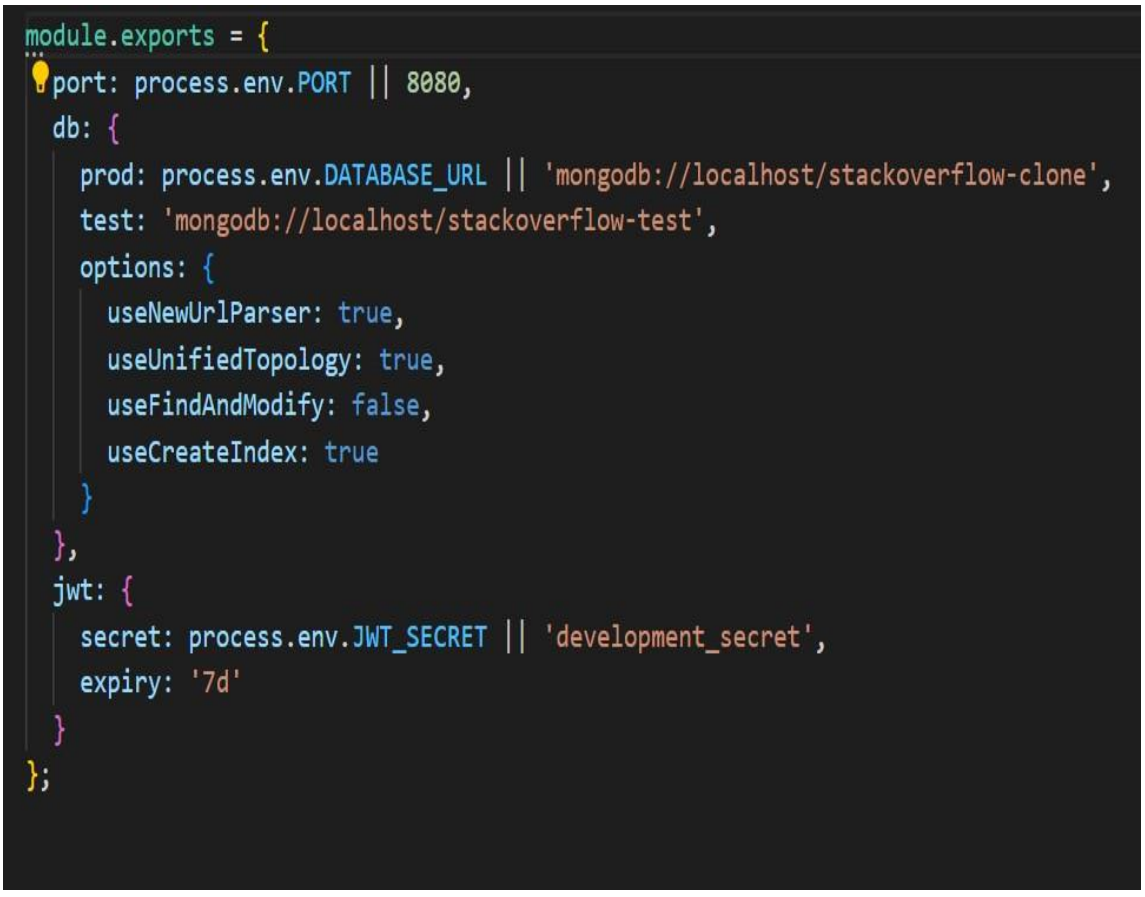



**Fig 4.7 MongoDB Atlas Connection (Users.js)**

# 5. RESULTS

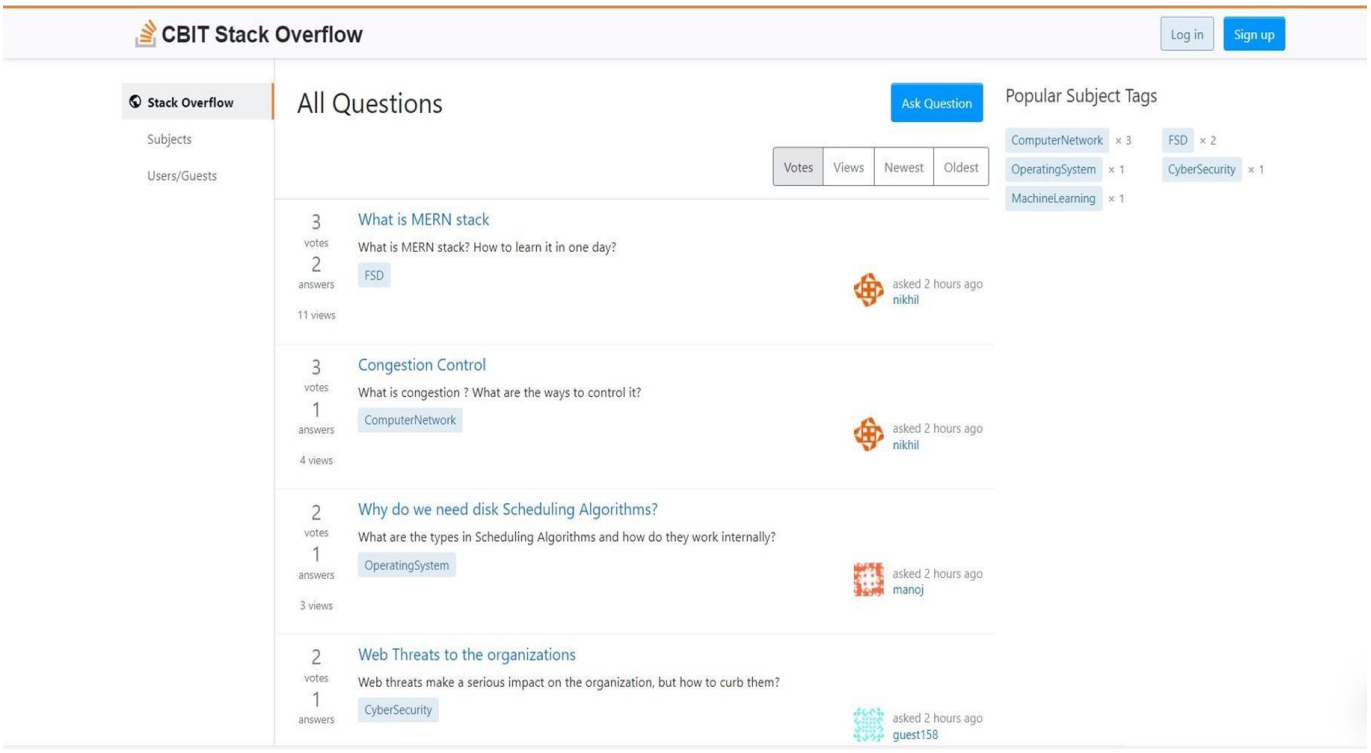



**Fig 5.1 Dashboard Page**

This is the dashboard page of the CBIT stack overflow website before a user logs in into the website.

