DevConnect
Making developer's lives easier one bug at a time

A Project Report
Presented to
The Faculty of the College of
Engineering

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Software Engineering

By

Danesh Vijay Dhamejani
Harshala Ashok Doiphode
Karan Pinakinbhai Jariwala
Yusuf Juzar Soni

December 2022

APPROVED

Professor Dan Harkey, Project Advisor

Professor Dan Harkey, Director, MS Software Engineering

Professor Rod Fatoohi, Department Chair

ABSTRACT

DevConnect: Making developer's lives easier one bug at a time.

By
Danesh Vijay Dhamejani,
Harshala Ashok Doiphode,
Karan Pinakinbhai Jariwala,
Yusuf Juzar Soni

Software development is a collaborative process. Building various modules of real-world software applications requires a large distributed and diverse talent pool composed of different software domain experts. This indicates that software development is a knowledge-intensive endeavor. With the burgeoning of Internet technology, Q&A forums like StackOverflow have made knowledge sharing more transparent and diverse.

The structure and organization of information is the key to extracting knowledge from any data. Q&A forums often make use of domain-specific tags to impose structure to the kind of questions asked. This method, although effective, results in a relatively high turnaround time for the question to be answered. The average time for a query to be answered on StackOverflow is about 3.71 days. Developers that need immediate assistance with a problem might not benefit from such a turnaround time.

Our team proposes to build an application that combines the best of both worlds. Message reply rates in social media platforms are much better than passive Q&A approaches used by knowledge-sharing platforms. Marshaling developers into technology-specific groups supplemented by the existing tag-based system, and various other features that can aid developers could be a viable solution . The proposed application will, therefore, help in making knowledge extraction from user-generated content much easier and user-friendly

Acknowledgments

The authors are deeply indebted to Professor Dan Harkey for his invaluable comments and assistance in participating in this study

Table of Contents

List of Figures

Chapter 1. Project Overview

Introduction to DevConnect

We live in a social media world where you can use LinkedIn to find a job and Facebook to broaden your social circle. However, there is no such thing as a website or place for coders to gather, exchange experiences, and learn from one another. These needs are met by DevConnect, a social site for developers.

In today's quickly changing environment, where technology is getting increasingly complex, developers are expected to widen their perspectives. There is no clear cutoff when it comes to the use of technology. A python developer, for example, would be forced to learn C++ in order to accomplish a certain project for the organization. In this case, you require immediate assistance from a mentor who can guide you through the process of learning various technologies or languages.

Despite the fact that organizations are organized hierarchically, it's likely that your senior developer will be preoccupied with another project and won't be able to provide you with enough direction. In these instances, developers are frequently forced to conduct their own online searches. There are many research materials and tutorials available on the internet, but sifting through them all takes a long time. In this instance, you'll need a website where you may ask your query and have it answered by a professional in the field. Although this is one of the most typical applications for DevConnect, it can also be used in a variety of other circumstances. For example, we live in a world where everyone must multitask. Freelancing is an excellent option for some educated individuals, such as working mothers or others who, for whatever reason, cannot afford to work a fixed-time schedule. These people are continuously on the hunt for freelancing opportunities where they can work on their own time and be paid for it. Through DevConnect, these specialists will be able to demonstrate their knowledge.

We live in a social media world where there is LinkedIn for job seekers and Facebook for expanding your friend circle. However, there is no such thing as a community for coders where they can connect, share their experiences, and learn from one another. DevConnect is a social platform for developers that satisfies these demands.

Proposed Areas of Study and Academic Contribution

Before we implemented our project, we had to understand similar implementations of this project. Crowdsourcing expertise via social media sites like Stack Overflow is one such niche sector that has gained traction in recent years. Stack Overflow is a question-and-answer site aimed largely at software engineers and IT professionals. On the forum, millions of questions have been asked, with twice as many being answered. This shows that the questions and answers are a rich source of information that, when properly arranged, can be put to many different applications. Our project seeks to offer Stack Overflow information in a user-friendly and adaptable style. Extrapolating on the concept of a social media-based Q&A forum, our concept intends to incorporate elements from a variety of other social media applications to make Q&A platforms more user-friendly. As a result, we intend to delve deeply into the aforementioned technologies and contribute to the field of software engineering by making developers' lives easier.

Current State of the Art

There are several existing solutions that we can look at to comprehend the Q&A forum space.

<u>Stack Overflow</u>

Stack Overflow is a community-driven knowledge crowd-sourcing platform that is primarily geared toward software developers and technology enthusiasts. It is a Q&A platform where fellow software engineers can ask technical queries and get their problems solved by the large developer base that the website has. It employs a reputation system that is governed primarily by rewards. Rewards for each contributor is primarily determined by the number of upvotes a post gets. Tags are used in order to lend some structure to the posts. Stack overflow solutions are highly reliable, and research indicates that even direct reuse from the website does not increase the technical debt of the product. This indicates the trust and reliability of the websites.

<u>Yahoo Answers</u>

Yahoo Answers was a knowledge marketplace owned by Yahoo and had features like Stack Overflow. A Platform that grouped questions into categories and subcategories like beauty, health, lifestyle, etc. The platform form was often mocked for its quality of answers. It was discontinued in April 2021. Research indicates that the answers provided on the platform were superficial and extremely broad in its scope. Due to the lack of quality in the answers, it was converted into a read-only platform after which it was discontinued. It serves as a reminder that the lifespan and engagement of a Q&A platform are greatly determined by the content on the platform.

Quora

Quora was founded in 2009 by former Facebook employees. It is a widely used Q&A platform that caters to a variety of topics as opposed to the technology-specific nature of Stack Overflow. People can post any kind of question and answer any of the questions. Quora was dubbed to be a more organized Yahoo Answers, a classier Reddit, and an opinionated Wikipedia. This led to an increase in its popularity. One major drawback of Quora is the lack of moderation in the kinds of questions asked and the kind of answers provided. This leads to questions and answers being vulgar and often unsuitable for younger audiences. Quora is still an active and very popular platform indicating the viability of such products.

Facebook

Social media giant Facebook has revolutionized the way in which we interact with people. The way Facebook organizes people into groups and communities provides us with insights into organizing users in a social media-like application. Additional features such as in-app messaging, the ability to set up a marketplace, and leveraging social media platforms for advertising one's businesses are pathbreaking. Using such a model of segregation and organization of information and users can help improve Q&A platforms.

Chapter 2. Project Architecture

Introduction

The architecture is separated into two parts: the front end and the back end. These components communicate with one another and with external systems, allowing the web application to work smoothly. We'll use React for the entire front end and Node.js for the back end. The front end allows users to interface with the web application, and the front end converts user requests into readable format. When the request is complete, the front end makes the call to the backend. The backend processes the request and makes the required calls to ensure that it is properly responded to. If database access is necessary to process the query, the backend communicates with MongoDB.

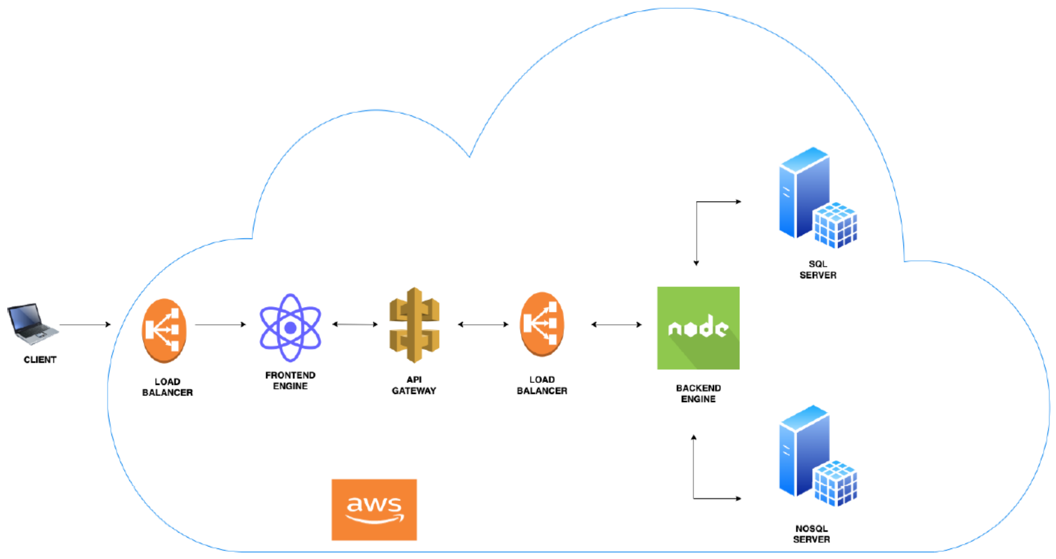


Figure 2.1. *Project Architecture Diagram*

We are planning to deploy the entire system on the AWS cloud with a load balancer. The given diagram depicts the following points:

- The client accesses the web application UI through his system. The UI is built using ReactJS.
- Client makes a request to the server on the front end and the server returns a response that serves the client's request.
- When a request has been made by the client, the API calls are made to the backend NodeJS server. This NodeJS server then queries the SQL/NoSQL databases and returns the desired response to the client.
- Entire web application is hosted on AWS Instances and increasing loads are handled by AWS Load Balancer.
- Load Balancing is done for both the frontend and backend to handle the increasing loads on the respective servers.

Architecture Subsystems

In our project architecture, there are mainly four subsystems. Please find a description of each subsystem below:

Frontend: Because this subsystem connects directly with users, the frontend is the most significant aspect of our web application. We're using React to create the front end. Material UI is used to make our front end more user-friendly and appealing.

Backend: This is the brain of our web application; it will receive requests from the front end and make required calls to other systems in order to offer an acceptable answer to the front end's request.

Load balancer: In our system, we want to employ an AWS load balancer with an auto-scaling rule. We want to utilize a load balancer for both the front end and the back end. The frontend load balancer will deliver requests to a healthy frontend instance, after which the requests will be transferred to the backend load balancer, which will identify a healthy backend instance for request processing.

Database Engine: With MySQL server and MongoDB, we propose to implement SQL and NoSQL databases. These engines take requests from the backend, process them, and then provide a response.

