**TRUBA QnA**

**MAJOR PROJECT REPORT**

**Submitted in partial fulfillment of the requirement for the Degree of Bachelors of Engineering in Computer Science & Engineering**

**Submitted To: Prof. Amit Saxena (HOD,CS Dept.)**

****

**[RAJIV GANDHI PRODYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)]**

**Submitted By: Abhishek Argarey (0114CS161004)**

**Group Members:**

**Abhishek Argarey (0114CS161004)**

**And Amber Pandey (0114CS161020)**

**Under The Guidance of:**

**Miss Neha Gupta**

**(Teacher , Computer Science Department)**

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING / INFORMATION TECHNOLOGY**

**TRUBA Institute of Engineering & Information Technology, Bhopal**

**SESSION: 2019 - 2020**

**TRUBA INSTITUTE OF ENGINEERING & INFORMATION TECHNOLOGY, BHOPAL**

****

**(Session: 2019 - 2020)**

**DEPARTMENT OF Computer Science And Engineering**

**CERTIFICATE**

This is to certify that **Abhishek Argarey** and **Amber Pandey** Students of **CSE - VIII Semester** of **“Truba Institute of Engineering & Information Technology, Bhopal”** has completed their Major Project titled **“PROJECT NAME”**, as per the syllabus and has submitted a satisfactory report on this project as a partial fulfillment towards the award of degree of **Bachelor of Engineering** in **Computer Science and Engineering / Information Technology** under **Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal**.

**NEHA GUPTA Prof. AMIT SAXENA Dr. RAJEEVARYA**

**(Project Guide) Head(CSE / IT) Director**

**(Teacher) TIEIT, Bhopal TIEIT, Bhopal**

**(CSE / IT)**

**DECLARATION**

We the undersigned solemnly declare that the project report “**TITLE OF PROJECT”** is based on my own work carried out during the course of our study under the supervision of **Miss Neha Gupta, Teacher, CS Dept.**.

We assert the statements made and conclusions drawn are the outcomes of my own work. I further certify that

1. The work contained in the report is original and has been done by us under the general supervision of our supervisor.
2. The work has not been submitted to any other Institution for any other degree / diploma / certificate in this university or any other University of India or abroad.
3. We have followed the guidelines provided by the university in writing the report.

Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

**ABHISHEK ARGAREY [0114CS161004] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**AMBER PANDEY [0114CS161020] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ACKNOWLEDGEMENT**

In this semester, we have completed our project on **“TRUBA QnA”**. During this time, all the group members collaboratively worked on the project and learnt about the industry standards that how projects are being developed in IT Companies. We also understood the importance of teamwork while creating a project and got to learn the new technologies on which we are going to work in near future.

We gratefully acknowledge for the assistance, cooperation guidance and clarification provided by **“Miss Neha Gupta”** during the development of our project. We would also like to thank our Head of Department **Prof. Amit Saxena** and our Director **Dr. Rajeev Arya** Sir for giving us an opportunity to develop this project. Their continuous motivation and guidance helped us overcome the different obstacles for completing the Project.

We perceive this as an opportunity and a big milestone in our career development. We will strive to use gained skills and knowledge in our best possible way and we will work to improve them.

**ABHISHEK ARGAREY [0114CS161004] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**AMBER PANDEY [0114CS161020] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**LIST OF FIGURES**

**LIST OF TABLES**

**ABSTRACT**

**TRUBA QnA**

**TRUBA QnA** is an Institution based question-and-answer website where questions are asked, answered, followed, and edited by the faculties and students.

TRUBA QnA allows users to create user profiles with visible real names, photos, site use statistics.

