Civitas: A Community Engagement Website

Christopher Ugbomah
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
efy163@usask.ca

Yongchang He
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
yon534@usask.ca

***Abstract*—This social computing project is motivated by the need to ease the difficulties faced by newcomers e.g., students to a new university community, in accessing answers to solve their urgent problems. The purpose of this project is to explore the possibilities of harnessing the power of virtual communities and getting people to offer help to new members of a community who are not familiar with the new environment. This project uses PostgreSQL for the database deployment, AngularJS for building the client-side framework, NestJS for server-side application, and Prisma for database management. These technologies work together leading to a full-stack web application. Various engagement strategies are employed in our system to keep registered users active.**

Keywords—social computing, engagement, web, full-stack app

I. Introduction

Individuals frequently find themselves in new communities and environments in a constantly changing world where people are now more on the move than ever before. The joys of arriving newly to a community can often be short-lived, usually replaced by fears and anxieties about how to fit in. The greatest worries are often triggered by the uncertainty of whether one has all the necessary information required to succeed and thrive in the new environment.

The challenge we seek to address for people who find themselves in this situation as often expressed in concerns such as – where can I find people who might have experiences from similar situations and be willing to share them? Often, personal experiences and ideas can be further enriched by the knowledge of people who might have passed through similar decision-making stages and come out of it with some valuable insight to share – helping make better-informed decisions and plans.

This project seeks to explore the possibilities of harnessing the power of communal living and getting a group of people to aid individuals they may not have previously met, just by they share a common space e.g., neighbourhood, campus, department building, lab, etc? In a world driven by the continued and growing influence of social media, we sought to answer two questions viz: 1. Would new members of a community be sufficiently motivated to seek help from members of their community if it is made easily accessible to them? 2. Can members of a physical community be motivated enough to help integrate new community members through voluntary online social interactions in the form of information and knowledge exchange?

We expect that adopting and using solutions such as what we propose would provide valuable insights into the above questions.

II. Motivation

The purpose of this site is to provide new community members an avenue to ask questions and hear directly from individuals like themselves who have lived through the same experience, leveraging on the benefits of shared, communal knowledge and insights. A secondary aim is to provide an avenue where members can catalogue and share personal experiences through blog posts and/or articles, often with the aim of eliciting insights, views, and opinions of others through comments and feedback.

III. Audience

The site is created primarily for new arriving members at a community (e.g., Usask, Saskatoon) or group (e.g., MADMUC Lab, Usask Athletics Team, etc). Existing community members are inclined to share their knowledge and experience with others, particularly members of the primary target audience.

IV. Review of similar apps

Some solutions in the market offer some of our proposal's listed goals. One notable solution is provided by the website - nextdoor.com [1], as is shown in Figure 1. On review of the site, however, we found some features that were fundamentally different from some of our key features like ease of sign-up.

We also do not believe, at this time, that our solution should incorporate features like a “marketplace” etc.

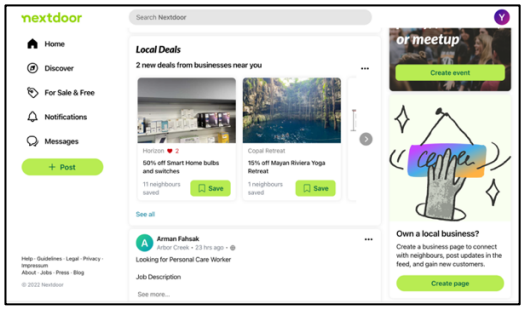

Fig. 1. Screenshot of Nextdoor official website

V. DIFFERENCE

Our proposed solution will not cover the following features: the non-restriction of the solution to a single, pre-defined community e.g. a specific neighbourhood, town, or city; the ability to create a hierarchical community structure e.g. the provision of features to allow for the creation of sub-groups within the community; the inclusion of other activity types that occur within a community e.g. business advertising, buying/selling, the direct connection within the website to other members of the community, etc.

VI. ARCHITECTURE

A. Methodology

This project proposes a web-based solution driven and animated by community members. Older or more experienced community members would drive the site's main contents through blog posts, articles, comments, and general answers to questions raised by new community members or questions they might have had themselves when they first arrived.