Chapter 3. Technology Descriptions

We set out with a very simple idea to build a product that has a positive impact on the developer community. Often when developers get stuck on a problem, their first instinct is to visit a Q&A forum on the internet to figure out a possible solution. One of the most popular Q&A forums in today's time is Stack Overflow. Although extremely helpful the forum is highly technical and sometimes daunting to new developers. Our goal was to improve the user experience for novice as well as accomplished developers. In order to make the content on Q&A forums more user-friendly and palatable we decided to build a platform that incorporates the best social media has to offer with the best technical help provided on the internet. Most applications today are built on the principles of Distributed Software Systems and possess a three-layered architecture. It was a surprise to us that Stack Overflow is actually built in a monolithic fashion. The current architecture is shown in the Figure below.

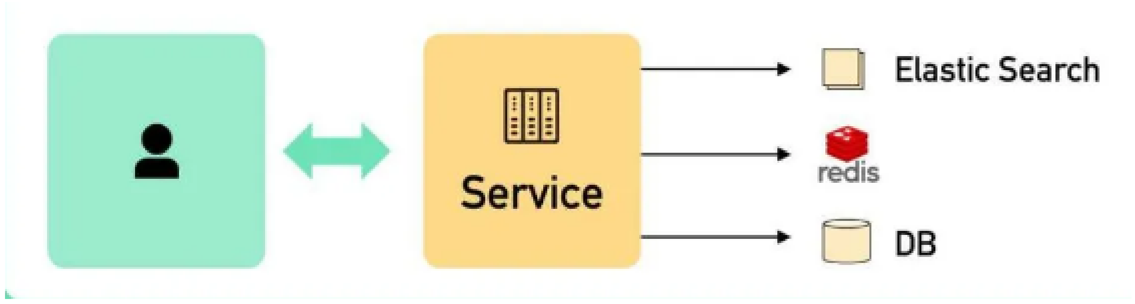
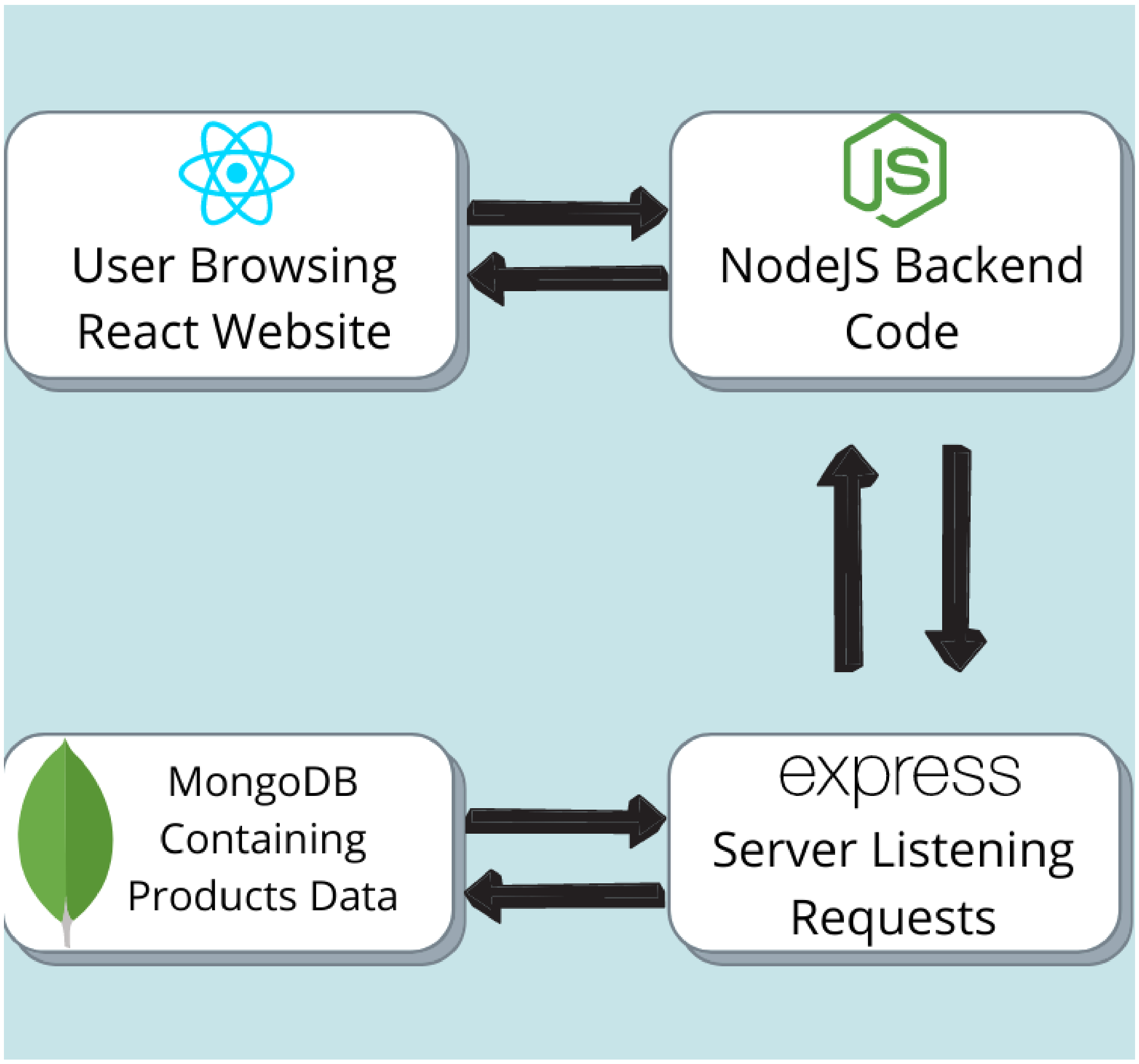


Figure 3.1 *Stack Overflow Architecture*

Since our idea involved merging two different concepts together, We had to consider the challenges that came with implementing social media elements we planned on incorporating into our project. Some of the challenges we faced included factoring in scalability, reliability, data distribution, etc. We decided to use the MERN stack to implement the project. A typical MERN application would look like this.



User Browsing React Website
NodeJS Backend Code
MongoDB Containing Products Data
express
Server Listening Requests

Figure 3.2 *MERN Application Architecture*

Why did we choose the MERN Stack?

MERN stack automatically provides logical segregation of components into Frontend, Middleware, and Backend components. Moreover, the MERN stack is optimized for the web, which is the primary ingress point for our application. The technologies involved in the MERN stack are discussed in the sections that follow.

Client Technologies

The single-page application (SPA), which necessitates having a web application viewed from a single web page, is the foundation of React.js. This dramatically streamlines the user experience by preventing the need to load a new page for every activity. Facebook is a one-page application, which is obviously seen in the ease and comfort with which the social network may be used. This kind of smooth and streamlined user experience is crucial for any social media-based website to stay competitive. React is also widely used, and has excellent support to build in react native, which would help our application scale even in the mobile application space without a lot of rework.

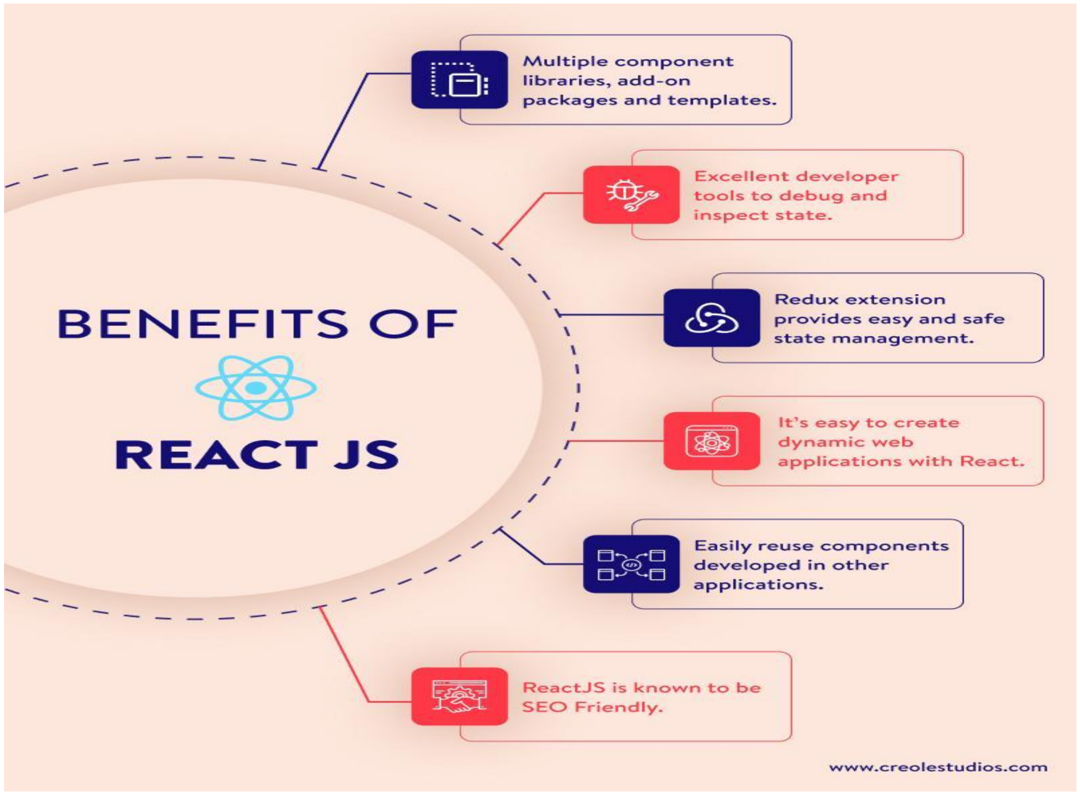

BENEFITS OF
REACT JS
Multiple component libraries, add-on packages and templates.
Excellent developer tools to debug and inspect state.
Redux extension provides easy and safe state management.
It's easy to create dynamic web applications with React.
Easily reuse components developed in other applications.
ReactJS is known to be SEO Friendly.
www.creolestudios.com

Figure 3.3 *Benefits of React as a Client Side Technology*

Middle-Tier Technologies

Node JS is a platform that operates on the Single Threaded Event Loop model, inspired by the javascript callback mechanism. This capability allows Node JS to handle more concurrent client requests with ease.