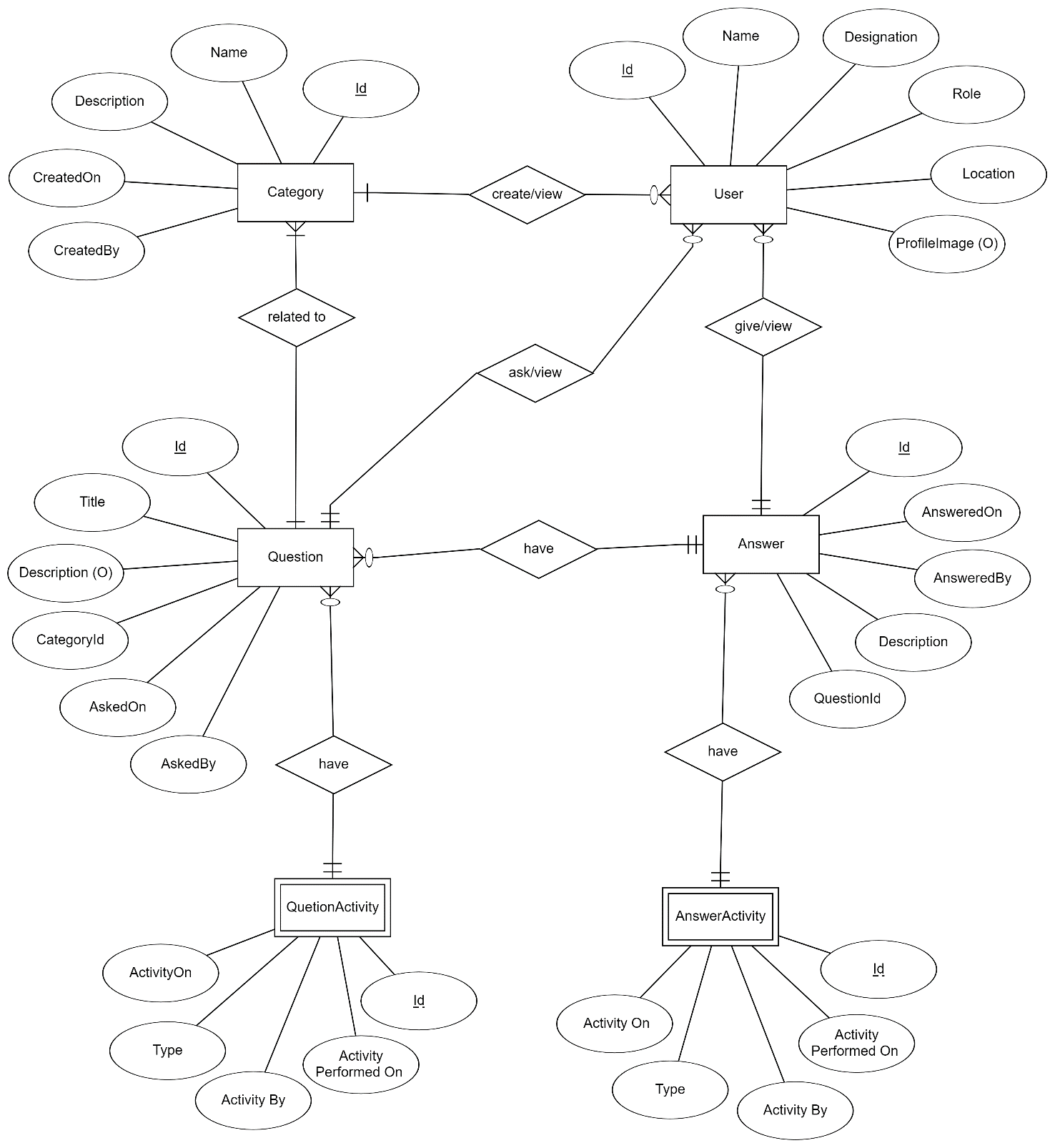
After you have created your account on it, it is important to understand some basic features that will help you when using the site. After login you will be navigated to home page where you can find all kind of previously asked questions and their details. Here you can ask your questions or give answers to existing questions.

In Category page you will find all the categories and their details. And can create your own category.

In User page you will find all the active users of the application and their details and their asked answered and solved queries details.

There are two important things in Truba QnA you need to know i.e. Questions and categories. each question is related to a category. You need to select a category while adding a question

If you don’t find your required category you can create it from category page. All questions and answers are visible to every user. You can choose best answer only for questions which are asked by you only.

**ER Diagram {need to paste in right chapter}**

**Introduction to angular FrameWork {need to paste in right chapter}**

Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.

The architecture of an Angular application relies on certain fundamental concepts. The basic building blocks are NgModules, which provide a compilation context for components. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules. An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

* Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data.
* Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.

Both components and services are simply classes, with decorators that mark their type and provide metadata that tells Angular how to use them.

* The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.
* The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI).

An app's components typically define many views, arranged hierarchically. Angular provides the Router service to help you define navigation paths among views. The router provides sophisticated in-browser navigational capabilities.

## Modules

Angular NgModules differ from and complement JavaScript (ES2015) modules. An NgModule declares a compilation context for a set of components that is dedicated to an application domain, a workflow, or a closely related set of capabilities. An NgModule can associate its components with related code, such as services, to form functional units.

Every Angular app has a root module, conventionally named AppModule, which provides the bootstrap mechanism that launches the application. An app typically contains many functional modules.

Like JavaScript modules, NgModules can import functionality from other NgModules, and allow their own functionality to be exported and used by other NgModules. For example, to use the router service in your app, you import the Router NgModule.

Organizing your code into distinct functional modules helps in managing development of complex applications, and in designing for reusability. In addition, this technique lets you take advantage of lazy-loading—that is, loading modules on demand—to minimize the amount of code that needs to be loaded at startup.

