Dissertation on

Empirical study of Lazy Loading on Angular Web Application Performance

Thesis submitted towards partial fulfilment

of the requirements for the degree of

Master of Technology in IT (Courseware Engineering)

Submitted by

SHREYASI BANERJEE

EXAMINATION ROLL NO.: M4CWE22009

UNIVERSITY REGISTRATION NO.: 154491 of 2020-2021

Under the guidance of

Mr. Joydeep Mukherjee

School of Education Technology

Jadavpur University

Course affiliated to

Faculty of Engineering and Technology

Jadavpur University

Kolkata-700032

India

2022

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **"Empirical study of Lazy Loading on Angular Web Application Performance"** is a bonafide work carried out by **Shreyasi Banerjee (**Examination Roll No.: M4CWE22009) under our supervision and guidance for partial fulfillment of the requirements for the degree of Master of Technology in IT (Courseware Engineering)in School of Education Technology , during the academic session 2021-2022.

SUPERVISOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DIRECTOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DEAN - FISLM
Jadavpur University,
Kolkata-700 032

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL **

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination ----------

for evaluation of Thesis

** Only in case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of his/her Master of Technology in IT (Courseware Engineering) studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME: Shreyasi Banerjee

EXAMINATION ROLL NUMBER: M4CWE22009

THESIS TITLE: Empirical study of Lazy Loading on Angular Web Application Performance

SIGNATURE: **DATE**:

Acknowledgement

I feel extremely glad in presenting this thesis at School of Education Technology, Jadavpur University, Kolkata in the partial fulfilment of the requirements for the degree of M.Tech. IT(Courseware Engineering).

I really want to convey my gratitude to my guide Mr. Joydeep Mukherjee for his constant help, adequate information and suggestions.

I would also like to thank my classmates of M.Tech. IT(Courseware Engineering) who motivated me always to complete my research work successfully. I thank to all those who were associated with this research work.

Thanks and regards,

Shreyasi Banerjee

Examination Roll No.: M4CWE22009

Class Roll No.: 002030402009

Registration No: 154491 of 2020-2021

M.Tech. IT (Courseware Engineering)

School of Education Technology,
Jadavpur University,
Kolkata-700032

TABLE OF CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

Finding a suitable modern strategy that would enable quick creation and easy scalability of an educational application was the major goal of several of the current research articles based on the analysis of numerous contemporary web frameworks. After comparing the numerous features provided by the other frameworks and those in Angular, it was found that Angular is the best framework to work with because of its efficiency, practicality, and cross-platform compatibility.

But it was made clear during the comparison that Angular is a complete solution even without considering routing. Developers instantly put up routing in the companion module (App Module) when creating a simple web application..However, because routing has a significant impact on application performance, it is necessary to consider it when the configuration is complex and includes specific guards and resolver services. This made it possible to pursue this as a study topic.

1.Introduction

A web application is a sort of computer program that runs in a web browser and makes use of a variety of web technologies to do various tasks on the internet. To run, a web application needs a Web server, an application server, and a database. A web application can be designed for a variety of purposes and can be used by anyone, such as a person or an entire company, for a variety of reasons. With the increasing influence and need for mobile applications, the demand for cross-platform frameworks has risen. Angular is a Google-created open-source front-end framework for building dynamic modern web apps.To minimise redundant code and make programmes lighter and faster, it uses the TypeScript programming language, which is based on JavaScript.

Web technology is the use of markup languages and multimedia packages to allow computers to connect with one another. Web technology has evolved dramatically over the last few decades,

from marked-up web pages to the capacity to execute extremely precise tasks on a network without interruption. Web technology has a big impact on advertising, easy trade, research work, business promotion and innovation, digital transaction, money management.

In today's world, to build a reliable, user friendly web app Angular is a one of the best client-side open-source framework. It is primarily aimed to develop Single Page Applications. It is designed for web, desktop and mobile platforms.

2. Background Study

The main objective of some of the existing research papers based on the analysis of various contemporary web frameworks was to find a suitable contemporary approach that would enable quick development and simple scaling of an educational application. It was determined that Angular is the finest framework to work with due to its effectiveness, convenience, and cross-platform compatibility after comparing the many

characteristics offered by the other frameworks and those in Angular. However, it was emphasized throughout the comparison that Angular is a comprehensive solution without taking routing into account. When building a straightforward web application, developers immediately set up routing in the companion module (App Module). Routing, however, is crucial to take into account when the configuration is complicated and involves specialized guards and resolver services because it has a big impact on application performance. Thus, this opened up the possibility of using this as a research topic.

In this section let's talk about the current emerging versions of angular and then will pave the way through angular routing & lazy loading and how it impacts on angular performance.

Technology evolves rapidly and at the same time computing devices are becoming more powerful, capable of performing complicated tasks. Modern web based applications need to be

secure, flexible, user-friendly. So, in this paper we are going to use Angular to build an application.

Angular version makes it easy to build applications with the web. It was created with the intention of generating dynamic web applications. It combines declarative templates, dependency injection, end to end tooling and integrated best practices to solve development challenges.

In order to provide more benefits to business owners and programmers, Angular has developed an ecosystem around it and is still growing. Every six months, Angular releases a new version that includes new modules and functionality. Google has already made services that assist users write less code better, enabling bundles for various browser versions, and is presently working to enhance the Angular SEO skills.