The single-threaded event loop has the following advantages over the multi-threaded request/response stateless model:

- Can easily manage the requests of an increasing number of concurrent clients.
- Removes the need to continually create new threads due to the Event loop.
- The smallest number of threads are used by applications created with node.js to conserve memory or other resources.

Node JS combined with the REST API architecture helps the application to perform much better over the web greatly improving developer experience too. Applications may be connected to one another using the REST API like jigsaw pieces. The REST APIs are extremely useful for usage in the MERN stack since they are based on HTTP and emulate online communication techniques. REST APIs give developers the opportunity to create different components in the front end that can quickly connect to the same system.

Data-Tier Technologies

We decided to use a nonrelational database to store the data required for our project. MongoDB is a schemaless nonrelational database that offers a lot of flexibility when the shape of the data to be stored and the requirements of the data are dynamic. MongoDB's flexible query system makes retrieval of data pretty easy without the need to perform complicated joins on data tables. Since the data in a social media-based Q&A

platform is dynamic and might need modification and easy querying in the near future, MongoDB seemed to be a good choice to store our data. Aggregation. flexibility and the ability to design schemaless data models that can be easily updated without major rework prompted us to select this technology.

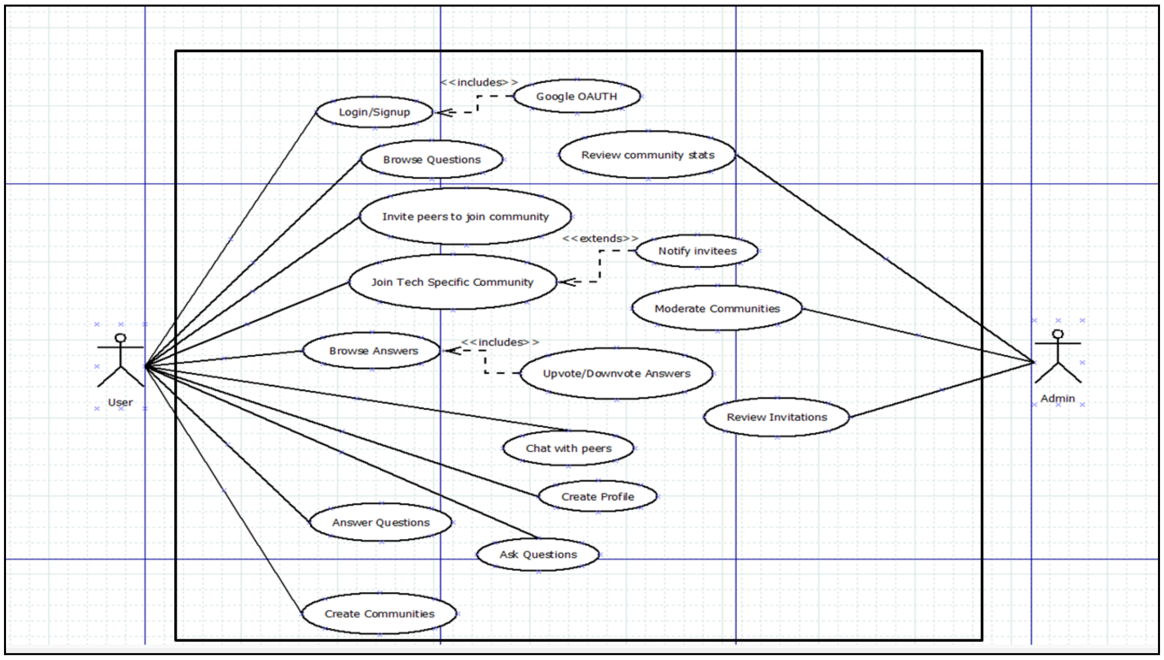
.

Chapter 4. DevConnect Design

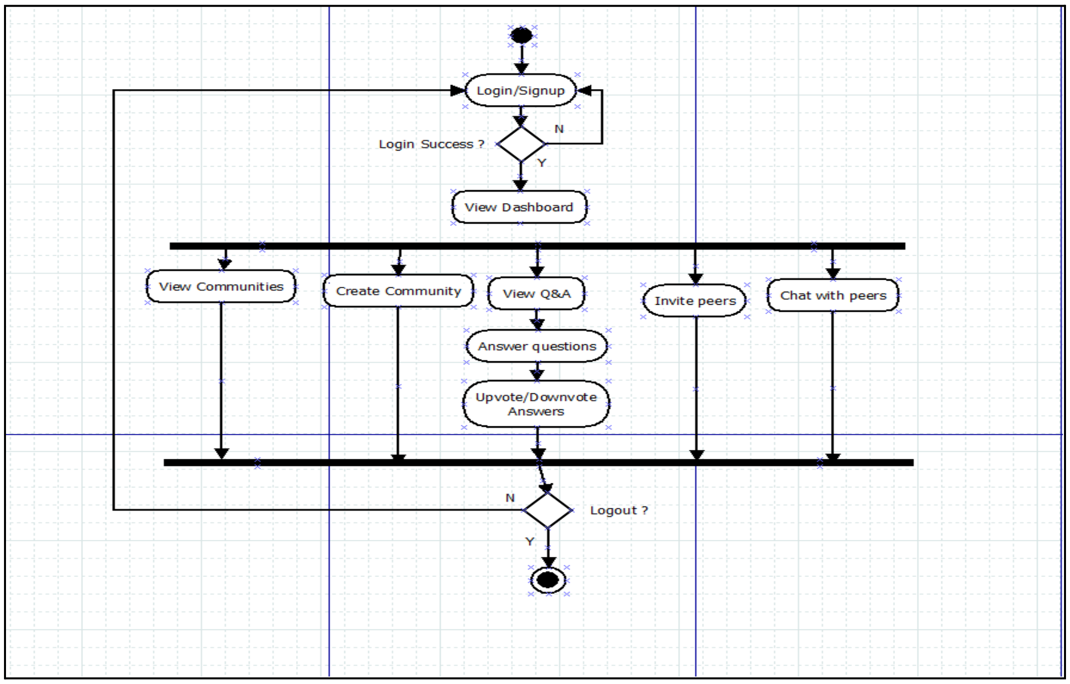
Client Design

Client-side UML Diagrams are mentioned in the below section.

The following UML Diagram for the overall DevConnect design is as follows:

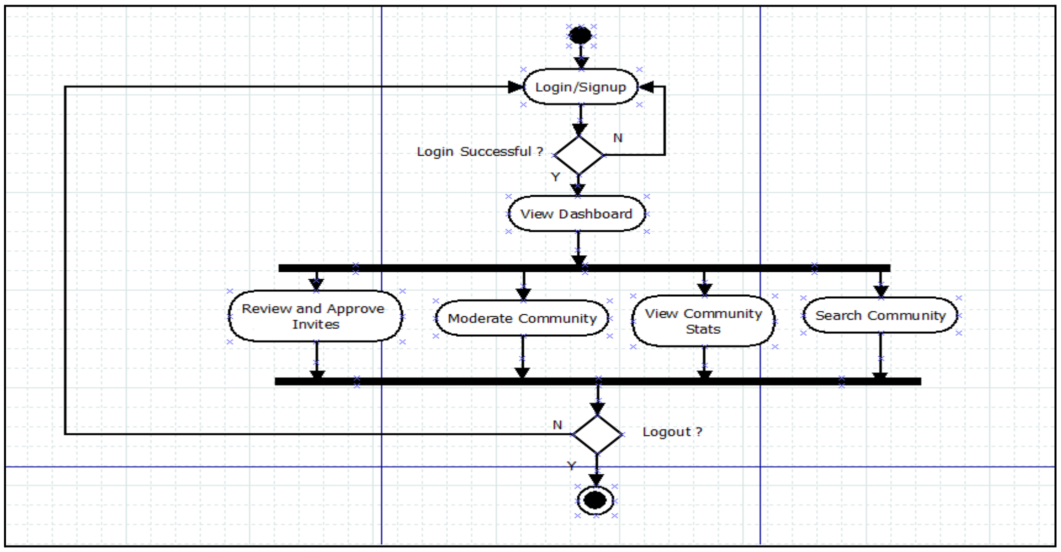


UML Diagrams when shown functionality-vise are as follows:



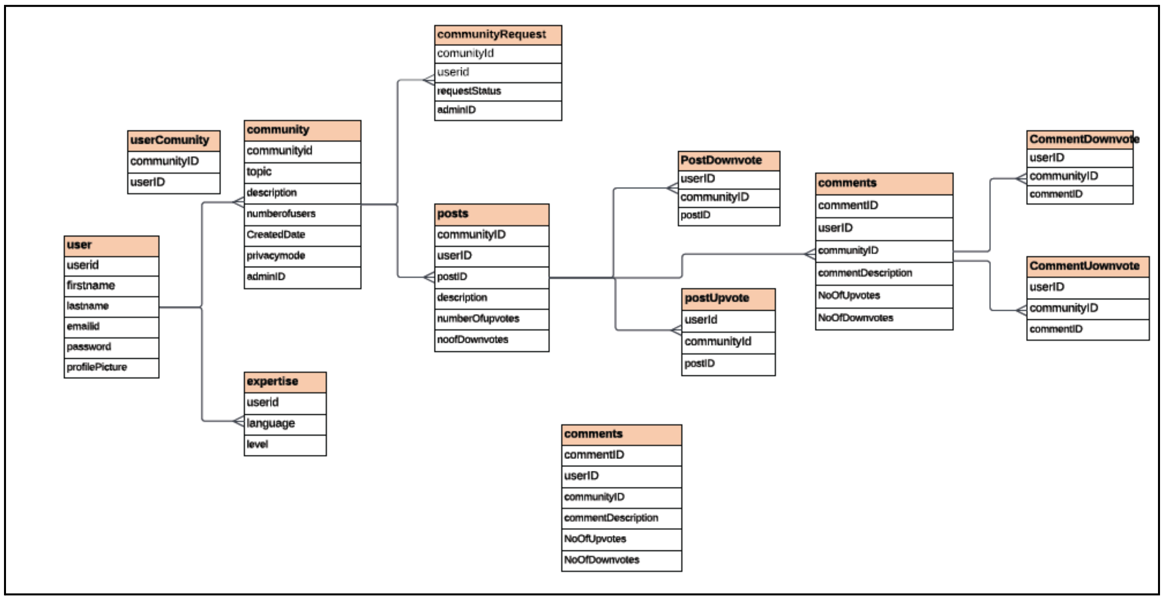
Middle-Tier Design

UML Diagram for middle tier functions are as follows:



Data-Tier Design

The Entity- Relationship Diagram for Database design for DevConnect is highlighted below.