The strategies adopted would be to attract and retain membership through the following:

1. Immediate access to community archived resources for first-time visitors without the requirement to sign-up or register; 2. A simple, minimal sign-up and onboarding process for newly registered site members; 3. An integrated email notification system to help disseminate requests, thereby improving the chances for a new-immediate response; 4. Access to the site's "Help" and "FAQ" pages to ensure a user/visitor receives value while on the site; 5. A shared responsibility to help build trust and confidence in the site through the adherence to the site's terms and conditions, member involvement in the identification and flagging of spam messages, as well as the availability of a human moderator to help curate content and maintain the site's standards.

B. Engagement

First time visitors to the site can immediately search through existing topics, either featured as blog posts or previously asked questions, to which answers have been provided, without the need to register. Unrestricted access is also provided to a select number of "Frequently Asked Questions" (FAQs) to quickly get the site's visitors to gain some immediate insight and value. A simple, minimal sign-up process (first name, email, and password) is provided for visitors who might: have questions or topics not covered in the FAQs or found in the question bank; or want to comment, rate, or provide answers to existing questions. An integrated mail notification system to notify members of the community of a new request, answer, or blog post.

Other strategies to keep users engaged include: A "Help" page with basic, simple instructions on how to interact with the site; the provision of a micro blogging feature to allow registered members keep a personal journal they would like to share with others; an "Active Members" page that shows the profiles of the top 10 members of the community; a "leader board" to recognize the contributions of members, based on the following factors: response time to questions, rating of responses, the number of engagements/comments generated from a member's question, response or blog.

The engagement strategies employed would attract visitors to and retain members on the site because of the following: the site provides immediate benefit to first-time visitors without the compulsion to register on the first visit; the sign-up process is minimal and can be completed with very little data collected from a visitor; the rewards accruable to members who contribute to the site's goal are constantly on display (through the "Active Members" page). Contributions to the site by registered members are always noted and "celebrated" here. Email notifications are always sent to members who raised questions or requested assistance each time a response or answer is given. A measure of "healthy competition" to support the community is incorporated using leader boards. Confidence and trust will be built in the system by giving members the ability to flag spammers and report offensive speech, cyberbullying, and general anti-social behaviours. The presence of a human moderator to further help curate content and ensure the site's standards are kept.

C. Technologies

Our web solution would be built using the following technology stack: NestJS Framework for server-side communication; AngularJS Framework for Front-end services; Prisma ORM for database access; Postgres DB database for data persistence.

NestJS

NestJS is a cutting-edge Node.js framework for creating massive, scalable applications. It was developed by Kamil Mysliwiec as a framework facilitating the development of scalable server-side applications. As of January, it boasts over 23k GitHub stars and its weekly npm download rate is almost 180k [2]. New modules like GraphQL, Microservices, or ORM may be readily added to the codebase thanks to its modular Angular-like module architecture with good documentation and recommendations, making it a great solution[3]. Through a built-in module for testing, the framework supports testing in Jest20, a popular JavaScript testing library. When creating a monolith using NestJS, the rules recommend breaking the codebase into smaller modules. Creating more maintainable modules saves time when a program is divided into microservices. The concept below are the core fundamentals of NestJS to build a basic application:

NestCLI: a programme with a command-line interface that not only accelerates the creation of web applications overall but also automates the project's app initialization[4]. The project directory, node modules, and a few boilerplate files may all be built and installed with only two commands.

Controllers: Manages requests and returns data to the client. The goal of a controller is to gather requests for the application. Which controller receives which requests is determined by the routing mechanism. Each controller may have more than one route on occasion, and different actions may be carried out by different routes[5]. The controllers' routing mechanism is shown in Figure 2. Nest makes use of classes and decorators to build a fundamental controller. Decorators associate classes with necessary metadata and allow Nest to build a routing map. To define HTTP methods of given endpoint, there are: @Get(), @Post(), @Put(), @Delete(), etc.

Providers: A fundamental concept in Nest. A provider's primary function is to inject dependencies, which enables objects to develop a variety of connections with one

another[6]. A provider is, in essence, just a class that has the decorator @Injectable() added.

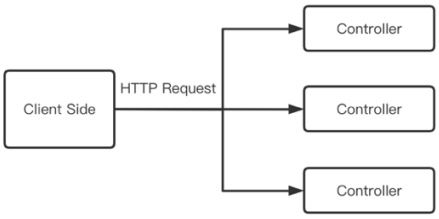

Fig. 2. NestJS Controllers' routing mechanism

Services: Responsible for operating the database and is intended to be managed by the controller. So, it could also be defined as a provider. Thus, NestJS decorate the class with @Injectable().