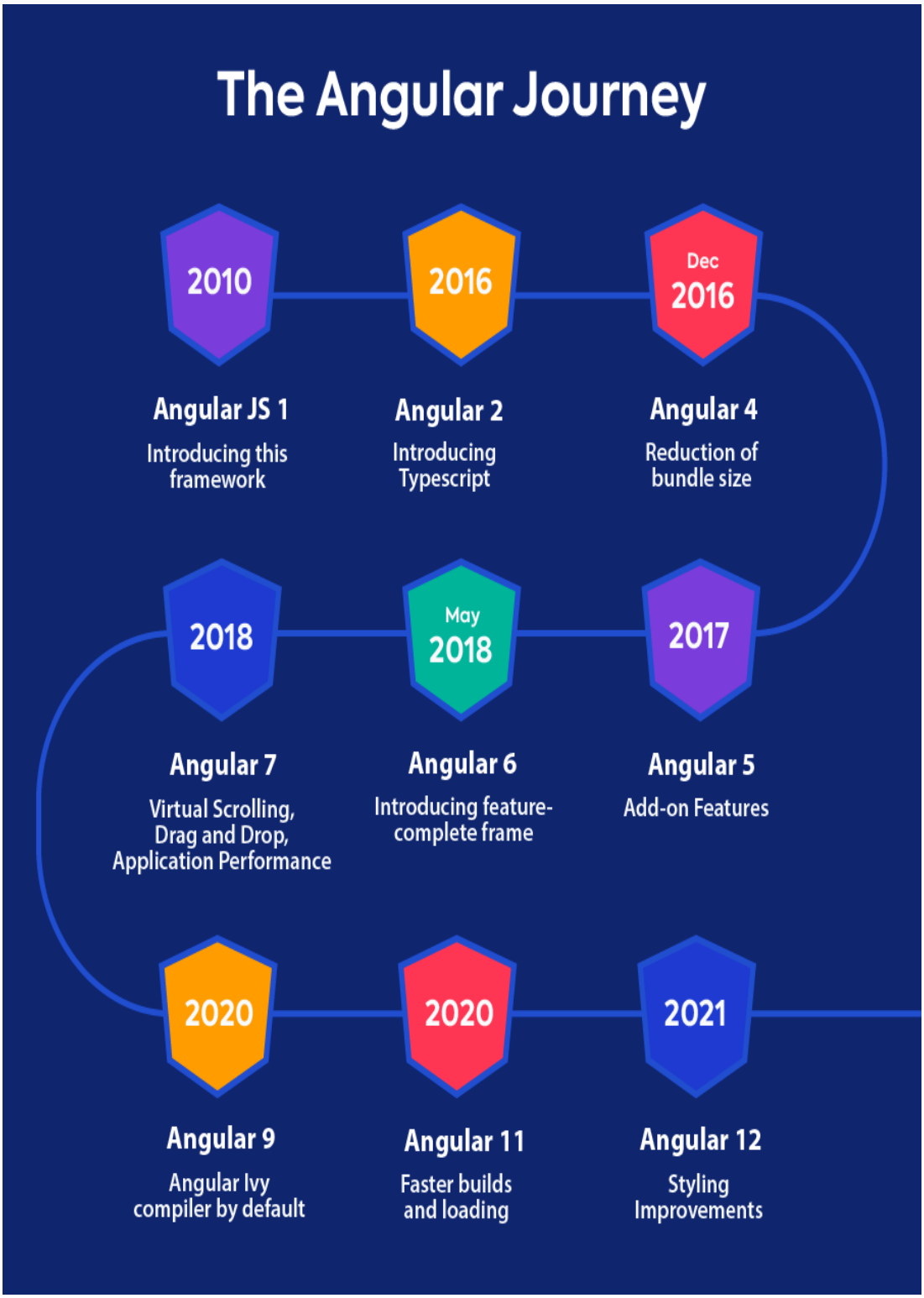


Figure 1

Source: https://miro.medium.com/max/1400/0*SC6makrT6zrnDtOh

3. Literature Survey

[1] the author Mohamed Sultan proposed comparative analysis of different modern web frameworks, to find a powerful front-end solution that offers components to help people build easy-to-use, readable, and maintainable code. After considering various aspect available in the other frameworks and that in angular it was concluded Angular to be the best framework to work with because of its efficiency, convenience, and cross-platform support.

[2] the authors Slavina Ivanova and Georgi Georgiev while showing contrast of the most recent emerging frameworks for creating mobile and online applications from various angles mentioned that Angular is a complete solution without considering routing. When a simple web app is created routing are directly configured into the companion module (App Module).But when the configuration is complex and includes specialized guards and resolver service it is very important to

consider routing as it makes significant impact when it comes to application performance .Thus this paved our way to consider this as our research topic.

[3] the author Sanja Delčev, Dražen Drašković investigation revealed while doing side-by-side examination based on specific criteria and to bring the analysis to a close.on the emerging frameworks. As Backbone gives developers a lot of decision-making power, it differs dramatically from AngularJS, Ember, and Knockout. The latter three frameworks offer a lot more fully developed functionality and handle most of the decision-making on their own.

[4] the author M.A.Jadhav , B.R.Saswant and A.Deshmukh proposes an algorithm that shows building a single-page application is relatively simple. When it loads completely the first time, it updates only a portion of a certain page with each request after that. adjusting based on how the server reacts to client interactions without performing a full page refresh.

Angular is a fully featured SPA framework which make its robust and efficient.

[5] The authors Hroje Puškarić 1,Aleksandar Đorđević ,Miladin Stefanović ,Marija Zahar ,Đorđević put forwards that SPA client functions similarly to desktop programmes in that it keeps track of its state, stores local data, reacts to events, and communicates with the user. Then, an SPA client searches data off the server when a desktop programme searches data from a database. Because of the speed of loading, an SPA client is typically built so that when an application is loaded from a server to a client, it is loaded only the amount required for the application's initial functionality: html, basic work, auxiliary modules, and data. The remaining information is loaded from the server as needed.

[6] as per the author Lassen, Anders proposed t hypothesis it might be argued that Angular's rigid structure makes understanding it and implementing it in apps less complicated

than anticipated, yet because of its thorough documentation, it serves the purpose of helping beginners master its capabilities quickly. As a result, Angular may be adopted by both seasoned engineers and novices, supporting the initial concept. React allows for more customization and does not come with a rigid structure. However, it can be argued that it is exhausting to adopt for a beginning because it is not comprehensive in and of itself and is dependent on all the other external libraries that can meet the purpose of the supplied application. The developer would need to identify the missing components, thus it is vital to practise for a while before gaining some experience and becoming accustomed to it. As a result, the evaluation of this study partially confirms the idea that React would be favoured by students because to its less complex structure.

[7] Here the author Jesse Koivukoski put forwards that the time it takes to process the JavaScript code in a web application contributes to the bundle's download time, which in turn affects

loading time. The browser will begin processing the JavaScript file as soon as it has been downloaded, which could take a while and cause the JavaScript engine to become unresponsive while under heavy load. The visual content of the app can also be rendered in the browser without any JavaScript code even having to be processed or downloaded by using server-side rendering. From the benchmarks in this thesis, it can be inferred that a single-page application's lengthy loading time is mostly caused by the size of the JavaScript file(s) and other necessary assets.

[8] Author Klaus Nyg˚ard proposed how the development of web applications is currently centred on single page architecture. It doesn't introduce any unexpected tactics and is, at its core, a fairly straightforward concept. However, SPA's strength lies in how it significantly alters the manner in which web applications are created.

[9] author S.A.Mousavi formulated the importance of software maintenance, also highlighted the importance of selecting a suitable framework for software development alongside showed the lack of research in JavaScript frameworks and technologies as a problem in the domain of software quality, particularly in web technologies.

Based on the above observation which paved our way to consider lazy loading and its impact on application performance as our research topic .

In our proposed model, to put forward the significance of routing in angular while developing user-friendly, complicated, and effective web applications, a straightforward survey is conducted. This illustrates how routing can have a big effect on an application's performance. When compared to an application without routing, the load time of an application with routing is significantly lower. In support of this two web applications are created

- Education Application (without routing) and
- Student Application (with routing).

As per the analysis outcome it was observed applications without routing, when compared to those with routing take substantially less time to load in turns improve the overall application performance.

Web application supporting educational software are created which focuses on Students enrolled. The advanced features like as data binding, templating , extensibility, variable observation, routing, testing Angular is one of the best when it comes to PWA. Angular code is much logical, consistent, and easy to follow as a front-end framework. A common issue noted by the developer community in prior versions of the framework was too big bundle size, which slowed the loading of apps. The designers of Angular have addressed this issue in subsequent versions with some enhancements. So, in Angular the loading time is reduced.

When a given route is active, Angular's lazy loading technology allows to load JavaScript components. It reduces the time it takes for a programme to load by breaking it into many bundles. Bundles are loaded as needed as the user navigates within the programme.

Lazy loading keeps the bundle size minimal, resulting in faster load times. To construct an Angular module @NgModule, the class decorator should be utilised, which uses a metadata object to specify the module.

The following are the most important characteristics:

import: This module's components are used with Array in other modules.

Declarations: It gets an array of the components.

Export: Defines a collection of components, directives, and pipes that other modules can utilise.

Provider: If the module is a root module, it declares services that are available to the entire application.

A. Single Page Application-

Single Page Applications are online applications that load a single HTML page and update only a portion of the page rather than the complete page with each click. The page does not reload or transfer control to another page during the procedure. This guarantees good performance and faster page loading. With each click, the SPA just delivers the information user need, and browser renders it. This differs from a typical page load, in which the server re-renders the entire page for each click and transmits it to the browser. It also makes the application much cost-effective.

B. Progressive Web Application (PWA)-

A Progressive Web Application (PWA) is a sort of web-based application that is constructed with HTML, CSS, and JavaScript. It can be used on any platform that has a standards-compliant browser. Businesses are turning to web apps to reach out to mobile consumers while avoiding the Play Store and App Store's

app constraints. Progressive web apps provide businesses more options. As mobile Internet browsing and app usage have dominated the marketplace, this has made them popular. It is not required to publish a PWA on any app store. Users can download your PWA directly from their browser and save it to their home screen just like any other app.

Progressive web apps have numerous advantages (Secure, Sharable).

It's possible to install it without using an app store.

4. Problem Statement

In the mother paper the importance of routing in angular is not shown. Routing can make a huge difference while it comes to make a user-friendly, reliable application. In this paper I'll try to create an educational application that will demonstrate the importance of routing and how it may improve speed and security of a web application.