Chapter 5. Project Implementation

We implemented our project keeping in mind the principles of modularity and reusability in mind. Let us understand the layout of the project using the developer communities dashboard in our minds.

Client Implementation

Our application consists of pages that define our application and components that are used to build those pages. Attached below are code snippets that provide some idea of how we implemented our front end. In our example, we have a Community Dashboard page as shown below.

```
 return (
    <div>
      <TopBar />
      <div className="CommunityDashboard">
        <SideBar />
        <div className="CommunityDashboard_Right">
          <h4 style={{ float: "left", marginLeft: "20px" }}>
            <ExploreTwoToneIcon /> &nbsp;Explore Developer
Communities
          </h4>
          <Button
            className="CreateNew"
            variant="outline-dark"
            size="sm"
            onClick={handleShow}
          >
            <AddCircleTwoToneIcon /> &nbsp;Create Community
          </Button>
          <div className="modal-container">
            <Modal show={show} onHide={handleClose}>
              <Modal.Header
                closeButton
                style={{ background: "black", color: "white" }}
              >
                <Diversity3OutlinedIcon />
                &nbsp;
                <b>Create your Community !! </b>
              </Modal.Header>
              <Modal.Body>
                <CreateCommunity />
              </Modal.Body>
            </Modal>
          </div>
          {communitiesData.map((community) => (
           _ <CommunityTile variant="outline-dark"
key={community._id} community={community}/>
            ))}

        </div>
        <RightBar profile={"profile"} />
      </div>
    </div>
  );
}
```

Figure 5.1 *CommunityDashboard Page*

The underlined text indicates one such component that is used to render parts of the page.

Let us take a look at the Community Tile component:

```
return (
    <div>
      <div className="CardContainer">
        <Container>
```

```
          <Card>
            <Card.Header>
              <i>
                <div className="CommunityHeader">
                  <div className="middleInfo">
                    <GroupOutlinedIcon
                      sx={{
                        color: "#808080",
                        fontSize: "1.5rem",
                      }}
                    />
                    <span className="middleInfoText">
                      {community.numberOfMembers}
                    </span>
                  </div>
                  <div className="middleInfo">
                    <QuickreplyTwoToneIcon
                      sx={{
                        color: "#808080",
                        fontSize: "1.5rem",
                      }}
                    />
                    <span className="middleInfoText">
                      {community.numberOfPosts}
                    </span>
                  </div>
                  <div className="middleInfo">
                    <AccountCircleTwoToneIcon
                      sx={{
                        color: "#808080",
                        fontSize: "1.5rem",
                      }}
                    />
                    <span className="middleInfoText">
                      Creator: {community.createdBy}
                    </span>
                  </div>
                </div>
```

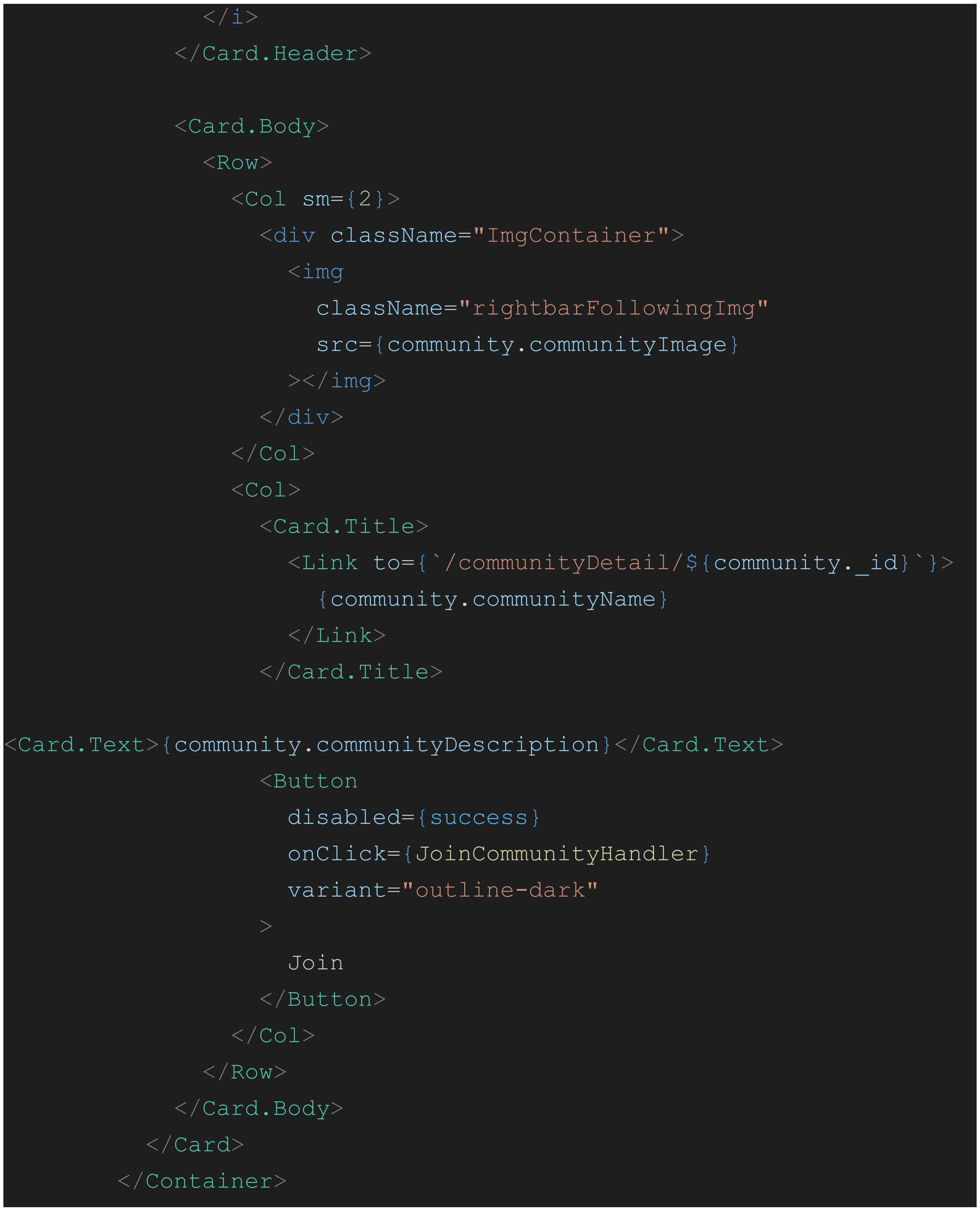
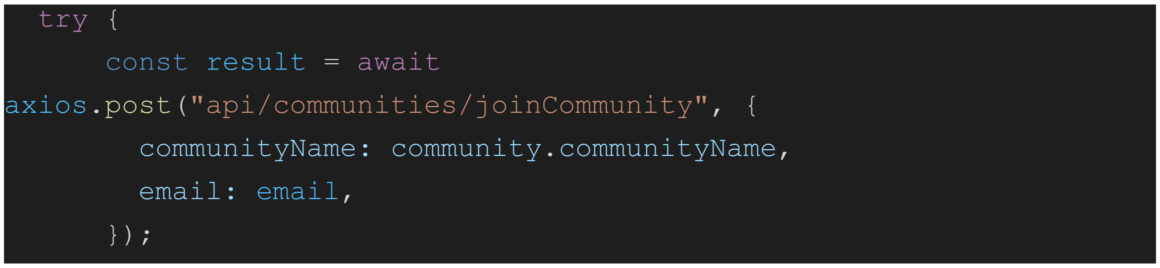

```
              </i>
            </Card.Header>

            <Card.Body>
              <Row>
                <Col sm={2}>
                  <div className="ImgContainer">
                    <img
                      className="rightbarFollowingImg"
                      src={community.communityImage}
                    ></img>
                  </div>
                </Col>
                <Col>
                  <Card.Title>
                    <Link to={`/communityDetail/${community._id}`}>
                      {community.communityName}
                    </Link>
                  </Card.Title>

<Card.Text>{community.communityDescription}</Card.Text>
                  <Button
                    disabled={success}
                    onClick={JoinCommunityHandler}
                    variant="outline-dark"
                  >
                    Join
                  </Button>
                </Col>
              </Row>
            </Card.Body>
          </Card>
        </Container>
```

Figure 5.2 *Community Tile Component*

Middle-Tier Implementation

After having built the skeleton, it is important to add functionality to these components. REST APIs are the workers that help us make simple frontend components interactive. Imagine a car chassis without engine wheels and seats. The car might look complete from the outside but is of no use to a person. APIs are the parts that help in performing various tasks with the application. Let us consider the joint community action. Let us assume a developer wants to join a particular community. This action is accomplished via an API. An API has two major components, a call from the front end to our middleware, and a call from the middleware to the database. One complete cycle helps the action to complete.

Let us take a look at the call made from the front end:



```
  try {
      const result = await
axios.post("api/communities/joinCommunity", {
        communityName: community.communityName,
        email: email,
      });
```

Figure 5.3 *Frontend call to joinCommunityAPI*

This request is then matched with a corresponding API in the backend that fetches the actual data to be displayed.