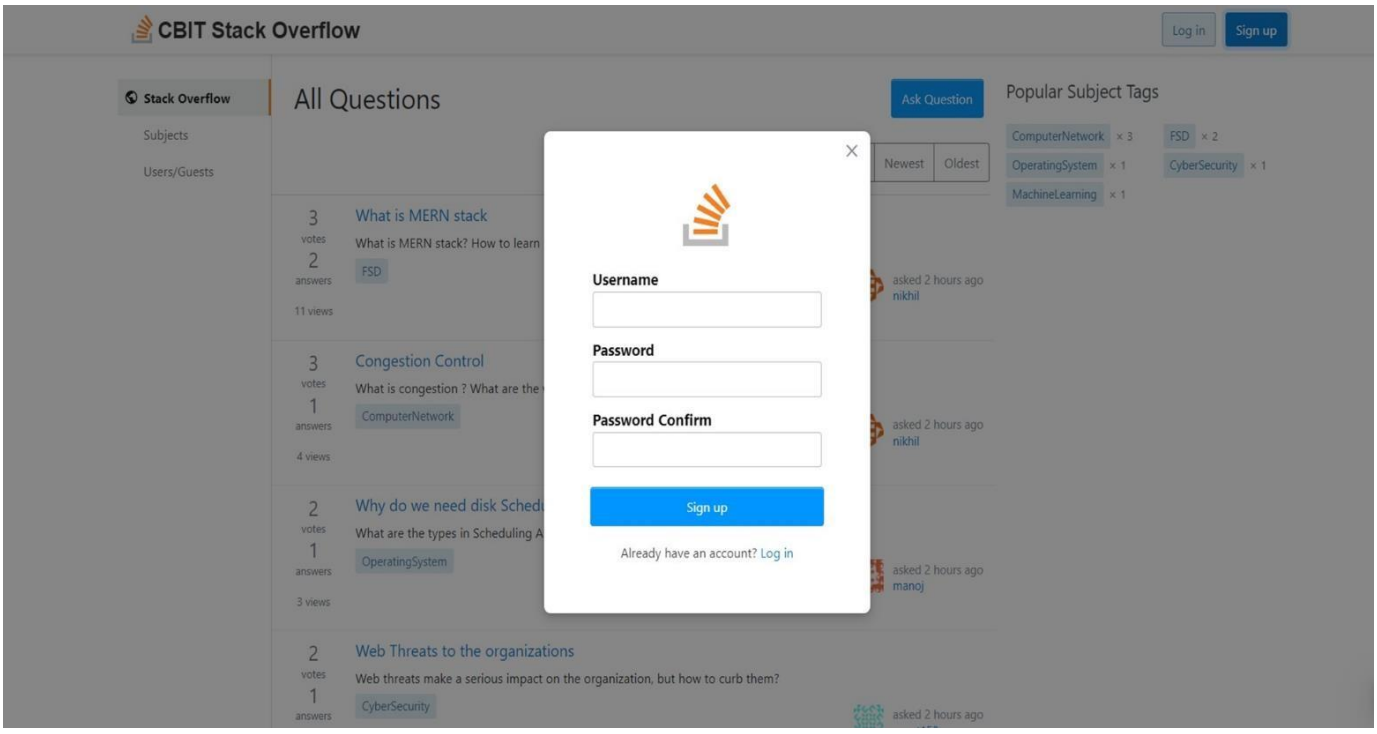



**Fig 5.2 Sign Up page**

If at all the user is new to the website he needs to sign up, by clicking on which the above page appears, where one needs to enter their credentials, and then can login to enter the website.

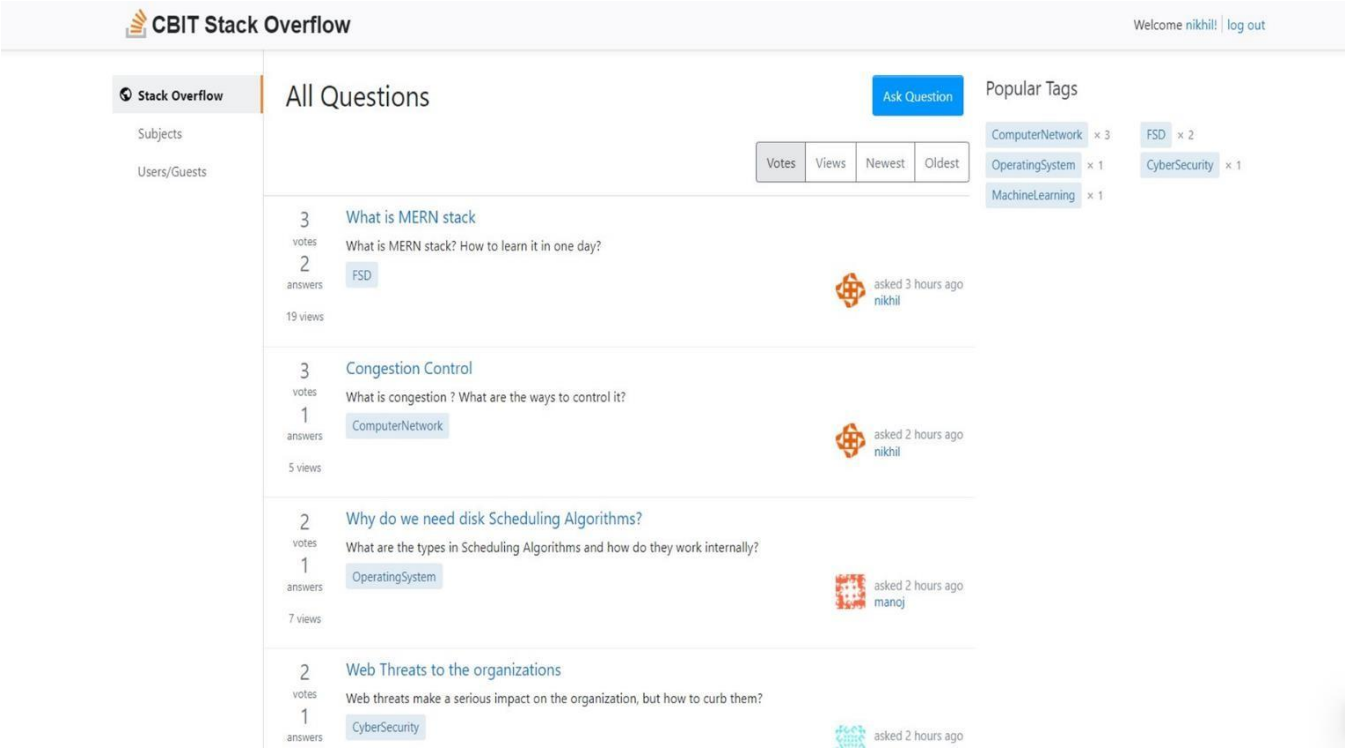



**Fig 5.3 Dashboard page after Logging in**

This is the dashboard page of the CBIT stack overflow website after a user logs in into the website.