5. Features Of Angular

Frameworks, in general, improve web development efficiency and performance by providing a consistent framework that eliminates the need for developers to rewrite code from the ground up. Frameworks are time savers that provide developers with all these additional functionality that can be added to software with minimal effort. While providing a common structure for developers to work with, Angular has distinct advantages. It enables users to create large, easy-to-manage applications.

Angular framework that makes developing web apps in HTML and typescript simple. Angular blends declarative templates, dependency injection, end-to-end tooling, and integrated best practices to overcome development difficulties faced by developers. Angular allows developers to create client apps for mobile, web, and desktop platforms.

Some features of angular are:

5.1 Cross-Platform: User can create progressive web applications with Angular (PWA). User can deploy a native or progressive app depending on your requirements. Ionic is a hybrid mobile SDK that allows to publish apps to the app store and then deploy them to the mobile web as PWAs.Additionally, Angular may be used to create desktop apps.

5.2 High Speed & Optimum Performance: The loading time of Angular apps is faster than any other front end framework.

5.3 Document-Object-Model (DOM): DOM treats an XML or HTML document as a tree structure in which each node is an object representing a part of the document. Angular uses regular DOM. This will update the entire tree structure of HTML tags until it reaches the data to be updated.

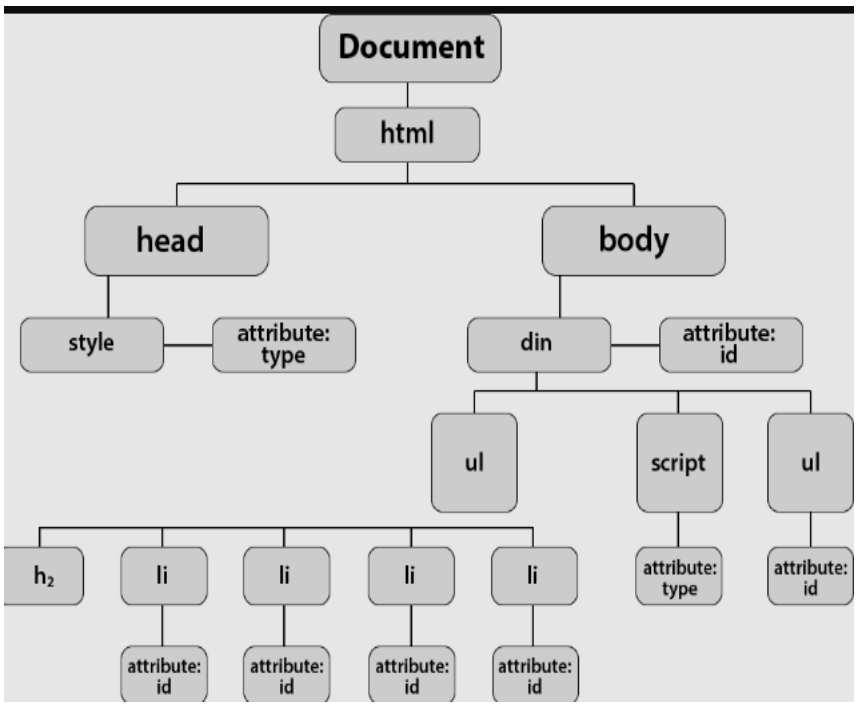


Figure 2

Source:https://miro.medium.com/max/1400/0*Sk5AAj4ze_bDFPA0.png

5.4 Angular Applications For Everyone: Angular is a magical front end platform that not only allows developing mind-blowing applications, but also enables to create high-end animations to enhance the user experience. Developers may create animation with less code because to Angular's API's simplicity.

You may also fix your bad code at any time with efficient unit testing frameworks like Jasmine and Karma. Angular has more than 11 build-in testing modules to ensure error-free code.

5.5 Less Code Framework: Angular is a low code framework comparing to the other front end technologies. You don't need to write separate code to link the MVC layers. And you don't require specific code to view manually also. The directives are also separated from the app code.These all together automatically minimize the development time.

5.6 Typescript: TypeScript defines a set of types to JavaScript,which helps users write JavaScript code that is easier to understand. All TypeScript code can be compiled with JavaScript and run on any platform. It is quite effective at detecting problems, which cuts down on development time. TypeScript additionally populates the root file settings automatically for quick

compilation. However, it is highly recommended as it offers better syntactic structure—while making the codebase easier to understand and maintain.One can install TypeScript as an NPM package with the following command: npm install -g typescript.

5.7 Data Binding: Data binding is a way for keeping data synchronised between a component and a view. . Angular refreshes the component whenever the user updates the data in the view. Angular refreshes the view when the component whenever the user updates the data in the view. Angular refreshes the view when the component receives new data. Angular's most powerful feature is the two- way data binding architecture. The View layer accurately depicts the Model layer, and the two layers are always in sync. If you make a modification to the model, the users will automatically see it in the view model. As a result, a large amount of time is saved during development.

5.8 Directives: Directives are the most challenging Angular features. You can create custom HTML tags serving as custom widgets with the help of directives. Developers can use these to decorate behavior-driven elements.These can be used to enhance behavior-driven elements by developers. With the use of directives, one may also change DOM attribute as per the requirement.

5.9 Testing: Angular uses Jasmine to run various tests. The Jasmine framework allows various functionalities to write different kinds of test cases. Karma is the task-runner for the tests.

6. Angular Architecture

6.1 MVC Architecture: MVC stands for Model-View-Controller. This architecture separates the application layer into Model, View, Controller. Model is related to all the data related to logic. View manages the data

display. While the controller plays as a connector between the view and model layers. In general, with MVC architecture, you can divide your app into sections and create code to connect them. In Angular, however, developers only need to divide the application into MVC and the framework will take care of the rest. It also saves a lot of time when it. comes to coding.

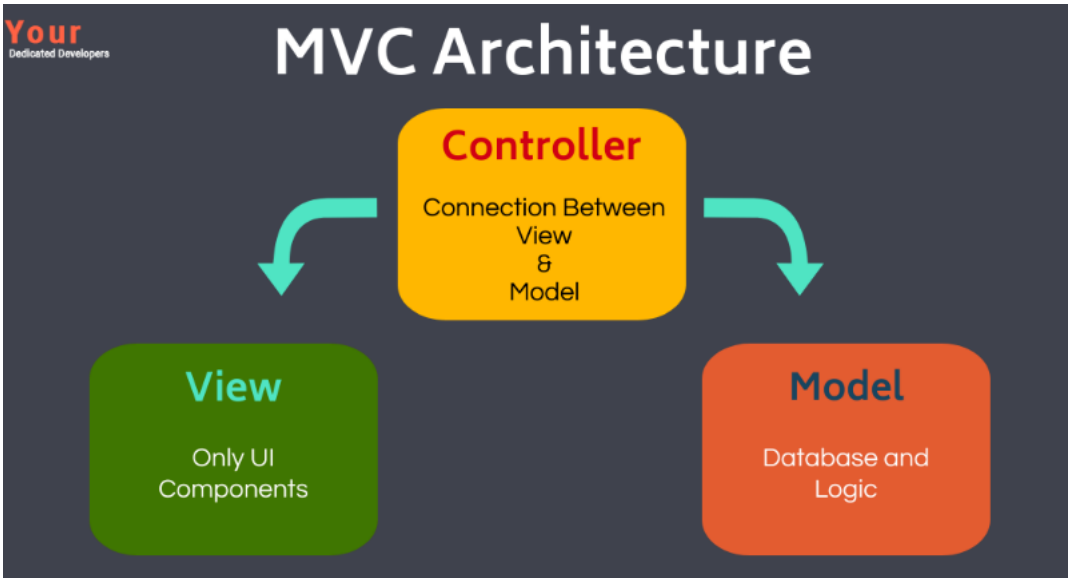


Figure 3

Source: https://yourdedicateddevelopers.com/wp-content/uploads/2020/01/twitter_41833911-16.png

The building blocks of an Angular application are as follows:

6.1.1 Components:

A component is a Javascript class that contains the application logic and data. A component is a term that refers to a section of the user interface (UI).A component encompasses the Data, HTML Template, and Logic in a view in simple terms. Every angular app will, in most cases, have at least one component. That's app.component.ts. A component is a Javascript class that has the @Component decorator applied to it. The component decorator accepts parameters such as selector for defining the component custom tag, which allows us to render that component wherever in our application, as well as template, templateUrl, styles, and stylesUrl. Angular components aren't meant to capture and save data; instead, they're meant to represent and give services access to that data.