The call to the database would look like this:

```
router.post("/joinCommunity", async (req, res) => {
  console.log(req.body.email)
  console.log(req.body.communityName)
  try {
    communityMembers = await Community.find(
      { communityName: req.body.communityName },
      { communityMembers: 1 }
    );
    let members = communityMembers[0].communityMembers;
    if (members.includes(req.body.email)) {
      res.status(409).json({ message: "User is already part of
Community" });
    } else {
           const check1 = await Community.findOneAndUpdate(
        { communityName: req.body.communityName },
        {
          $push: {
            communityMembers: req.body.email,
          },
          $inc: { numberOfMembers: +1 },
        }
      );
      console.log("check 1 complete")
      const check2 = await User.findOneAndUpdate(
        { email: req.body.email },
        { $push: { followers: req.body.communityName } }
      );
      console.log("check2 complete")

      if (check1 && check2) {
        res.status(200).json({ message: "Successfully joined
Community" });
      }
    }
  } catch (err) {
    res.status(500).json({ message: err });
  }
});
```

Figure 5.4 *joinCommunityAPI that fetches data from the database.*

Data-Tier Implementation

The final tier involves the actual data. The middleware layer interacts with the database to fetch the data the user has asked for. MongoDB being a schemaless design helps us organize data using a JSON object-like format. This so-called model can be very easily modified to incorporate changes as and when required. Let us take the Community model responsible for defining records in the database.

```
const communitySchema = new mongoose.Schema({
  communityId: {
    type: mongoose.Types.ObjectId,
    auto: true,
  },
  communityName: {
    type: String,
    required: true,
  },
  communityDescription: {
    type: String,
    required: true,
  },
  communityRules: {
    type: Array,
    required: true,
  },
  communityImage: {
    type: String,
    default:

"https://www.china-admissions.com/wp-content/uploads/2021/06/Divi-Co
mmunity-Update-May-2020-scaled-1.jpeg",
  },
  creationTime: {
    type: Date,
    default: Date.now,
  },
  createdBy: {
    type: String,
    required: true,
  },
  communityMembers: {
    type: Array,
  },
  numberOfMembers: {
    type: Number,
    default: 1,
  },
  numberOfPosts: {
    type: Number,
    default: 0,
  },
})
```

Figure 5.5 *Community data model*

Chapter 6. DevConnect Implementation Screenshot

In this section we have the screenshot of the DevConnect web application. We have been able to deliver all the functionalities that we thought to implement in our application from the start. So let's go through the images functionality-wise.

Landing Page:

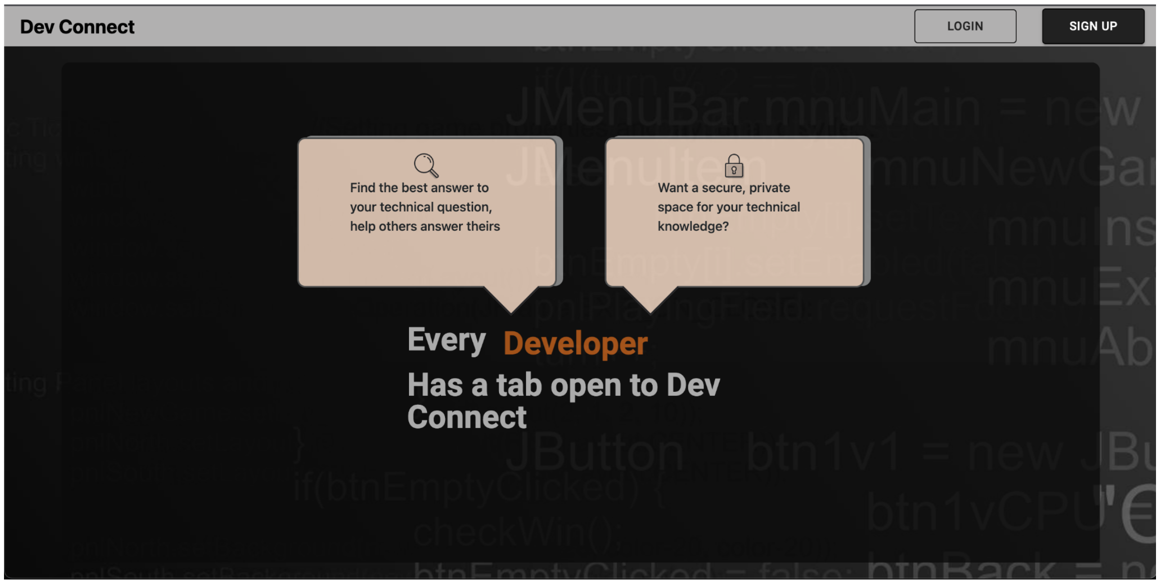

Dev Connect
LOGIN
SIGN UP
Find the best answer to your technical question, help others answer theirs
Want a secure, private space for your technical knowledge?
Every Developer
Has a tab open to Dev Connect

Figure 6.1 *Landing Page*

This is the page that a user will see when they first time visit the website. It will have brief information about the application and a Login and Signup Button.

Signup Page: Users can signup for the web-app through this page by providing a few details and setting up password.

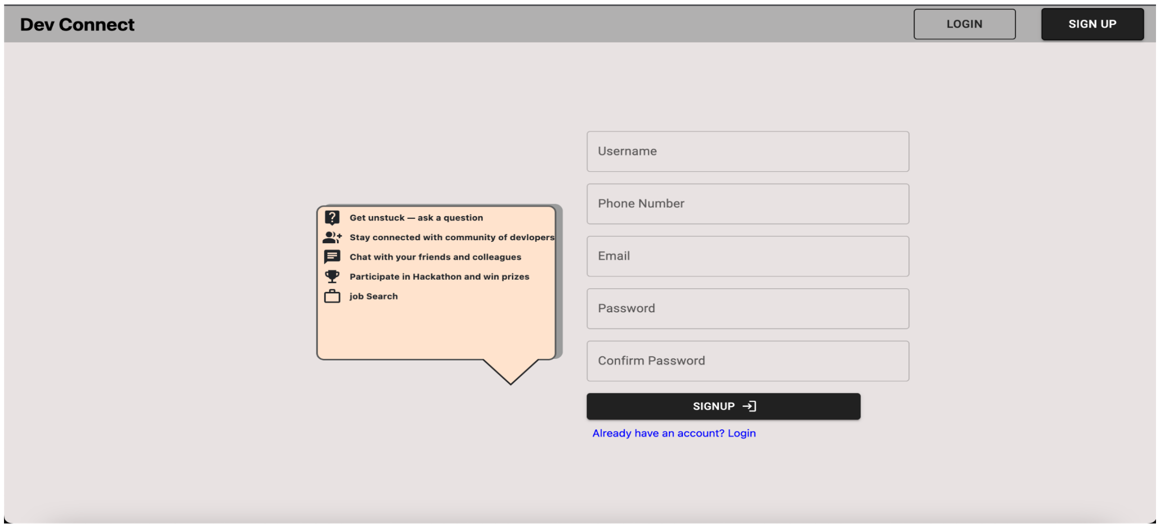
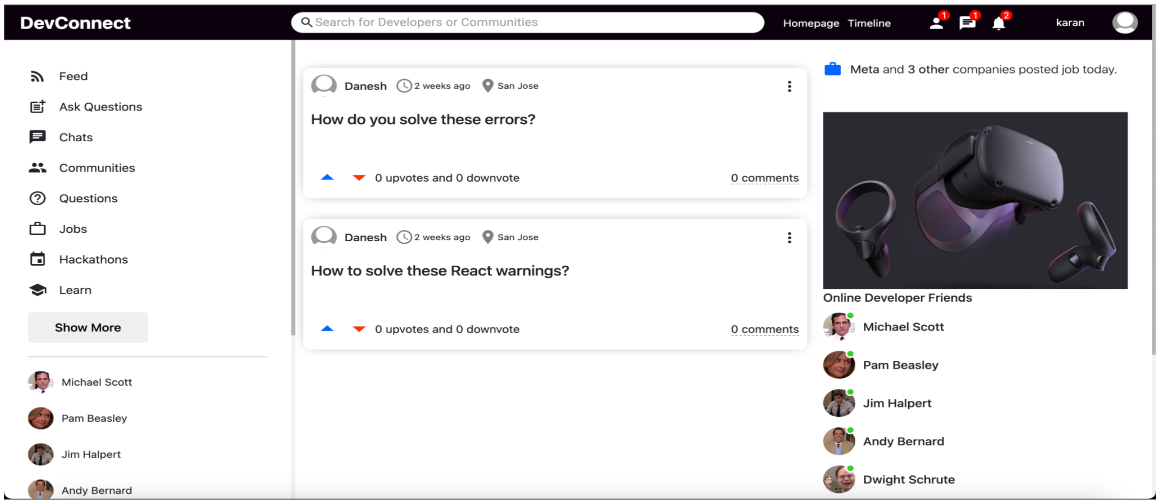

Dev Connect
LOGIN
SIGN UP
Get unstuck — ask a question
Stay connected with community of developers
Chat with your friends and colleagues
Participate in Hackathon and win prizes
job Search
Username
Phone Number
Email
Password
Confirm Password
SIGNUP
Already have an account? Login

Figure 6.2 *Signup Page*

Home Page: It is the homepage of the applications where users can see the latest posts and navigate to different sections of the application.



Danesh 2 weeks ago San Jose

How do you solve these errors?

0 upvotes and 0 downvote 0 comments

Danesh 2 weeks ago San Jose

How to solve these React warnings?

0 upvotes and 0 downvote 0 comments

Meta and 3 other companies posted job today.