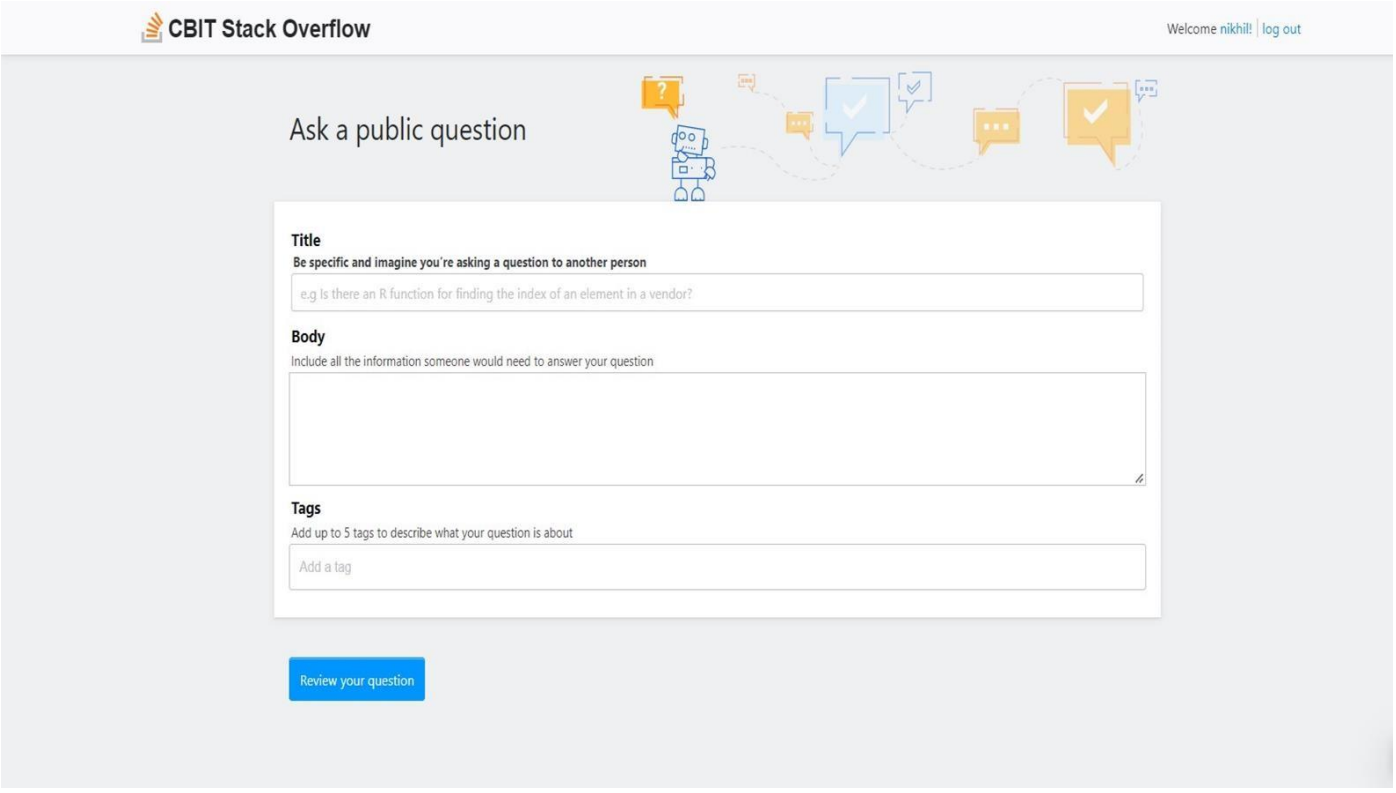



**Fig 5.4 Asking and posting a question**

By clicking on the Ask Question button the above page appears where we need to provide the required details to post a question.