Module: A class with @Module() decorator. The module class is the place to connect Services, Controllers, etc. Every application has at least one root module (or the application module). The root module is basically the starting point that NestJS uses to build the application graph. Figure 3 demonstrates the diagram of module in NestJS. The Application Module is the starting module for Users Module, Orders Module and Chat Module. Similarly, Orders Module is the starting point for Feature_1 Module and Feature_2 Module. Chat Module is the starting point for Feature_3 Module.

Dependency injection: a programming paradigm where a class requests dependencies from external sources rather than making its own dependencies. With this approach, the class will receive the needed dependency in the constructor.

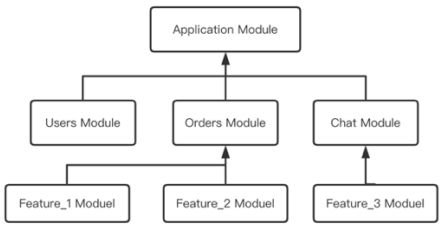

Fig. 3. NestJS modules diagram

AngularJS

AngularJS is a structural framework for dynamic online applications. Google created the framework. It is used for efficiently creating advanced single-page web applications[7]. And TypeScript, JavaScript's superset, is used in its writing. Types, interfaces, async functions, decorators, and many other capabilities are available, but in the end, it compiles simple JavaScript code that can be run on any browser.

Angular JS was originally developed in 2009 at Brat Tech LLC as the software behind an online JSON storage service, for Easy-make application for the enterprise. AngularJS was used as the frontend of the MEAN stack, that consisted of MongoDB database, Express.js web application server framework, AngularJS itself (or Angular), and Node.js server runtime environment[8].

As a method of software design, Angular applications adhere to modular programming ideals. The foundation of modular programming is the division of program functionality into interchangeable, logically separate modules. The angular modularity system is called NgModules. A logical unit's components or service providers may be contained within any NgModule[9]. The contained NgModule establishes the scope of those components' and service providers' functions. At least one module, the root module known as AppModule, is present in every Angular application. A NgModule is represented by a class with the @NgModule() decorator. Several properties in the decorator @NgModule() are used to characterize the module:

- Declarations: components and directives that belong to this module;
- Exports: components and directives that should be visible in other modules;
- Imports: other modules which are needed by this module;
- Providers: services that this module exports to the global set of all services;
- Bootstrap: root component.

Smaller programs often only have one root module, whereas more sophisticated applications frequently divide their functionality into a greater number of feature modules. Other modules from the Angular library can frequently be imported into Angular apps. For instance, the following code can be used to import FormModule from the form library:

Import { FormsModule } from '@angular/forms';

A class decorated with @Component() represents a component class. Each component has its own business logic in the TypeScript file and its template in HTML and CSS files. Together, components with their template define a view. @Component() decorator has the following properties:

- Selector: a CSS selector that tells Angular where HTML tag to insert an instance of the component;
- TemplateUrl: relative path to the template of the component;
- Providers: services that the component requires.

For connecting part of the template with the parts of the component, Angular uses two-way data binding. There are four forms of data-binding markup:

- Accessing the component's property value from the template using: {{ component's property name }};
- Passing a value from parent to child component using property binding: [child's component property]= " parent's component property";
- Calling component's method from the template using event binding: (click)= " component's method ";
- Connecting input boxes from template forms with component properties [(ngModel)]=" component's property name ".

Prisma

Prisma is an open-source ORM for Node.JS and Typescript that supports a broad range of SQL-based database adapters[10]. Prisma describes models and generates migrations using schema. Prisma also supports MongoDB at

the time of writing, although only as a preview feature in beta. Prisma works by having the generator generate type definitions from models defined in the schema. Model-generated types give type-safe definitions, resulting in fewer mistakes. Prisma, unlike other libraries, has migration support. Because of the use of shadow schema instead of other libraries like TypeORM, where migrations may be put in a separate file, these migrations are safe to employ.