Online Developer Friends

Michael Scott

Pam Beasley

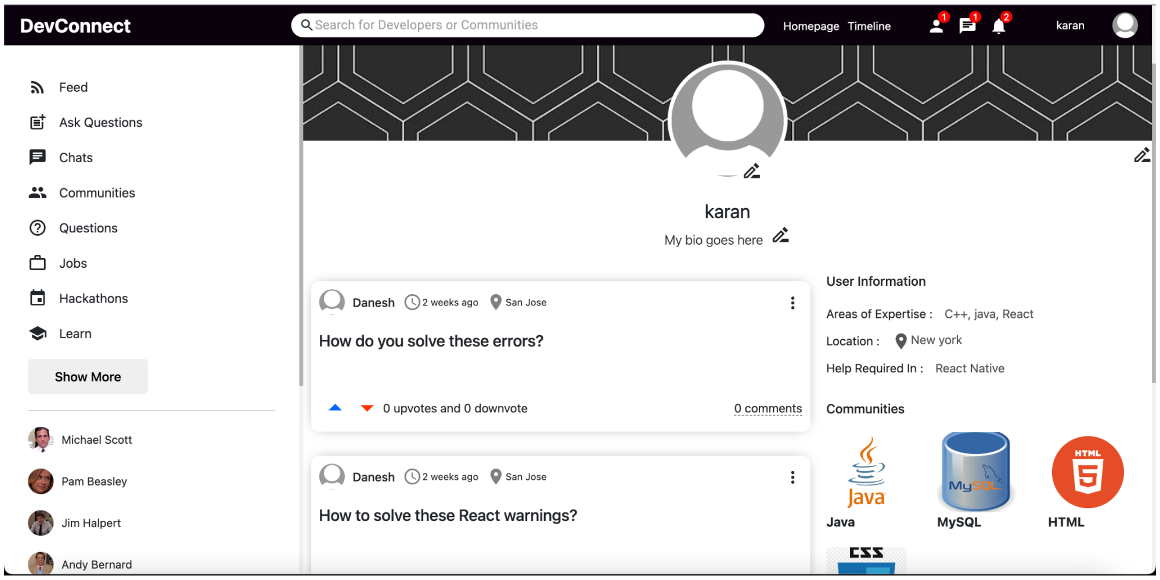
Jim Halpert

Andy Bernard

Dwight Schrute

Figure 6.3 *Home Page*

Profile Page: Users can see their and their friend's profiles.



karan

My bio goes here

Danesh 2 weeks ago San Jose

How do you solve these errors?

0 upvotes and 0 downvote 0 comments

Danesh 2 weeks ago San Jose

How to solve these React warnings?

User Information

Areas of Expertise : C++, java, React

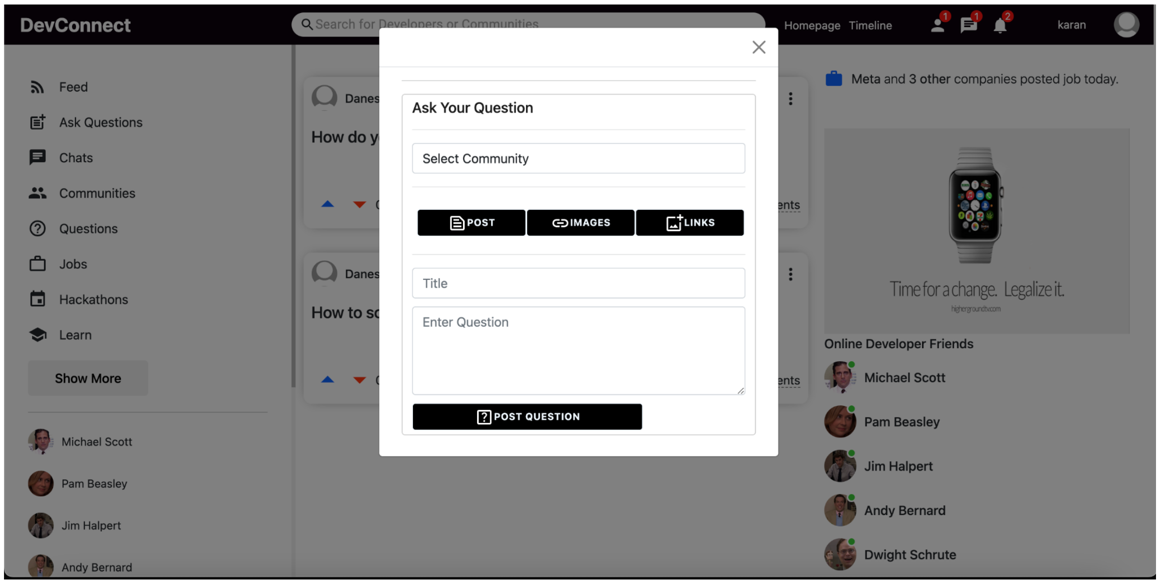
Location : New york

Help Required In : React Native

Communities

Figure 6.4 *Profile Page*

Add Post: Users can add posts with questions and images to get answered by other.



Meta and 3 other companies posted job today.

Ask Your Question

Select Community

POST IMAGES LINKS

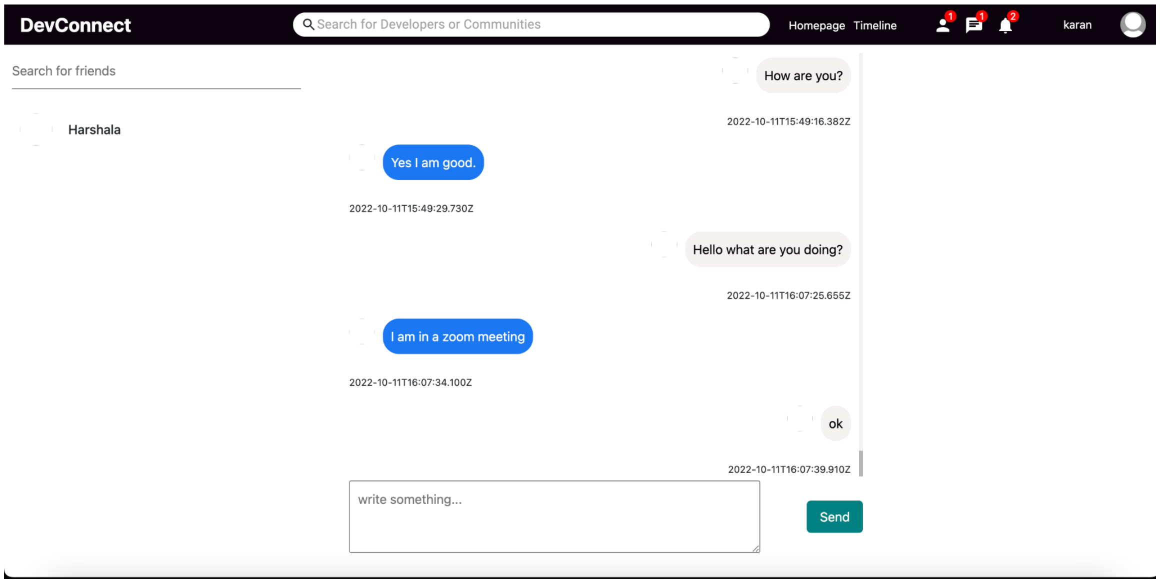
Title

Enter Question

POST QUESTION

Figure 6.5 *Add Post Page*

Messaging Friends: Users can live chat with their friends/peers to get instant replies.


DevConnect

Search for Developers or Communities

Homepage Timeline karan

Search for friends

Harshala

How are you?
2022-10-11T15:49:16.382Z

Yes I am good.
2022-10-11T15:49:29.730Z

Hello what are you doing?
2022-10-11T16:07:25.655Z

I am in a zoom meeting
2022-10-11T16:07:34.100Z

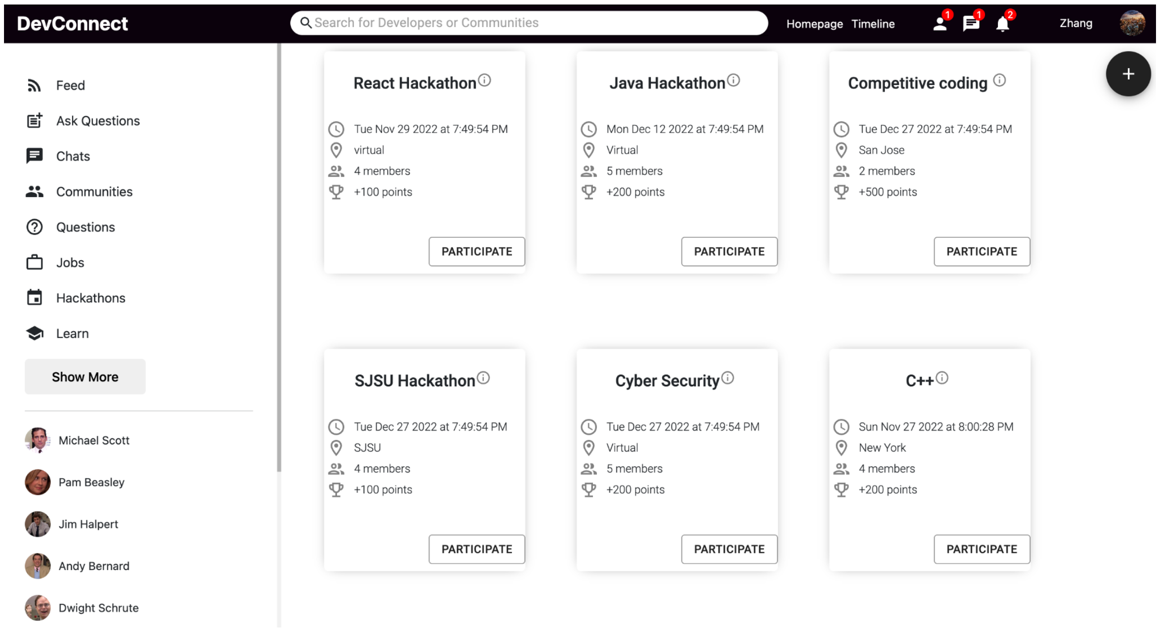
ok
2022-10-11T16:07:39.910Z

write something...

Send

Figure 6.6 *Message Page*

Hackathon Page: Users can see and participate in various listed Hackathons.