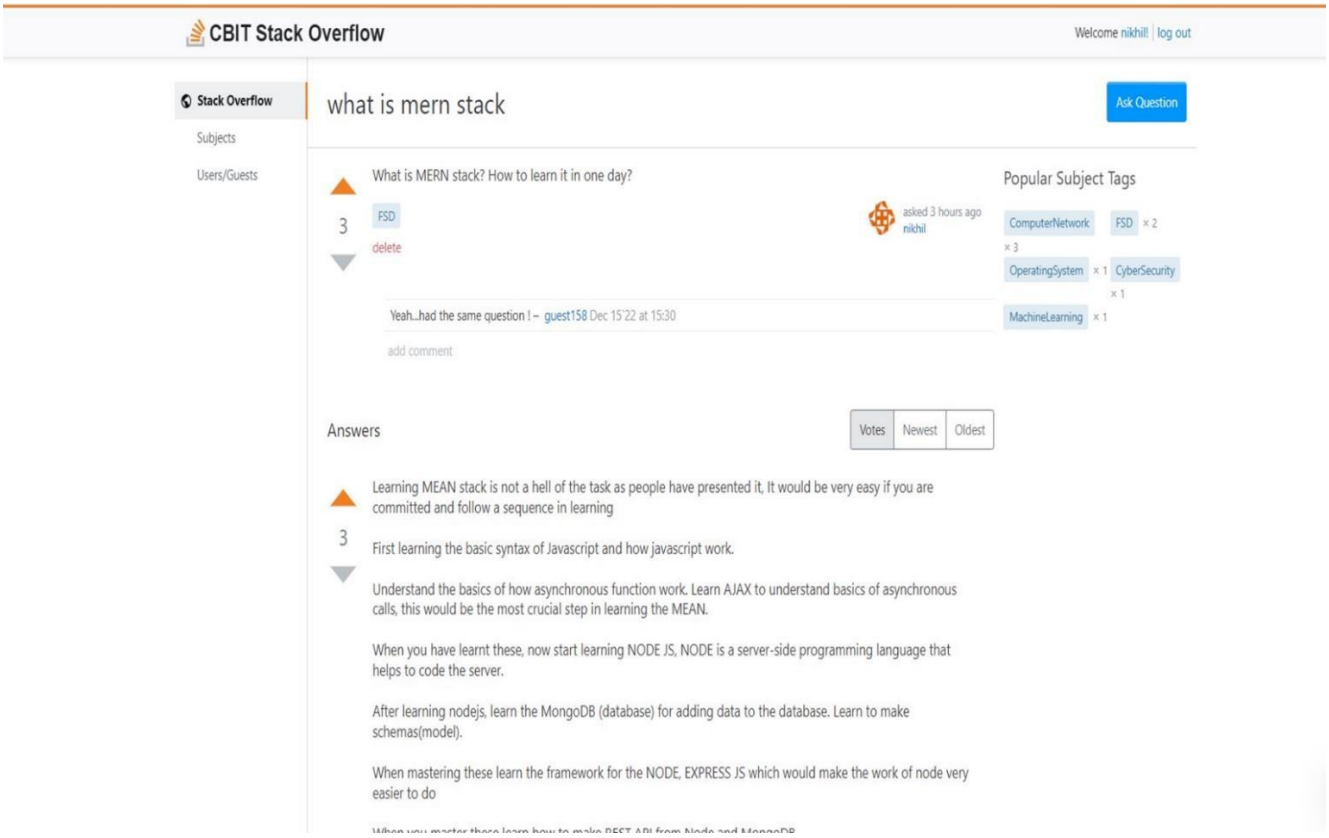



**Fig 5.5 Displaying Question, Comments and Answers**

By clicking on the question , the question posted along with the comments and answers given to it are shown as above.