Prisma Client JS is a type-safe database client that replaces traditional ORMs like Sequelize, Bookshelf, and Mongoose[11]. It allows us to access the database through plain JavaScript methods and objects without having to write the query in the database language itself. It acts as an abstraction in front of the database, so it's easier to write CRUD (create, read, update, and delete) applications.

Prisma Migrate is a powerful database schema migration tool. It uses a declarative data modelling syntax to describe our database schema. Prisma Migrate stores our entire migration history and easily lets us revert and replay migrations. It also allows users to run before-and-after hooks to execute scripts while migrating so users can populate the database with required values during a migration.

Prisma Studio allows us to visualize data via an Admin UI. Using this, users can perform CRUD (create, read, update, and delete) operations on the data. This is the easiest way to visualize data from the database as well as manipulate it.

PostgreSQL

PostgreSQL is an open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads[12].

PostgreSQL originates back to 1986 as part of the POSTGRES project at the University of California at Berkley and has been under active development. PostgreSQL has earned a reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open-source community behind the software to consistently deliver performant and innovative solutions[13], [14]. PostgreSQL supports all major operating systems. It is more powerful and faster for larger systems than SQLite or even MySQL.

D. Project Structure

The project structure is shown in Figure 4. This is a typical full-stack project structure. On the Client-side application, end users try to access all the features that are visible to them by making RESTful requests for HTTP web pages. This layer is developed with AngularJS framework which combinedly uses JavaScript, HTML&CSS. These requests go to Server-side application. In this Server-side application, NestJS framework is developed on top of NodeJS server application, and it determines what data/webpages a client needs and make corresponsive data request to the database for fetching this information. On listening the requests, the PostgreSQL database selects the required data and passes it to NestJS server, and finally the AngularJS application renders the data on webpage for the clients. Prisma is responsible for data management and working between server-side layer and database application layer.

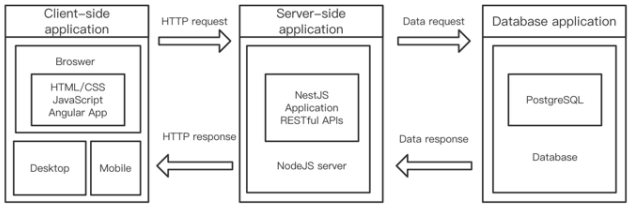

Fig. 4. Diagram of the project structrue

E. Client-side framework

Client-side framework (also called front-end framework) contains a single AngularJS application. The application has been refactored into multiple modules, as is shown in Figure 5.

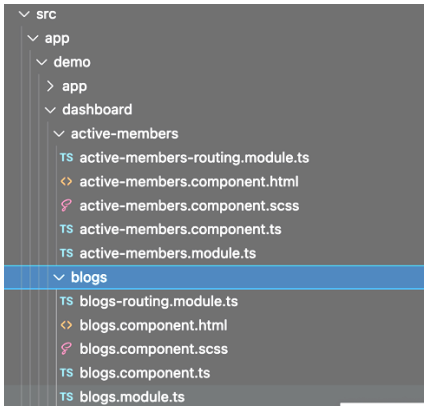

Fig. 5. Part of the structrue of modules

Angular application uses modularity system called NgModules. They are containers for a cohesive block of code dedicated to a workflow, or a closely related set of capabilities. They contain components, service providers, HTML&CSS files, and routing definition. Modules can import functionality that is exported from other modules and can also export selected functionalities for use by other modules.

All pages in this application are defined as module, and the root module, which is named AppModule is resides in app.module.ts. This front-end application can be launched by bootstrapping the root NgModule. Each module defines at least one component, which associated with a specific template. The client-side Angular application runs on port 4200 by default. Users can view the Angular application by navigating to localhost:4200 in any web browser.

Routing definition is simple in this client-side application. Unregistered users will only be limited to Home Page and FAQs page. They will be forwarded to Sign Up or Sign In page if they click 'Login' or 'Create Post'. Blogs page hosts all the existing blogs that created by current users. We use NgFor structural directive to render a template for each item in a specific collection[15]. If users click any titles on this page, they will be linked to show-page webpage, where complete content for specific posts will be rendered. Users can also click 'create post' upper right corner on this webpage to start composing their post. 'Publish' button will bring the user again to Blogs webpage. Users can have access to Profile page by clicking the link in the navigation bar on the left, and they can also proceed to 'create post' on this page.