While simultaneously tidying up the code, Angular Services makes it easy to connect business logic with app UI.A programmer only needs to import a service once in their code and then utilise it whenever they need it As a result, because there is less code to write, a specialist may be able to work more rapidly.

6.1.2 Modules:

In Angular, a module is a file that groups together related components, directives, pipes, and services to do or perform a specific activity. The Angular-based app can be considered as a puzzle where each module is needed to be able to see the full picture . There are a number of ways to add different elements to a module. Every angular application needs at least one root module, which is usually app.module.ts. Module, like component, is a javascript class that is marked with the @NgModule

decorator. Declaratives, imports, providers, and other parameters are passed to the NgModule decorator.

6.1.3 Directives:

Directives are the most difficult Angular functionality to master. In Angular, directives allow user to add dynamic content to DOM elements. Two types of directives are,

- **Structural Directives –**

Structural directives modify the structure of DOM elements, such as the existence or removal of an element from the DOM. These directives are added along with * (asterisk).Structural directives include *ngIf, *ngFor, ngSwitch, and *ngSwitchCase.

- **Attribute Directives –**

Attribute directives change the look or behaviour of an existing element in the Document Object Model (DOM).Attribute directives include ngClass, ngStyle, ngModel, formGroup, and formControlName.

6.1.4 Templates:

The Angular template uses Angular markup in conjunction with HTML to alter HTML components before they are shown. Data binding can be divided into two categories:

Event binding allows your app to update its data in response to user interaction in the target environment.

Users can interpolate values computed from your application data into HTML using property binding.

6.1.5 Metadata:

Angular uses metadata to determine how to handle a class. It is used to decorate a class so that the anticipated behaviour of the class may be configured.

6.1.6 Services:

A service class is developed when you have data or logic that isn't related with the view but needs to be shared across components. The @Injectible decorator is always

connected with the class. This makes our Angular app more modular, as the service is injected and call the function in the service that generates HTTP requests, rather than writing the same code in multiple components.

6.1.7 Dependency Injection:

Angular's built-in dependency injection makes it easier for developers to create applications. It helps to improve an application's modularity and efficiency. It allows us to keep the component classes as simple as possible. It just asks about your dependencies. It doesn't pull data from a server, check user input, or log to the terminal directly. Instead, it delegated these responsibilities to the services. Angular enables easier some server-side services to the client-side part using dependency injection.

6.1.8 Angular CLI (Command Line Interface):

Angular CLI provides a collection of useful coding tools which makes the developer's job easier. With notable

built-in capabilities like SCSS support and routing, the Angular CLI follows industry-best-practices for frontend development. Furthermore, the standard Angular CLI, such as ng-new or ng-add, makes it simple for developers to find ready-made features. The Angular CLI is as follows:

- ng New:

It's a primary initial for any Angular app development. and user can also create a new workplace with this command.

- ng Generate:

It allows you to create new components, services, routes, and pipes with ng Generate. One can also use ng Generate to construct simple test shells.

- ng Serve:

This command allows us to run your Angular app on a local server to test it.

- Test, Lint:

ng lint assists in the execution of applications in order to examine code for probable problems.

6.1.9 CDK and Angular Material:

With each version release, Angular as a leading frontend language has improved its Component Development Kit (CDK). The current version of the Angular CDK includes capabilities like refreshing and virtual scrolling. It aids in the dynamic loading and unloading of the DOM, allowing for the creation of a big list of high-performing data. The ScrollingModule and DragDropModule can both be imported into the application.

7. Methodology

When an Angular application runs Ng Modules are loaded by default,whether or not they are required instantly.To increase efficiency in large applications with multiple

features, a design style called lazy loading is used, which loads NgModules as per the requirement. It helps in the reduction of the initial bundle size, which in turn reduces the load time.Lazy Loading can be achieved by creating a feature module and configuring the route.

7.1 Creating feature module with routing-

- When a feature module is created it's reference is not declared in the application root module file. Instead it adds the declared route to the **routes** array declared in the module.
- Routing allows user to navigate from one part of an application to another part or from one view to another view.

The three important parts of angular routing are:

- The router outlet
- Routes and paths
- Navigation

Steps related to angular routing are:

- ❖ Installing the @angular/router package
- ❖ Setting the base location in index.html
- ❖ Adding dependencies to app.module.ts
- ❖ Configuring RouterModule
- ❖ Loading Components
- ❖ Creating Routes
- ❖ Navigation

Let's see a practical implementation of Lazy Loaded Modules-

The commands listed below are required to generate lazy loaded modules,

ng generate module modulea --route a –module app.module

ng generate module moduleb –route b –module app.module

Two folders named modulea and moduleb will be created as a result of the commands. Module.ts, routing.ts, and component files will be found in each folder.The following code for routes can be found in app-routing.module.ts file:

```
const routes: Routes = [
        {
         path: ‘a’, loadChildren: () =>
        import('./module/modulea.module').then(m
        => m.ModuleaModule)
        },
        {
         path: 'b', loadChildren: () =>
        import('./module/moduleb.module').then(m
        => m.ModulebModule)
        },
];
```

It means that when route a or b is accessed, their corresponding modules should be loaded slowly.

- User will be directed to page an after clicking the Load Module A button.

- A screen stating that m moduleb works! After clicking on Load Module B.

  The following steps are needed to see Lazy Loading is working or not,

  - Open the developer tools by pressing F12 to ensure that the files have loaded.
  - After that, as shown in the screenshot below, visit to the Network tab. When user refresh the page, a few files that were requested will appear.
  - Clear user's requests list by using the clear button
  - When user select Load Module A, a request for modulea-modulea- module.js appears, as shown in the screenshot below. This confirms that Module A was loaded slowly.

- The moduleb-moduleb-module.js file is loaded when you click Load Module B. This confirms that Module B was loaded lazily.
- When user try to click the buttons now, these js file will not load.

7.2 ROUTING IN ANGULAR

Angular Router is a core part of the angular platform.It enables developers to create multi-view Single Page Application and navigation between these views.The concepts related to routing are-

7.2.1 THE ROUTER-OUTLET :

The Router-Outlet is a router library directive that allows the Router to insert a component that is matched based on the current browser's URL. Multiple outlets can be added to your Angular application, allowing you to perform advanced

routing situations.

<router-outlet></router-outlet>

Any component that the Router matches will become a sibling of the Router outlet.

7.2.2 ROUTES AND PATHS :

Routes are object definitions that have at least a path and a component (or a redirectTo path) property. Component refers to the Angular component that needs to be associated with a path, and path refers to the section of the URL that identifies a unique view that should be presented. The Router can lead the user to a certain view based on a route specification that are supplied (through a static RouterModule.forRoot(routes) method). Each Route associates a component with a URL path. The path might be empty, indicating the default path of an application and it's usually the start of the application.

A wildcard string (**) can be used in the path. If the requested URL does not match any of the defined routes, the router will choose this route. For displaying a “Not Found” view or redirecting to a specific view this can be used if no match is found.

An example of a route is as follows:

```
{

path: ’contacts’,

component: ContactListComponent

}
```

If the Router configuration has this route definition,

When the browser URL for the web application becomes /contacts,the router will render ContactListComponent.

7.2.3 ROUTE MATCHING STRATEGIES :

Different route matching strategies are available with the Angular Router. The default technique is to see if the path is prefixed to the URL in the current browser.