DevConnect

Search for Developers or Communities

Homepage Timeline Zhang

Feed
Ask Questions
Chats
Communities
Questions
Jobs
Hackathons
Learn
Show More

Michael Scott
Pam Beasley
Jim Halpert
Andy Bernard
Dwight Schrute

React Hackathon
Tue Nov 29 2022 at 7:49:54 PM
virtual
4 members
+100 points
PARTICIPATE

Java Hackathon
Mon Dec 12 2022 at 7:49:54 PM
Virtual
5 members
+200 points
PARTICIPATE

Competitive coding
Tue Dec 27 2022 at 7:49:54 PM
San Jose
2 members
+500 points
PARTICIPATE

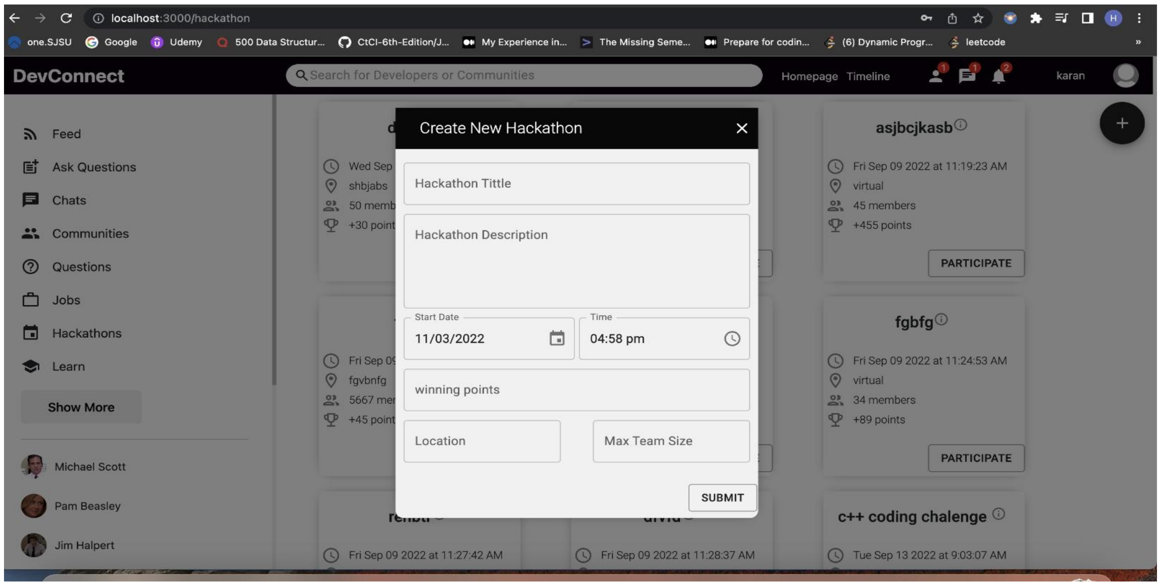
SJSU Hackathon
Tue Dec 27 2022 at 7:49:54 PM
SJSU
4 members
+100 points
PARTICIPATE

Cyber Security
Tue Dec 27 2022 at 7:49:54 PM
Virtual
5 members
+200 points
PARTICIPATE

C++
Sun Nov 27 2022 at 8:00:28 PM
New York
4 members
+200 points
PARTICIPATE

Figure 6.7 *Hackathon Page*

Add Hackathon Event: Users can create their own hackathon and post it.



Create New Hackathon

Hackathon Tittle

Hackathon Description

Start Date: 11/03/2022

Time: 04:58 pm

winning points

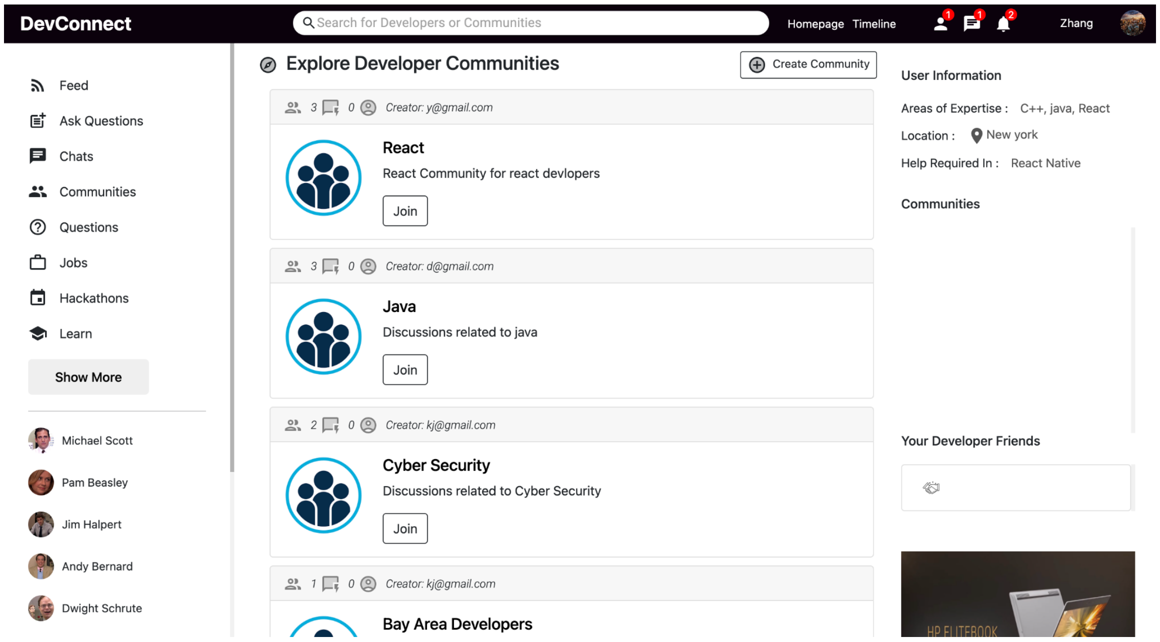
Location

Max Team Size

SUBMIT

Figure 6.8 *Add Hackathon Page*

Communities Page: Users can join and create various communities and get updates.


DevConnect

Search for Developers or Communities

Homepage Timeline

Zhang

Feed

Ask Questions

Chats

Communities

Questions

Jobs

Hackathons

Learn

Show More

Michael Scott

Pam Beasley

Jim Halpert

Andy Bernard

Dwight Schrute

Explore Developer Communities

Create Community

3 0 Creator: y@gmail.com

React

React Community for react devlopers

Join

3 0 Creator: d@gmail.com

Java

Discussions related to java

Join

2 0 Creator: kj@gmail.com

Cyber Security

Discussions related to Cyber Security

Join

1 0 Creator: kj@gmail.com

Bay Area Developers

User Information

Areas of Expertise : C++, java, React

Location : New york

Help Required In : React Native

Communities

Your Developer Friends

Figure 6.9 *Communities Page*

Chapter 7. Testing and Verification

Testing on Postman

We utilized postman for our testing, verifying, and sharing of our backend APIs with teammates and for the development of this project. On postman, we created a group for all our APIs and defined the endpoint and sent requests to those endpoints, and tested and verified the expected results that the API gave after sending requests. All the results were monitored and checked if they were giving output as expected.

Integration Testing

To have integration testing we have evaluated all our API's interactions with each other. For our project, we divided it into various components like authentication which is login and logout. Other components were posts, profiles, jobs, and hackathons. The third component was messaging. We have ensured that each component works independently and there is no dependency on other components when integrated the output was as expected and when all the components interacted with each other all the results were as expected. There was no clash in APIs when we tested the interaction of the components with each other.

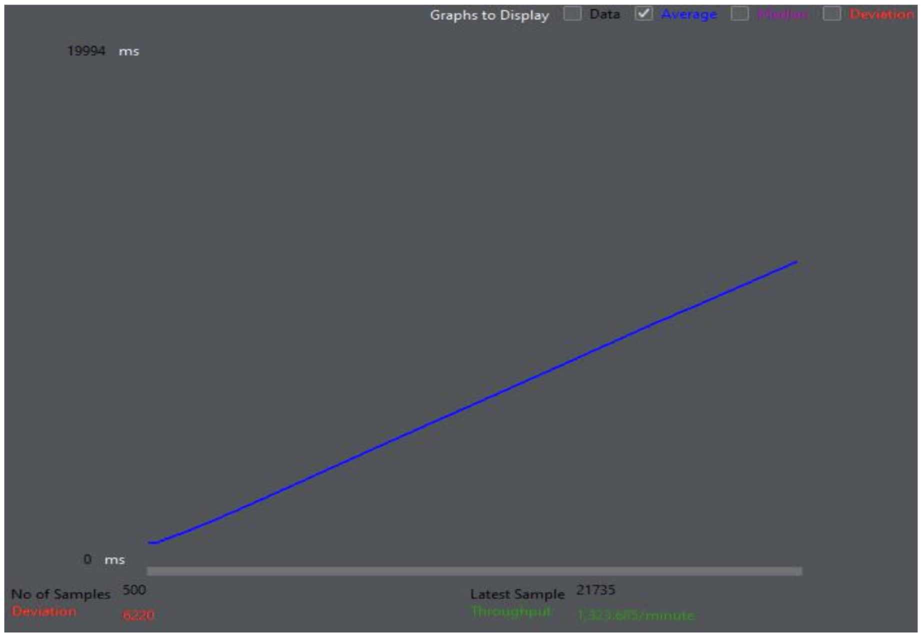
Chapter 8. Performance and Benchmarks

Main processing of data or any request happens in our backend server. In general, if we observe performance load, our frontend server handles 30% of the total load while the backend server handles around 70% of the total load. This is a clear indication that the backend server needs to be scalable. To make the backend server scalable we've used aws load balancer with the auto-scaling rule in between the frontend server and the backend server. After careful observations of overall web application performance, we have decided to set the auto-scaling rule to a CPU capacity setting of 50%. This means whenever the CPU capacity of the backend instance reaches 50%, AWS load balancer will spin another backend instance giving a seamless experience to users. We have observed that this load balancer has increased the uptime of our application by 100%.

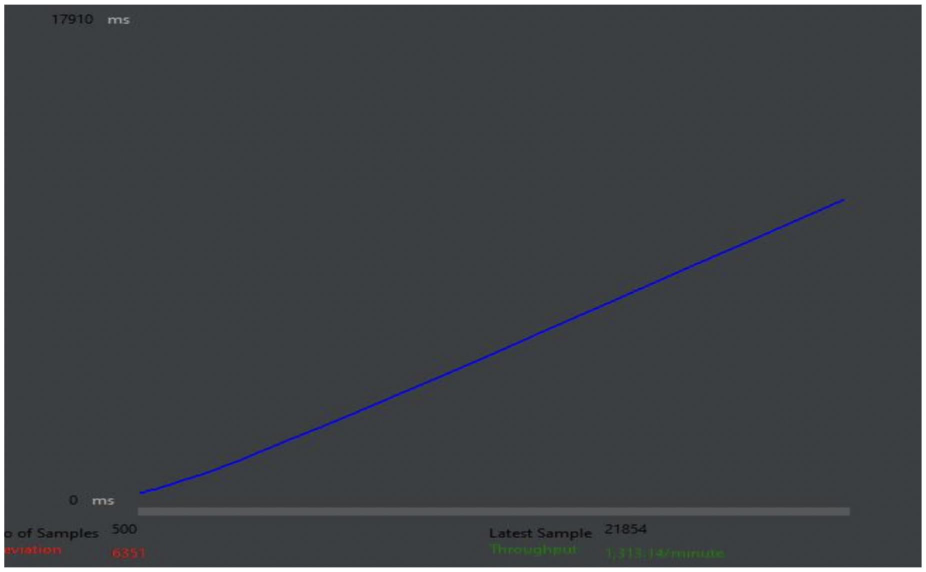
To prove our assumptions, we have done performance testing on our heavy-load backend API's. The main two APIs which are heavy processing in our application are "getAllPosts" and "message".