F. Server-side framework

The Server-side framework (also referred to as the back-end) is a monolithic NestJS application refactored into several module to handle various "business" processes such as sign-up and sign-in, blog post creation, update, publish etc. The back-end exposes these resources as RESTful Http endpoints accessible only via the front-end. The structure of the back-end is as shown in the Figure 6 below.

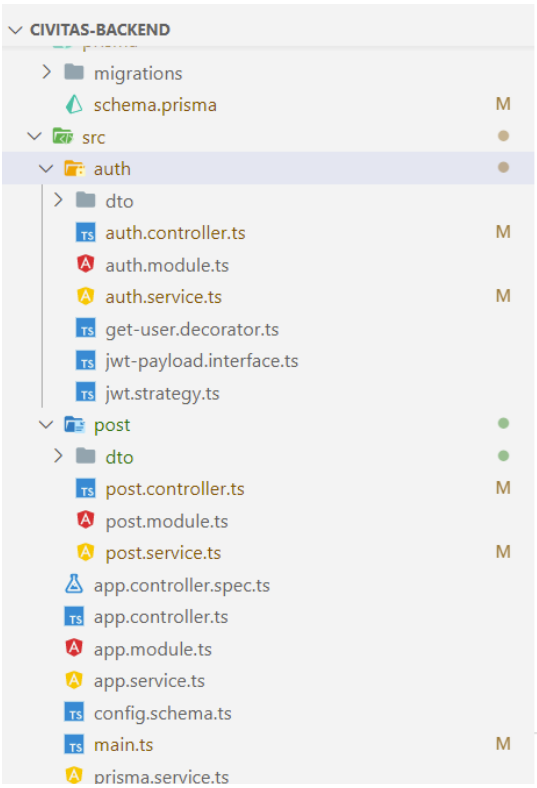

Fig. 6. Structrue of back-end project

Interaction with protected resources is only possible through signed security tokens, enabled by the use of the web industry standard JSON Web Tokens (JWT) [16].

G. Database framework

Database framework diagram is shown in Figure 7. To keep the database schema simple and to develop a minimal viable product, it does not cover the more advanced options like versioning and reviewing the posts.

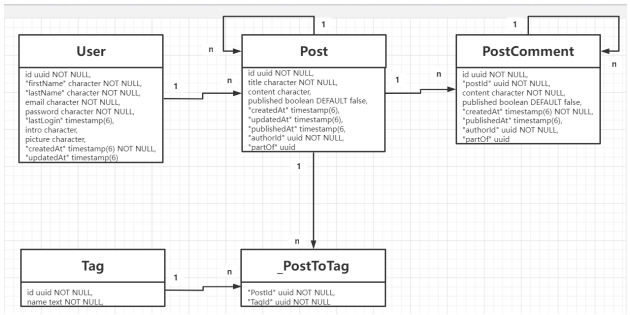

Fig. 7. Database framework

It provides options to review the comments by the post authors to avoid spamming so that only legitimate comments will be published and displayed on the Post Page.

User Table

The User Table is used to store user information of all the post authors. The same table can be used to relate the post authors so that all the authors can manage their own posts. Below mentioned is the description of all the columns of the User Table.

TABLE I. PROPERTIES OF USER TABLE

Properties	*Descriptions*
Id	The unique id to identify the user.
First Name	The first name of the user.
Last Name	The last name of the user.
Email	The email of the user. It can be used for login and registration purposes.
Password	The password hash generated by the appropriate algorithm. We must avoid storing plain passwords.
Created At	This column can be used to calculate the life of the user with the blog.
Updated At	This column can be used to store the last time a user updated his/her profile
Last Login	It can be used to identify the last login of the user.
Intro	The brief introduction of the Author to be displayed on each post.
Picture	The author profile picture storage address

Post Table

The Post Table is used to store the post data. Below mentioned is the description of all the columns of the Post Table.

TABLE II. PROPERTIES OF POST TABLE

Properties	*Descriptions*
Id	The unique id to identify the post.
Author Id	The author id to identify the post author.
Part Of	The parent id to identify the parent post. It can be used to form the table of content of the parent post of series.
Title	The post title to be displayed on the Post Page and the lists.
Published	It can be used to identify whether the post is publicly available.
Created At	It stores the date and time at which the post is created.
Updated At	It stores the date and time at which the post is updated.
Published At	It stores the date and time at which the post is published.
Content	The column used to store the post data.