For example the previous route:

```
{
    path: 'contacts',
    component: ContactListComponent
}
```

Could be also written as:

```
{
    path: 'contacts',
    pathMatch:'prefix',
    component:ContactListComponent

}
```

The patchMath attribute specifies the matching strategy. It's Prefix is the default in this scenario.The second matching strategy is **full**. When it's defined for a route, the router checks if the path is exactly the same as the current browser's URL:

```
{
    path: 'contacts',
    pathMatch:'full',
    component: ContactListComponent
}
```

7.2.4 ROUTE PARAMS:

In webapplications, creating routes with parameters is a frequent feature..

Angular Router allows you to access parameters in different ways:

- Using the ActivatedRoute service,
- Using the ParamMap observable available starting with v4.

The colon syntax can be used to construct a route parameter. This is a route with an id parameter as an example:

```
{
        path:'contacts/:id',
        component:ContactDetailComponenet
}
```

7.2.5 ROUTE GUARDS :

The Angular router has a route guard feature that allows developers to run some logic when a route is requested, and It enables or denies the user access to the route based on that logic, it allows or denies the user access to the route. It's typically used to see if a user is logged in and has the

appropriate permissions. A person first obtain authorization before accessing a page. You can add a route guard by implementing the CanActive interface which is available from the @angular/router package and extends the canActivate() method which holds the logic to allow or deny access to the route.The following guard, for example, will always allow access to a route:

```
class MyGuard implements CanActivate
{
        canActivate()
        {
            return true;
        }
}
```

One can then protect a route with the guard using the canActivate attribute:

```
{

        path:'contacts/:id,
```

```
        canActive:[MyGuard],

        component:ContactDetailComponent

    }
```

7.2.6 NAVIGATION DIRECTIVE :

The routerLink directive is provided by the Angular Router to construct navigation links. This directive uses the component's related path to navigate to.For example:

```
<a [routerLink]="'/contacts"'>Contacts</a>
```

7.2.7 MULTIPLE OUTLETS AND AUXILIARY ROUTES :

In the same application, Angular Router allows numerous outlets. A component has primary route and auxiliary routes.Auxiliary routes enable developers to navigate multiple routes at the same time.

To create an auxiliary route, user need a named router outlet in which the component that associated with the auxiliary route will be displayed.

```
<router-outlet></router-outlet>

<router-outlet name="outlet1"></router-outlet>
```

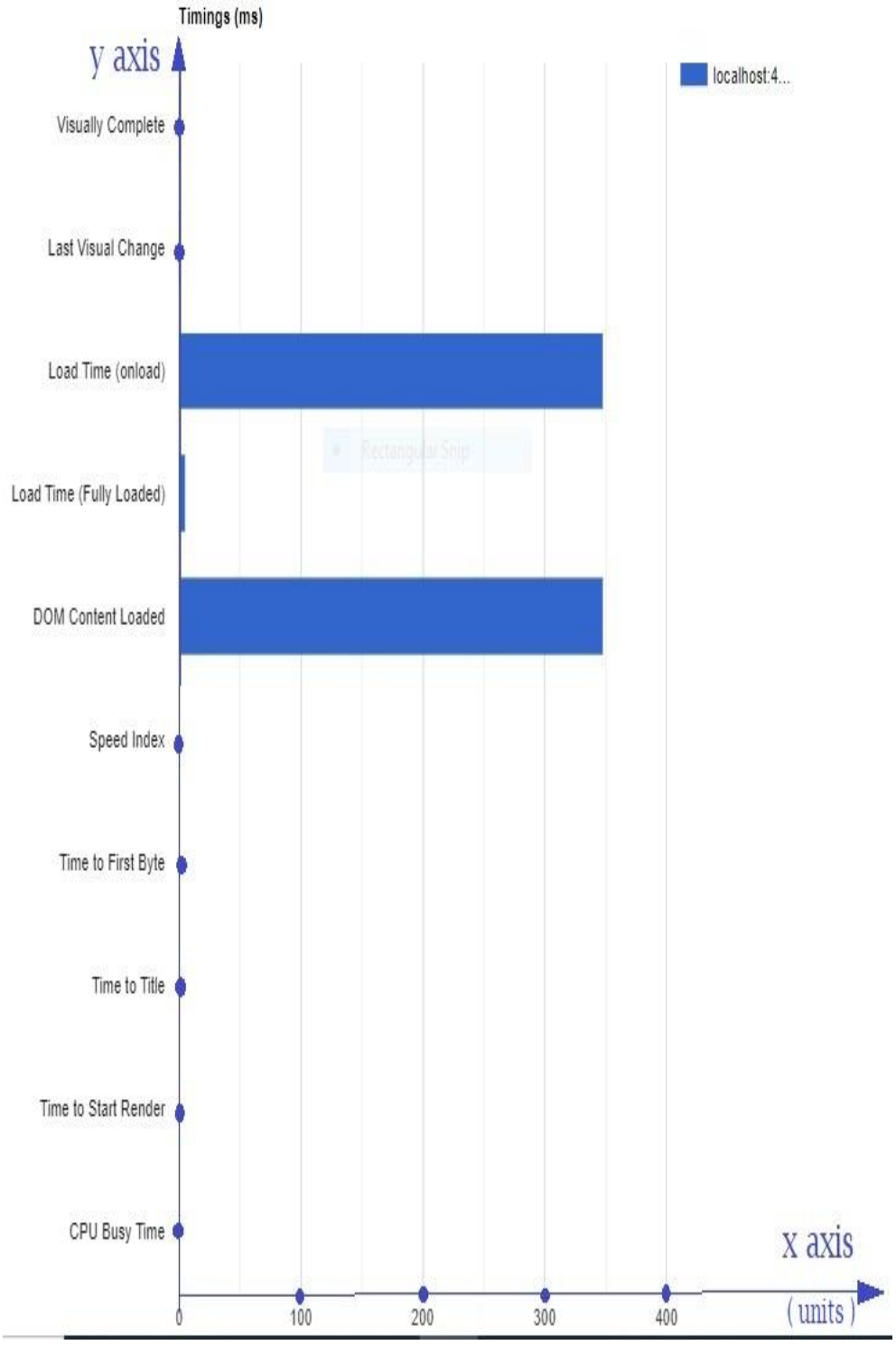
- The primary outlet is the one with no name.
- Except for the primary outlet, all outlets should be named.

Using the outlet attribute, one can define the outlet where user want the component to be rendered:

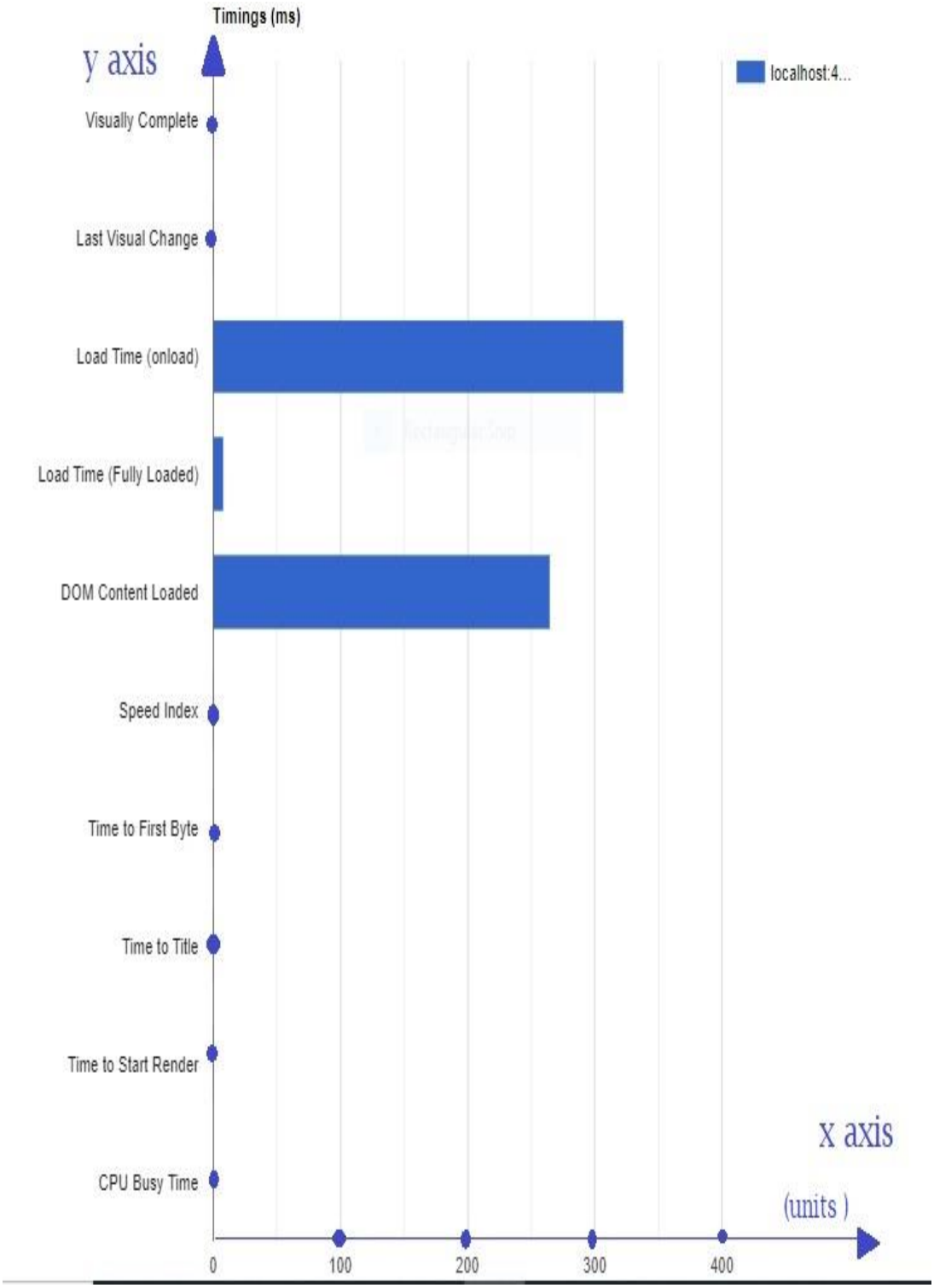
```
{

    path: "contacts",

    component: ContactListComponent,

    outlet: "outlet1"

}
```

8. Comparative Analysis

A simple survey is done to show the importance of routing in angular while creating user-friendly,complex,efficient web application.How routing can make a significant impact on an application performance is shown here.The load time of an application with routing is much less compare to an application that does not use routing.Below are the screenshots attached in support of the above analysis done on two web application,Education Application(without routing),Student Application(with routing).



Education Application



Student Application

9.Advantages

Custom Components

Users can use Angular to create their own components that pack functionality and rendering logic into reusable chunks. It also works well with web components.

Browser Compatibility

Users can use Angular to develop their own components, which are reusable bits of functionality and rendering logic. It also works well with web components.

Improved Speed and Performance

Angular features like template syntax, Angular CLI, routers, and others make programming easier and enable for faster application loading. The framework is compatible with a range of back-end programming languages in order to efficiently display obtained data in the UI.

Lazy-Load Modules

Modules are logical divisions between business components. Lazy-load modules can be used in large Angular projects to display different app components depending on where the user is in the app. This feature improves the performance of such

apps by lowering the size of the programme when it is first loaded.

Google Long-Term Support

Angular is Google-supported, therefore it's a reliable, trustworthy tool that'll most likely stay up with Google's infrequent changes and announcements. When it is combined with a lot of extensive documentation, then a solid structure is obtained that is backed by a lot of useful information and answers to frequently asked questions.

Unit-test friendly

Unit tests, or quality assurance techniques targeted at validating the performance of the application's smallest elements, or units, are made easier due to the independent nature of components.

Excellent Material Design Library

Angular Material is a library that allows user to use Material Design features in applications. Google created Material Design as a design foundation for creating highly responsive and productive user experiences. Programmers prefer Angular Material because it makes incorporating Angular design features in subsequent projects easier and faster.

10.Shortcomings of Angular

Limited SEO options

The platform's restricted SEO choices and low accessibility for search engine crawlers are two main disadvantages of using Angular. Given that Google is the most popular search engine on the planet, this issue is not expected to last much longer before a solution is offered in a future update.

Steep learning curve

While Angular is fantastic, it might not be appropriate for a total novice. Even if you've worked with HTML, CSS, and JS before, the steep learning curve may make user feel uneasy. It brings its own ideas and philosophy to the table, which user must accommodate. Because Angular is an opinionated framework, it has its own set of rules that developers must learn and follow. This can be both a good and a terrible thing, but for complete beginners, the learning curve can be difficult.

CLI documentation is lacking details

The present status of CLI documentation has some developers concerned. While the command line is really beneficial for Angular developers, there isn't enough information in their

official documentation on GitHub, so user will have to spend more time scouring GitHub forums for solutions.

11.Conclusion and Future Scope

The research work is successfully implemented by the proposed approach. By comparing, the research work with the existing work, the present worker find that the research work is giving the better result than the existing work. Currently, the Angular framework is powering market-leading commercial applications. This framework is chosen for its significant features and capabilities in the development of complex, customizable, modern, responsive, and user-friendly online applications.

The primary goal of this paper is to identify the reason how this framework suits to build an education application in Angular. Angular has a lot of features (modular structure, component architecture) which is very advantageous in web development. Because of it's flexibility, modularity and upgradation and amendment of new feature with every new release it will be widely used front-end-framework for building complex web and mobile application.Although learning Angular takes longer than learning other similar frameworks, but it is the best practise and the future of online and mobile applications.

12.References

1. Mohamed Sultan ,” Angular and the Trending Frameworks of Mobile and Web-based Platform Technologies : A Comparative Analysis ”, Future Technologies Conference(FTC)2017 29-30 November 2017|Vancouver,Canada.
2. Ivanova, Slavina, and Georgi Georgiev. "Using modern web frameworks when developing an education application: a practical approach." In 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1485-1491. IEEE, 2019.
3. Delcev, Sanja, and Drazen Draskovic. "Modern JavaScript frameworks: A survey study." In *2018 Zooming Innovation in Consumer Technologies Conference (ZINC)*, pp. 106-109. IEEE, 2018.
4. M.A.Jadhav , B.R.Saswant and A.Deshmukh , ” Single page application using AngularJS ”,.International Journal Of Computer Science and Information Technologies,6(3),pp.2876-2879,2015.
5. Stefanović, Miladin. "DEVELOPMENT OF WEB BASED APPLICATION USING SPA ARCHITECTURE."

6. Lassen, Anders. "JavaScript Frameworks A qualitative evaluation and comparison of the dominant factors in Angular and React Abdul Kadir Yorulmaz." (2020).
7. Jesse Koivukoski,"Reducing the loading time of a single-page web application." (2021).
8. K. N. Ard, "Single page architecture as basis for web applications", Master's thesis, 2015.
9. S.A.Mousavi," Maintainability Evaluation of SinglePage Application," Ph.D. dissertation,2016.
10. Official website and documentation athttps://angular.io/docs.
11. Google, "Lazy Loading." [Online]. Available: https://javascript.plainenglish.io/how-lazy-loading-impacts-angular-app-performance-9c1ee0cb11ea
12. Fortunato, David, and Jorge Bernardino. "Progressive web apps: An alternative to the native mobile Apps." In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1-6. IEEE, 2018.
13. Anirudh Bhaskar, Manjunath A,"An Interpretation and Anatomization of Angular: A Google Web Framework".
14. Google, https://danielk.tech/home/complete-angular-performance-guide#:~:text=Angular%20load%20performance%20refer

s%20to,of%20performance%20is%20runtime%20performance.