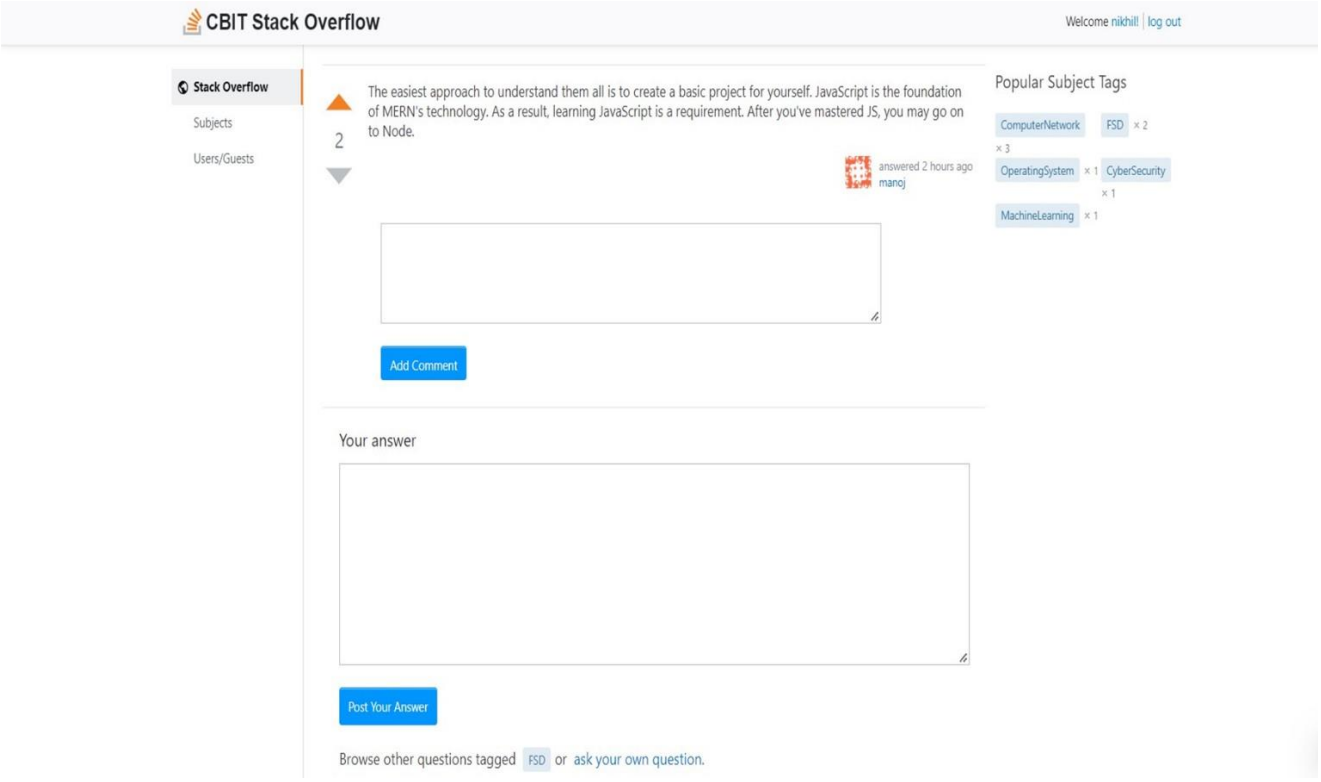



**Fig:5.6 Answering and Commenting a Query**

If at all a user wants to answer or comment a question, he can provide them in the fields given as shown above and post it.