Post Comment Table

The Post Comment Table is used to store the post comments. Below mentioned is the description of all the columns of the Post Comment Table.

TABLE III. PROPERTIES OF POST COMMENT TABLE

Properties	*Descriptions*
Id	The unique id to identify the post comment.
Post Id	The post id to identify the parent post.

Part Of	The parent id to identify the parent comment.
Published	It can be used to identify whether the comment is publicly available.
Created At	It stores the date and time at which the comment is submitted.
Published At	It stores the date and time at which the comment is published.
Content	The column used to store the comment data.

Tag Table & Post Tag Table

The Tag Table and Post Tag Table are used to store the post tags and their mappings. Below mentioned is the description of all the columns of the Tag Table.

TABLE IV. PROPERTIES OF TAG TABLE

Properties	***Descriptions***
Id	The unique id to identify the category.
Name	The tag title.

Below mentioned is the description of all the columns of the Post Tag Table.

TABLE V. PROPERTIES OF POST TAG TABLE

Properties	***Descriptions***
Post Id	The post id to identify the post.
Tag Id	The tag id to identify the category.

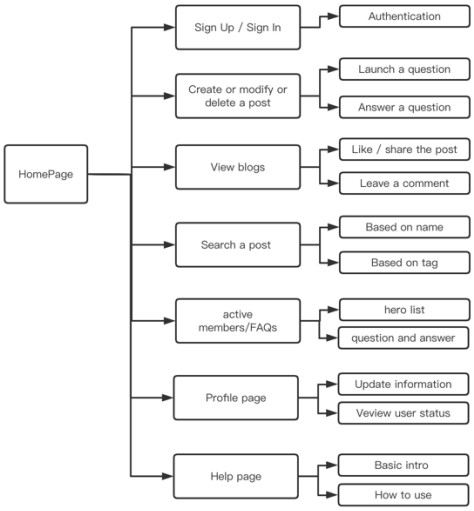

Fig. 8. Pages diagram

VII. STORYLINE

A. Pages

Pages are designed in this virtual community. The Sign Up / Sign In page is for user authentication. On Create Post page, users can create a post, modify a post, or delete a post. On the Blogs page, all the current posts are listed on the page, and the user can click on each title to see details in each blog. On the Home Page, there is an input area for the users to search the interesting content based on users' name, title or tags user created. There is a page called Active Members page, showing top 10 active members who give the most contribution to the community. The FAQs page shows the typical questions and answers that have been created previously, and users can have access to it even without registration. On the profile page, registered users can review their contributions, modify their profile information, and read notifications from the system. The Help page gives basic instructions on how to use resources on this virtual system. A diagram for all pages that designed in this project is shown in Figure 8.

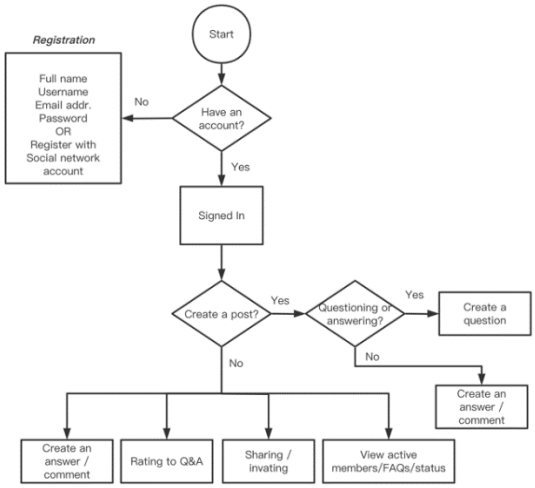

Fig. 9. Usage flow of the system design

B. Usage Flow

The system usage flow is shown in Figure 9. Typically, first time users can have access to Home Page and FAQs page to review posts and question, although all the contents are read only unless they finish registration. The sign-up process is simple and minimal. Only email, password, username, and full name required. After registration or sign-in, the users can have access to all the content existing on the virtual community. When they are logged in, they will directly see their contribution: How many questions they have answered, how many questions they have launched, and how many ratings for their questions and answers. The questions and answers with higher ratings indicate a solved situation will be posted longer and in a higher position, where most members can see them when logged into the platform. Now they can create a new post, add a comment to an existing post, create a question or provide answer to an existing question.