15. Filip Rysavy, Tomas Cerny and Martin Tomasek," Aspect-Oriented User Interfaces Design Integration to Angular 2 Framework,"In 2016 6th International Conference on IT Convergence and Security (ICITCS), pp. 1-3. IEEE, 2016.
16. According to https://blog.cloudboost.io/angular-faster-performance-and-better-user-experience-with-lazy-loading-a4f323b2cf4a
17. Referred https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa/
18. Official website and documentation at http://angularjs.org/
19. Official website and documentation at https://www.webcomponents.org/introduction
20. A. Pano, D. Graziotin and P. Abrahamsson, "What leads developers towards the choice of a JavaScript framework?", arXiv preprint arXiv:1605.04303, 2016
21. Github, "GitHub." [Online]. Available: https://github.com/
22. Miss. Vaibhavi A. Dhopate, Dr. Rahul J. Jadhav,"Role of Angular In Web Development", International Journal of

Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.8, Issue 6, page no.d783-d785, June-2021

23. Google,https://www.tutorialspoint.com/angular8/angular8_routing_and_navigation.htm

24. Referred https://codeburst.io/angular-spas-168a94a0959a

25. Miguel Ramos, Marco Tulio Valente and Ricardo Terra, AngularJS Performance A Survey Study.

13.Appendix

Student Application

- **student-routing.module.ts**

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AddStudentComponent } from './add-student/add-student.component';
import { DeleteStudentComponent } from './delete-student/delete-student.component';
import { LoginComponent } from './login/login.component';
import { SignupComponent } from './signup/signup.component';
import { StudentComponent } from './student.component';
import { UpdateStudentComponent } from './update-student/update-student.component';
import { ViewStudentComponent } from './view-student/view-student.component';
const routes: Routes = [
  { path: '', component: StudentComponent },
  { path: 'student', component: StudentComponent },
  { path: 'add-student', component: AddStudentComponent },
  { path: 'delete-student', component: DeleteStudentComponent },
  { path: 'update-student', component: UpdateStudentComponent },
  { path: 'view-student', component: ViewStudentComponent },
  {path: 'login', component:LoginComponent},
  {path:'signup', component:SignupComponent}
];
```

```
@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class StudentRoutingModule { }
```

- **app-routing.module.ts**

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [{ path: 'student', loadChildren: () =>
import('./student/student.module').then(m => m.StudentModule) }];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

- **Student.component.html**

```
<!-- Header - set the background image for the header in the line below-
->
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
   <div class="container">
      <a class="navbar-brand" >Student Dashboard</a>
```

```
        <button class="navbar-toggler" type="button" data-bs-
        toggle="collapse" data-bs-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent" aria-expanded="false"
        aria-label="Toggle navigation"><span class="navbar-toggler-
        icon"></span>

        </button>

      <div class="collapse navbar-collapse"
id="navbarSupportedContent">

        <ul class="navbar-nav ms-auto mb-2 mb-lg-0">

        <li class="nav-item"><a class="nav-link" *ngIf="isVisible"
href="#!">Home</a></li>

        <li class="nav-item"><a class="nav-link" *ngIf="isVisible" aria-
        current="page"  routerLink="/student/add-student">Add new
        Student</a></li>

        <li class="nav-item"><a class="nav-link" *ngIf="isVisible"
        routerLink="/student/view-student">View Student
        Enrolled</a></li>

        <li class="nav-item"><a class="nav-link" *ngIf="!isVisible"
        href="/signup">Sign up</a></li>

        <li class="nav-item"><a class="nav-link" *ngIf="!isVisible"
        href="/login">Login</a></li>

          </ul>

      </div>

   </div>

 </nav>

<header class="py-5 bg-image-full bg-img w-40" >

   <div class="text-center my-5">

      <h1 class="text-black fs-3 fw-bolder mb-5 pb-5">Education
System</h1>

      <p class="text-white-50 mb-20 pt-20 mt-20">
```

```
        A collection of institutions make up the education system
        (ministries of education, local educational authorities, teacher
        training institutions, schools, universities, etc.)       </p>
    </div>
  </header>
  <!-- Content section-->
  <section class="py-5">
   <div class="container my-5">
     <div class="row justify-content-center">
       <div class="col-lg-6">
         <h2>Education System</h2>
         <p class="lead">
        The word "education system" typically refers to public schooling,
        not private schools, and more frequently to K–12 curriculum. The
        smallest acknowledged "education system" is often a school or
        school district, whereas countries are the largest. States are
        thought of as having educational systems.
         </p>
        </div>
     </div>
   </div>
  </section>
```

- **add-student component.html**

```
<div class="bg-img ">
   <form class="form" [formGroup]="StudentForm">
     <div class="mb-3">
       <label for="FirstName" class="form-label">First Name</label>
```

```
        <input type="text" formControlName="FirstName" class="form-
control" id="FirstName" >
    </div>
    <div class="mb-3">
        <label for="LastName" class="form-label">Last Name</label>
        <input type="text" formControlName="LastName" class="form-
control" id="LastName" >
    </div>
    <div class="mb-3">
        <label for="Address" class="form-label">Address</label>
        <input type="textarea" formControlName="Address"
class="form-control" id="Address" >
    </div>
    <div class="mb-3">
        <label for="Email" class="form-label">Email address</label>
        <input type="email" formControlName="Email" class="form-
control" id="Email">
    </div>
    <div class="mb-3">
        <label for="MobileNumber" class="form-label">Mobile
Number</label>
        <input type="number" formControlName="MobileNumber"
class="form-control" id="MobileNumber" >
    </div>
    <div class="modal-footer">
        <button type="button" class="btn btn-primary"
*ngIf="addButton" (click)="postStudentDetails()">Add</button>
        <button type="button" class="btn btn-primary"
*ngIf="editButton" (click)="updateStudent()">Edit</button>
```

```
        <button type="button" class="btn btn-secondary"
routerLink="/">Close</button>
      </div>
   </form>
</div>
```

- **add-student.component.ts**

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup } from '@angular/forms';
import { ActivatedRoute, NavigationStart, Router } from
'@angular/router';
import { StudentService } from 'src/app/shared/student.service';
import { Student } from 'src/app/Student.model';

@Component({
  selector: 'app-add-student',
  templateUrl: './add-student.component.html',
  styleUrls: ['./add-student.component.css']
})
export class AddStudentComponent implements OnInit {
  StudentForm!:FormGroup;
  Student:Student= new Student();
  StudentData: any;
  editButton: boolean=false;
  addButton: boolean=true;
  id: any;
```

```
  constructor(private fb:FormBuilder,private
studentService:StudentService,private route:Router,private router:
ActivatedRoute) { }


  ngOnInit(): void {
    this.StudentForm = this.fb.group({
      FirstName: [''],
      LastName: [''],
      Address:  [''],
      Email: [''],
      MobileNumber: ['']
    });
    this.studentService.getStudent().subscribe((result:any) => {
      this.StudentData=result;
      console.log(result);
    })


  this.router.queryParams
      .subscribe(
        (params:any) => {
        console.log(params);

this.StudentForm.controls['FirstName'].setValue(params.FirstName);

this.StudentForm.controls['LastName'].setValue(params.LastName);
        this.StudentForm.controls['Address'].setValue(params.Address);
        this.StudentForm.controls['Email'].setValue(params.Email);
```

```
this.StudentForm.controls['MobileNumber'].setValue(params.MobileNu
mber);
      this.id=params.id;
      if(params.FirstName !== undefined ){
      this.editButton=true;
      this.addButton=false;
      }
    }
  );
 }

 postStudentDetails(){
  this.Student.FirstName=this.StudentForm.value.FirstName;
  this.Student.LastName=this.StudentForm.value.LastName;
  this.Student.Address=this.StudentForm.value.Address;
  this.Student.Email=this.StudentForm.value.Email;
  this.Student.MobileNumber=this.StudentForm.value.MobileNumber;

  this.studentService.postStudent(this.Student).subscribe((result:any)
=> {
    console.log(result);
    alert("Student added sucessfully");
    this.StudentForm.reset();
    this.route.navigate(['/student/view-student']);
  })
 }
```

```
 updateStudent(){
   this.Student.FirstName=this.StudentForm.value.FirstName;
   this.Student.LastName=this.StudentForm.value.LastName;
   this.Student.Address=this.StudentForm.value.Address;
   this.Student.Email=this.StudentForm.value.Email;
   this.Student.MobileNumber=this.StudentForm.value.MobileNumber;

this.studentService.updateStudent(this.Student,this.id).subscribe((resul
t:any) => {
     console.log(result);
     alert("Student updated sucessfully");
     this.route.navigate(['/student/view-student']);
   })
  }
}
```