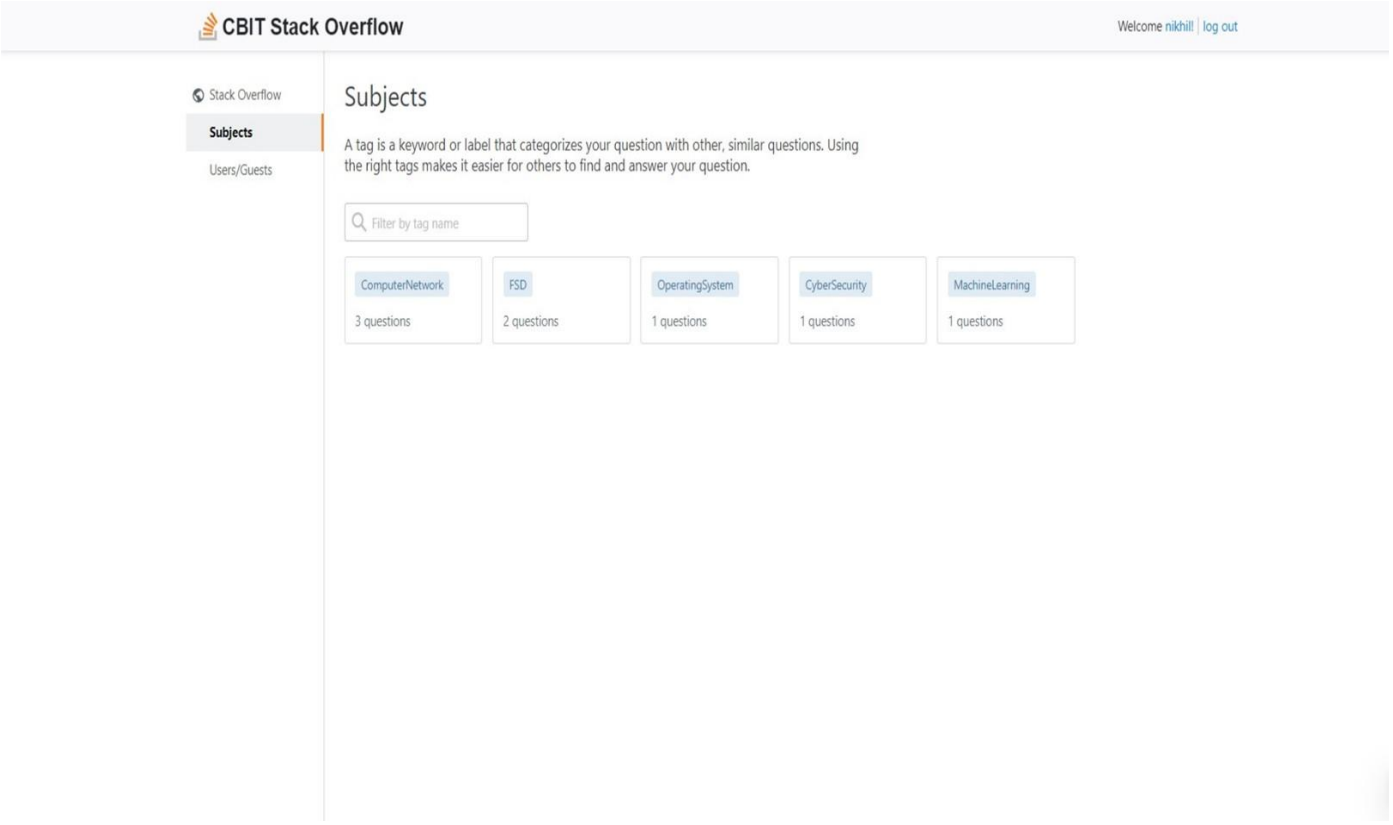



**Fig:5.7 Subjects section displaying subject tags**

This is the subject section which displays the various subject tags used while posting the question.