These questions on the platform will be displayed based on general tags. Users can search for interesting content based on the post name, title, or tags. They can edit their questions and answers after submission. Users can share their posts or questions/answers via a URL link or their social network platform.

C. Project Screenshots

Screenshots of the pages that included in this project are shown after the reference chapter.

VIII. EVALUATION

As at the time of writing, a public evaluation of the system is yet to be carried out, though responses received from individuals who have seen a walkthrough of our system has been positive thus far.

IX. Conclusion

Civitas has been designed as a community engagement platform, driven by members of specific individual communities with the altruistic aim of aiding other members of the community in accessing information in a safe and friendly environment.

While other similar systems exist out there, the sole focus of Civitas on being an online information exchange platform uniquely places it in a class of its own.

Currently in its early days, the reception it has received from individuals who have been briefly exposed to its workings show signs of encouragement, and its suitability for small communities and groups.

X. Future work

This community engagement website is still in the early stages of development. Deeper integration of client-side and server-side applications need to be implemented. This website has not been deployed to the public, so we haven't got feedback from volunteer users. Containerization and deployment of this full-stack project on commercial cloud platform will be considered.

Other features considered as part of our future work include

1. the ability for members of the community to "follow" particular users and/or "topics". This feature will enable the delivery of user-specific and/or topic-specific notifications to users who follow, say, a particular topic or user on the system.
2. It is also planned that each user on the system would be empowered to rate every individual post and/or comment. This feature will also be coupled with a "content filtering by rating" feature which will allow users specify the level of filtering they wish to apply to contents they are exposed to.
3. The support for sub-communities divided along specific interests and/or goals.

Acknowledgment

We would like to express our gratitude and appreciation to all those who gave us the possibility to complete this report. Special thanks are due to our Supervisor and Professor Dr. Ralph Deters, and Dr. Julita Vassileva, who give us precious courses and instructions leading to this report.

References

[1] "Newsfeed — Nextdoor." https://ca.nextdoor.com/news_feed (accessed Dec. 08, 2022).

[2] "NestJS: how and why to use it." https://www.merixstudio.com/blog/nestjs-how-and-why-use-it/ (accessed Dec. 09, 2022).

[3] "Documentation | NestJS - A progressive Node.js framework." https://docs.nestjs.com/ (accessed Dec. 08, 2022).

[4] "Scripts - CLI | NestJS - A progressive Node.js framework." https://docs.nestjs.com/cli/scripts#nest-cli-and-scripts (accessed Dec. 08, 2022).

[5] "Controllers | NestJS - A progressive Node.js framework." https://docs.nestjs.com/controllers#controllers (accessed Dec. 08, 2022).

[6] "Providers | NestJS - A progressive Node.js framework." https://docs.nestjs.com/providers#providers (accessed Dec. 08, 2022).

[7] "Angular - What is Angular?" https://angular.io/guide/what-is-angular (accessed Dec. 08, 2022).

[8] V. K. Kotaru, *Material Design Implementation with AngularJS: UI Component Framework*. Apress, 2016. doi: 10.1007/978-1-4842-2190-7.

[9] "Angular - NgModule." https://angular.io/api/core/NgModule (accessed Dec. 08, 2022).

[10] "What is Prisma? (Overview)." https://www.prisma.io/docs/concepts/overview/what-is-prisma (accessed Dec. 08, 2022).

[11] "Generators (Reference)." https://www.prisma.io/docs/concepts/components/prisma-schema/generators#binary-targets (accessed Dec. 09, 2022).

[12] "PostgreSQL: The world's most advanced open source database." https://www.postgresql.org/ (accessed Dec. 08, 2022).

[13] "PostgreSQL: Documentation: 15: 2. A Brief History of PostgreSQL." https://www.postgresql.org/docs/current/history.html (accessed Dec. 09, 2022).

[14] "PostgreSQL: About." https://www.postgresql.org/about/ (accessed Dec. 09, 2022).

[15] "Angular - NgFor." https://angular.io/api/common/NgFor (accessed Dec. 08, 2022).

[16] M. B. Jones, B. Bradley, and S. Sakimura, *JSON Web Token (JWT)*. IETF, 2015. doi: 10.17487/RFC7519.

XI. Project web pages screenshots