## Components

Every Angular application has at least one component, the root component that connects a component hierarchy with the page document object model (DOM). Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.

The @Component() decorator identifies the class immediately below it as a component, and provides the template and related component-specific metadata.

Decorators are functions that modify JavaScript classes. Angular defines a number of decorators that attach specific kinds of metadata to classes, so that the system knows what those classes mean and how they should work.

### Templates, directives, and data binding

A template combines HTML with Angular markup that can modify HTML elements before they are displayed. Template directives provide program logic, and binding markup connects your application data and the DOM. There are two types of data binding:

* Event binding lets your app respond to user input in the target environment by updating your application data.
* Property binding lets you interpolate values that are computed from your application data into the HTML.

Before a view is displayed, Angular evaluates the directives and resolves the binding syntax in the template to modify the HTML elements and the DOM, according to your program data and logic. Angular supports two-way data binding, meaning that changes in the DOM, such as user choices, are also reflected in your program data.

Your templates can use pipes to improve the user experience by transforming values for display. For example, use pipes to display dates and currency values that are appropriate for a user's locale. Angular provides predefined pipes for common transformations, and you can also define your own pipes.

## Services and dependency injection

For data or logic that isn't associated with a specific view, and that you want to share across components, you create a service class. A service class definition is immediately preceded by the @Injectable() decorator. The decorator provides the metadata that allows other providers to be **injected** as dependencies into your class.

Dependency injection (DI) lets you keep your component classes lean and efficient. They don't fetch data from the server, validate user input, or log directly to the console; they delegate such tasks to services.

### Routing

The Angular Router NgModule provides a service that lets you define a navigation path among the different application states and view hierarchies in your app. It is modeled on the familiar browser navigation conventions:

* Enter a URL in the address bar and the browser navigates to a corresponding page.
* Click links on the page and the browser navigates to a new page.
* Click the browser's back and forward buttons and the browser navigates backward and forward through the history of pages you've seen.

The router maps URL-like paths to views instead of pages. When a user performs an action, such as clicking a link, that would load a new page in the browser, the router intercepts the browser's behavior, and shows or hides view hierarchies.

If the router determines that the current application state requires particular functionality, and the module that defines it hasn't been loaded, the router can lazy-load the module on demand.

The router interprets a link URL according to your app's view navigation rules and data state. You can navigate to new views when the user clicks a button or selects from a drop box, or in response to some other stimulus from any source. The router logs activity in the browser's history, so the back and forward buttons work as well.

To define navigation rules, you associate navigation paths with your components. A path uses a URL-like syntax that integrates your program data, in much the same way that template syntax integrates your views with your program data. You can then apply program logic to choose which views to show or to hide, in response to user input and your own access rules.

The above details are the basics about the main building blocks of an Angular application. The following diagram shows how these basic pieces are related.



**.Net {need to place in right chapter}**

**Introduction**

.NET Core is a free and open-source, managed computer software framework for Windows, Linux, and macOS operating systems. It is a cross-platform successor to .NET Framework. The project is primarily developed by Microsoft and released under the MIT License.

**History**

.NET Core 1.0, announced on November 12, 2014, was released on June 27, 2016, along with Microsoft Visual Studio 2015 Update 3, which enables .NET Core development. NET Core 1.0.4 and .NET Core 1.1.1 were released along with .NET Core Tools 1.0 and Visual Studio 2017 on March 7, 2017

.NET Core 2.0 was released on August 14, 2017, along with Visual Studio 2017 15.3, ASP.NET Core 2.0, and Entity Framework Core 2.0. .NET Core 2.1 was released on May 30, 2018. NET Core 2.2 was released on December 4, 2018.

.NET Core 3 was announced on May 7, 2019, at Microsoft Build. Version 3.0.0 was released September 23 2019.[15] With .NET Core 3 the framework will get support for development of desktop application software, artificial intelligence/machine learning and IoT apps.

The next release after .NET Core 3.1 will be .NET 5. The .NET Framework will be deprecated, and .NET 5 will be the only .NET going forward – hence the removal of the "Core" branding and skipping of version 4 to avoid confusion with the .NET Framework 4.x. The first preview of .NET 5 was released on March 16, 2020.

**Language support**

.NET Core fully supports C# and F# (and C++/CLI as of 3.1; only enabled on Windows) and partially supports Visual Basic .NET.

Currently VB.NET compiles and runs on .NET Core, but the separate Visual Basic Runtime is not implemented. Microsoft announced that .NET Core 3 would include the Visual Basic Runtime, after two years the announcement was updated to .NET Core 5.