We concluded that a benchmark study on these APIs can determine the overall performance of our application. We have measured the throughput of these APIs with and without connection pooling.

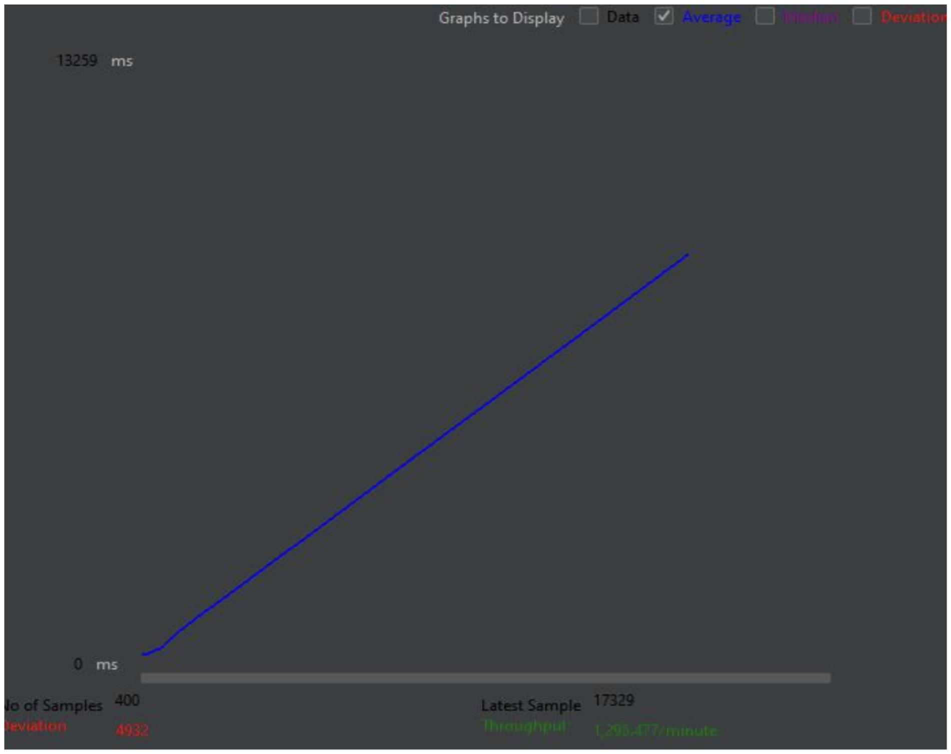
Apache Jmeter benchmark study result :



500 requests without connection pooling - getAllPosts

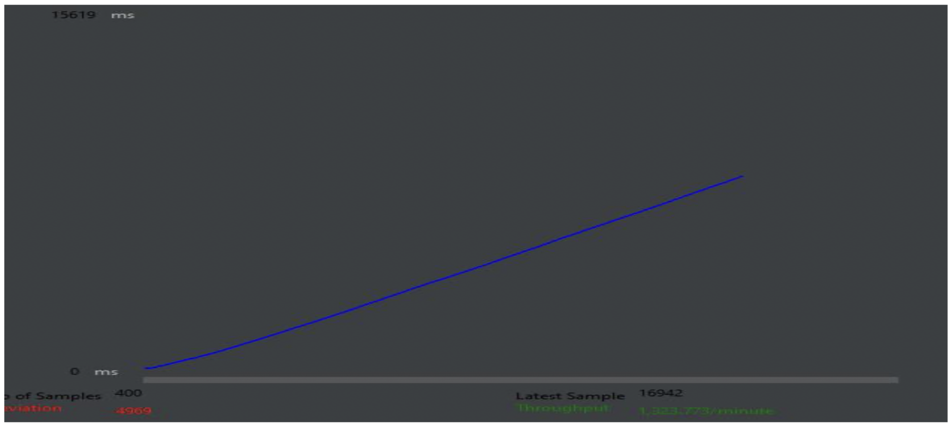


500 requests with connection pooling - getAllPosts



lo of Samples 400
Latest Sample 17329
eviation 4932
Throughput 1,295.477/minute

400 requests without connection pooling - messaging



of Samples 400
Latest Sample 16942
viation 4969
Throughput 1,323.773/minute

400 requests with connection pooling - messaging

API name	No of Requests	Connection Type	Throughput/minute
getAllPosts	500	Without Connection Pooling	1323.6
getAllPosts	500	With Connection Pooling	1313.14
messaging	400	Without Connection Pooling	1295.477
messaging	400	With Connection Pooling	1323.77

Chapter 9. Deployment, Operations, and Maintenance

Once its implementation is complete, DevConnect will be deployed on AWS. Load balancers will be enforced if the application fails to run effectively on a single AWS Instance. Separate AWS instances of front-end and back-end so that the application runs seamlessly. To ensure operational excellence, we will need to make the application scalable enough so that as the number of users grows, the application can handle it all. The application will be maintained and monitored by all team members, once it is deployed. There will be regular validation tests that will be running to test the load capacity of the application. As a future scope, we aim to do automated testing to test the application on a regular basis once it is deployed. Both APIs and front-end components will be tested so that the application works perfectly in all the domains.

Chapter 10. Summary, Conclusions, and Recommendations

Summary

DevConnect is a lightweight and interactive platform that is made from React, NodeJS, Socket io, ExpressJS, and MongoDB. This platform is a secure platform with jwt-based tokenization as the authentication. The platform helps millions of developers to resolve their issues instantly by posting their issues on the website or directly live messaging their peers in real-time. Along with these the platform also provides hackathons and jobs available and applies to them actively. Thus it provides all solutions to the developer's needs in one place.

The system is also scalable and can handle millions of requests. The system is deployed on AWS with a load balancer thus helping the platform to maintain scalability and work smoothly without any latency.

Conclusions

In conclusion, in Today's world technological advancement is growing by a second. There is a need for skilled software developers in every area. Although schools and universities are doing their best to train these engineers, there is always a scope for great learning when engineers come together and share in their community. To conduct this idea of engineers and technologists coming together for their own good, apps like DevConnect are essential.

With DevConnect, developers/engineers can come together and share their knowledge in an efficient way. In the coming future, we can see apps like DevConnect getting popular because of the increased number of engineers.

Recommendations for Further Research

We think that the following areas can be further explored to make the product more well rounded and future-proof.

- ML recommendation for users/communities to follow: Currently suggestions for the community shown to users are getting directly pulled from the database. However, in the future, we can add another backend to the application which will have a machine-learning recommendation system to give suggestions to the community based on the user's interest.
- Pair coding: right now we have implemented a messaging module in which connected users can message each other and share their code, we can further

extend this by implementing pair coding where two users can simultaneously code on a platform, providing close to an in-person experience.

Glossary

REST: It is an acronym for Representational State Transfer Application Programming. Rest API provides easy access to web services by setting up connections between the client and server.

AWS S3: It is a service provided by AWS that allows object storage through a web services interface. AWS S3 stands for Amazon Web Services Simple Storage Services.

AWS Load balancer: It is yet another service provided by AWS, which allows the automatic distribution of incoming traffic based on certain rules (auto-scaling rules).

Socket IO: Socket Io is an event-driven library for real-time web applications, this library allows real-time bidirectional communication. For this reason, it is used in messaging-based web applications where real-time processing is required.

JSON object: JSON is an open standard file format that consists of key and value pairs. JSON is in human-readable format and is used in REST API communication.

References

[1] B. V. Hanrahan, G. Convertino, and L. Nelson. “Modeling problem difficulty and expertise in StackOverflow”. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion (CSCW '12). *Association for Computing Machinery, New York, NY, USA*, 91–94.

[2] V. Bhat, A. Gokhale, R. Jadhav, J. Pudipeddi and L. Akoglu, "Min(e)d your tags: Analysis of Question response time in StackOverflow," *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2014.

[3] H. Sanchez and J. Whitehead, "Source Code Curation on StackOverflow: The Vesperin System," *IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015.

[4] N. Gantayat, P. Dhoolia, R. Padhye, S. Mani, and V. S. Sinha, "The Synergy between Voting and Acceptance of Answers on StackOverflow - Or the Lack Thereof," *IEEE/ACM 12th Working Conference on Mining Software Repositories*, 2015.

[5] P. Berger, P. Hennig, T. Bocklisch, T. Herold and C. Meinel, "A Journey of Bounty Hunters: Analyzing the Influence of Reward Systems on StackOverflow Question Response Times," *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2016.

[6] C. Treude, O. Barzilay and M. Storey, "How do programmers ask and answer questions on the web? NIER track," *33rd International Conference on Software Engineering (ICSE)*, 2011.

[7] K. Silveira, S. Musse, I. H. Manssour, R. Vieira and R. Prikladnicki, "Confidence in Programming Skills: Gender Insights from StackOverflow Developers Survey," *IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2019.

[8] G. Digkas, N. Nikolaidis, A. Ampatzoglou and A. Chatzigeorgiou, "Reusing Code from StackOverflow: The Effect on Technical Debt," *45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2019.

[9] C. Lam, and E. Biggerstaff, “Finding Stories in the Threads: Can Technical Communication Students Leverage User-Generated Content to Gain Subject-Matter Familiarity?”, *IEEE Transactions on Professional Communication*, vol. 62, no. 4, December 2019.

[10] J. R. Cedeño González, J. J. Flores Romero, M. G. Guerrero, and F. Calderón, "Multi-class multi-tag classifier system for StackOverflow questions," *IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, 2015.

Appendices

Appendix A.