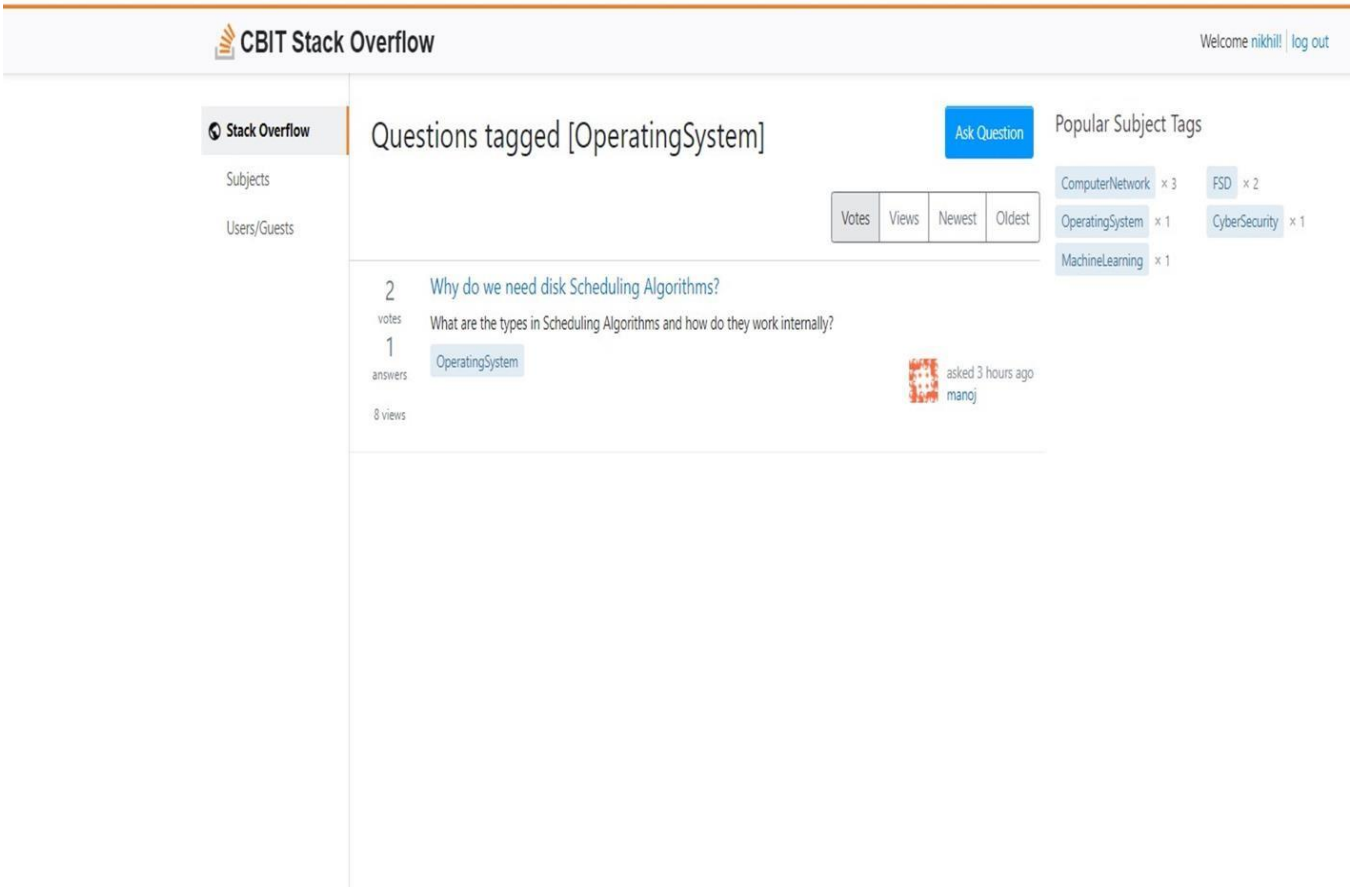



**Fig:5.8 Displaying subject tag related questions**

By clicking on any of the subject tags all questions related to that particular subject appears as shown above.