**Difference in .Net and .Net Core**

## .NET Framework Advantages

If you’re a .NET developer who needs to build and release something fast and you don't have time to learn .NET Core, then the .NET Framework is your choice. .NET Core does have a learning curve.

If you’re maintaining and upgrading existing .NET apps, .NET Framework is your choice. Porting an existing .NET app to a .NET Core app requires some work.

Looking for a job? You may not find too many jobs in .NET Core right now.

.NET Framework is what it is. The current version of .NET Framework, 4.8, is supposed to be the last version of .NET Framework. There will be no more new versions of .NET Framework planned in the future.

## .NET Core Advantages

If you’re building a new application and have a choice between .NET Core and .NET Framework, .NET Core is the way to go.

Microsoft just announced .NET Core v 3.0, which is a much-improved version of .NET Core. If you want to learn and build for the future, .NET Core is the way.

.NET 3.0 now supports WPF and Windows Forms. .NET Core 3.0 also supports cross-development between UWP, WPF, and Windows Forms. This provides developers the flexibility to bring modern interfaces of UWP into Windows Forms and WPF.

If you’re building a new app and your company has approved it, .NET Core is the way to go. .NET Core will continue to get better and better.

.NET Core is better suited for cross-platform needs. .NET Core apps are supported on Windows, Linux, and macOS. Microsoft’s popular open-source code editor, Visual Studio Code, is supported on Windows, Linux, and macOS. VS Code supports the modern needs of code editors including IntelliSense and debugging. Most third-party editors, such as Sublime, Emacs, and VI, work with .NET Core.

Microservices architecture is supported in .NET Core, which allows cross-platform services to work with .NET Core including services developed with .NET Framework, Java, Ruby, or others.

Containers are the VMs of today. .NET Core’s modularity, lightweight, and flexibility makes it easier to deploy .NET Core apps in containers. Containers can be deployed on any platform, cloud, Linux, and Windows. .NET Core works well with both Docker and Azure Kubernetes Service.

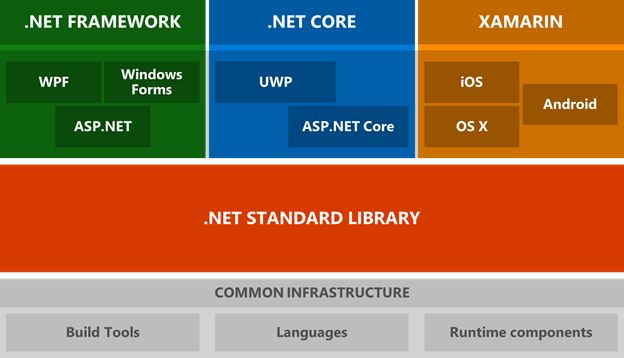
**A need for high performance and scalable systems.**

Performance and scalability were the two major key focus areas when .NET Core was developed. .NET Core and ASP.NET Core are the top-performing web frameworks according to some benchmarks.

.NET Core versions play well together. You can easily run multiple versions and apps side by side on the same machine.

## .NET Ecosystem

Let’s review the following .NET ecosystem architecture diagram, as of today.



As you can see from the above diagram, the .NET ecosystem has three major high-level components - .NET Framework, .NET Core, and Xamarin.

**Xamarin**is not a debate at all. When you want to build a mobile (iOS, Android, and Windows Mobile) apps using C#, Xamarin is your only choice.

## The .NET Framework

supports Windows and Web applications. Today, you can use Windows Forms, WPF, and UWP to build Windows applications in .NET Framework. ASP.NET MVC is used to build Web applications in .NET Framework.

## .NET Core

is the new open-source and cross-platform framework to build applications for all operating systems including Windows, Mac, and Linux. .NET Core supports UWP and ASP.NET Core only. UWP is used to build Windows 10 targets Windows and mobile applications. ASP.NET Core is used to build browser-based web applications.

|  |  |
| --- | --- |
| A high-performance and scalable system without UI | .NET Core is much faster. |
| Docker containers support | Both, but .NET Core is born to live in a container. |
| Heavily rely on the command line | .NET Core has better support. |
| Cross-platform needs | .NET Core |
| Using Microservices | Both, but .NET Core is designed to keep today's needs in mind. |
| User interface centric Web applications | .NET Framework is better now until .NET Core catches up. |
| Windows client applications using Windows Forms and WPF | .NET Framework |
| Already have a pre-configured environment and systems | .NET Framework is better. |
| Stable version for an immediate need to build and deploy | .NET Framework has been around since 2001. .NET Core is just a baby. |
| Have existing experienced .NET team | .NET Core has a learning curve. |
| Time is not a problem. Experiments are acceptable. No rush to deployment. | .NET Core is the future of .NET. |