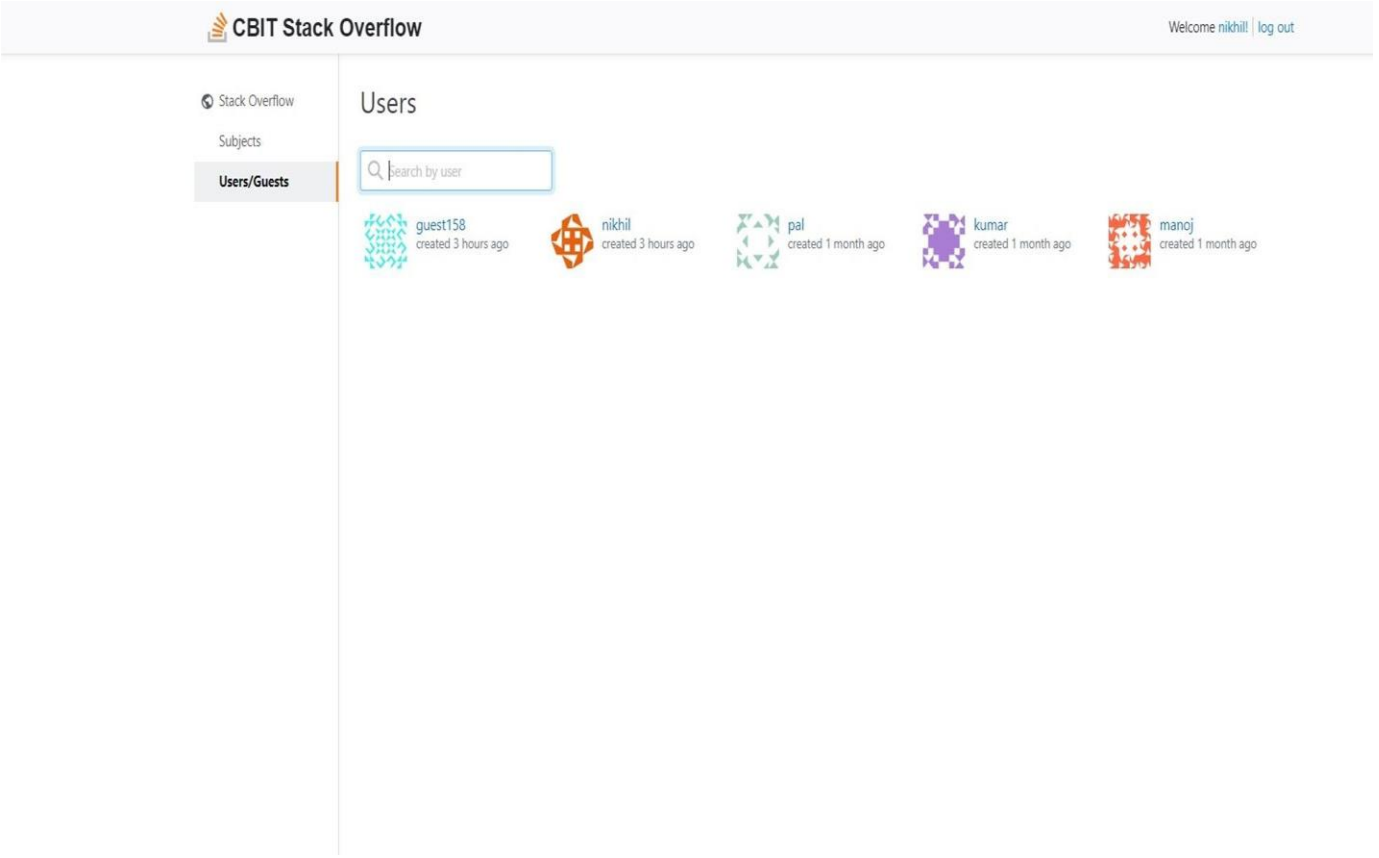



**Fig:5.9 Users section**

This is the Users section which displays all the users who have signed up to the website.

# 6. CONCLUSION

While Stack Overflow has played a massive role in empowering and enabling developers around the world to learn, write code, and build products faster, most of the organizations have not kept up with the evolution of the industry. The key to evolution is the millions of developers from around the world who find the site useful, but who haven't yet been welcomed into the community. We need to expand the reach and engagement to ensure these developers join the conversation and push their own learning to new heights.

In future we would like to use machine learning algorithms to find out the overlapping questions and to detect fake and illogical questions. The website can be made in such a way that there are separate logins for faculty and students.

# BIBILOGRAPHY


[1] https://www.geeksforgeeks.org/how-to-compare-password-and-confirm-password-inputs-using-express-validator/

[2] https://hevodata.com/learn/mongodb-atlas-nodejs/

[3] https://expressjs.com/en/guide/using-middleware.html

[4] https://blog.logrocket.com/how-to-make-http-requests-like-a-pro-with-axios/