- **delete-student component.html**

```
<p>delete-student works!</p>
```

- **delete-student.component.ts**

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-delete-student',
  templateUrl: './delete-student.component.html',
  styleUrls: ['./delete-student.component.css']
})
export class DeleteStudentComponent implements OnInit {
```

```
  constructor() { }

  ngOnInit(): void {
  }

}
```

- **update-student component.html**

```
<p>update-student works!</p>
```

- **update-student.component.ts**

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-update-student',
  templateUrl: './update-student.component.html',
  styleUrls: ['./update-student.component.css']
})
export class UpdateStudentComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Educational Application

- **education-dashboard.component.html**

```
<nav class="navbar navbar-expand-lg navbar-dark bg-primary mb-5">
 <div class="container-fluid">
  <a class="navbar-brand" href="#">
    <img src="assets/global-education-system-2502190-2094337.png" alt="" width="30" height="24" class="d-inline-block align-text-top">
    Navbar
  </a>
  <button type="button" class="btn btn-bd-download ml-auto" data-bs-toggle="modal" data-bs-target="#exampleModal">
     Add Student
    </button>
    <!-- <button routerLink="/login" type="button" class="btn btn-bd-download">Logout</button> -->
 </div>
</nav>

 <div class="container w-60">
    <div class="container w-20 mb-5">
    <div id="carouselExampleSlidesOnly" class="carousel slide " data-bs-ride="carousel">
       <div class="carousel-inner">
        <div class="carousel-item active">
          <img src="assets/Effective-education-system-—-A-must-for-nation-building.jpg" class="d-flex justify-content-center w-100" alt="...">
        </div>
        <div class="carousel-item">
          <img src="assets/download.jpg" class="d-flex justify-content-center w-100" alt="...">
        </div>
        <div class="carousel-item">
```

```
          <img src="assets/education-system-1-638.jpg" class="d-
flex justify-content-center w-100" alt="...">
        </div>
       </div>
      </div>
     </div>

   <table class="table table-hover pt-5 mt-8">
   <thead class="bg-primary">
    <tr>
     <th scope="col">Roll No.</th>
     <th scope="col">First Name</th>
     <th scope="col">Last Name</th>
     <th scope="col">Address</th>
     <th scope="col">Email Id</th>
     <th scope="col">Mobile No.</th>
     <th></th>
    </tr>
   </thead>
   <tbody class= "table-primary ">
    <tr *ngFor="let Student of StudentData">
     <td >{{Student.id}}</td>
     <td>{{Student.FirstName}}</td>
     <td>{{Student.LastName}}</td>
     <td>{{Student.Address}}</td>
     <td>{{Student.Email}}</td>
     <td>{{Student.MobileNumber}}</td>
     <td>
        <button type="button" class="btn btn-warning" data-bs-
toggle="modal" data-bs-target="#exampleModal"
(click)="onEdit(Student)">Update</button>
        <button type="button" class="btn btn-danger mx-3"
(click)="deleteStudent(Student)">Delete</button>
       </td>
    </tr>
   </tbody>
   </table>
  </div>
```

```
 <!-- Modal -->
 <div class="modal fade" id="exampleModal" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
   <div class="modal-dialog">
    <div class="modal-content">
     <div class="modal-header">
      <h5 class="modal-title" id="exampleModalLabel">Modal
title</h5>
       <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
     </div>
     <div class="modal-body">
       <form [formGroup]="StudentForm">
         <div class="mb-3">
           <label for="FirstName" class="form-label">First
Name</label>
            <input type="text" formControlName="FirstName"
class="form-control" id="FirstName" >

          </div>
          <div class="mb-3">
            <label for="LastName" class="form-label">Last
Name</label>
             <input type="text" formControlName="LastName"
class="form-control" id="LastName" >

           </div>
           <div class="mb-3">
             <label for="Address" class="form-
label">Address</label>
              <input type="textarea"
formControlName="Address" class="form-control" id="Address"
>

             </div>
         <div class="mb-3">
          <label for="Email" class="form-label">Email
address</label>
```

```
                <input type="email" formControlName="Email"
class="form-control" id="Email">
              </div>
              <div class="mb-3">
                <label for="MobileNumber" class="form-
label">Mobile Number</label>
                <input type="mobilenumber"
formControlName="MobileNumber" class="form-control"
id="MobileNumber" >

              </div>
            </form>
          </div>
          <div class="modal-footer">
            <button type="button" id="cancel" class="btn btn-
secondary" data-bs-dismiss="modal">Close</button>
            <button type="button" class="btn btn-primary"
*ngIf="showAdd" (click)="postStudentDetails()">Add</button>
            <button type="button" class="btn btn-primary"
*ngIf="showEdit" (click)="updateStudent()">Edit</button>
          </div>
        </div>
      </div>
    </div>
```

- **education-dashboard.component.ts**

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Student } from '../Student.model';
import { EducationService } from '../shared/education.service';

@Component({
  selector: 'app-education-dashboard',
```

```
  templateUrl: './education-dashboard.component.html',
  styleUrls: ['./education-dashboard.component.css'],
})
export class EducationDashboardComponent implements OnInit {
  StudentForm!:FormGroup;
  Student:Student= new Student();
  StudentData: any;
  showAdd: boolean=true;
  showEdit: boolean=false;

  constructor(private fb:FormBuilder,private
educationService:EducationService) { }

  ngOnInit(): void {
    this.StudentForm = this.fb.group({
      FirstName: [''],
      LastName: [''],
      Address:  [''],
      Email: [''],
      MobileNumber: [null, [Validators.required, Validators.pattern('[-
+()0-9]+')]]
    });
    this.getStudentDetails();
  }
  postStudentDetails(){
    this.Student.FirstName=this.StudentForm.value.FirstName;
    this.Student.LastName=this.StudentForm.value.LastName;
```

```
    this.Student.Address=this.StudentForm.value.Address;
    this.Student.Email=this.StudentForm.value.Email;
    this.Student.MobileNumber=this.StudentForm.value.MobileNumber;


this.educationService.postStudent(this.Student).subscribe((result:any)
=> {
      console.log(result);
      alert("Student added sucessfully");
      let ref=document.getElementById('cancel');
      ref?.click();
      this.StudentForm.reset();
      window.location.reload();
    })
  }

  getStudentDetails(){
    this.Student.FirstName=this.StudentForm.value.FirstName;
    this.Student.LastName=this.StudentForm.value.LastName;
    this.Student.Address=this.StudentForm.value.Address;
    this.Student.Email=this.StudentForm.value.Email;
    this.Student.MobileNumber=this.StudentForm.value.MobileNumber;

    this.educationService.getStudent().subscribe((result:any) => {
      this.StudentData=result;
      console.log(result);
```

```
  })
 }
  deleteStudent(row : any){
    this.educationService.deleteStudent(row.id)
    .subscribe(res=>{
    alert("Student Deleted");
    window.location.reload();
  })
 }

 onEdit(row : any){
 this.showEdit=true;
 this.showAdd=false;
 this.StudentForm.controls['FirstName'].setValue(row.FirstName);
 this.StudentForm.controls['LastName'].setValue(row.LastName);
 this.StudentForm.controls['Address'].setValue(row.Address);
 this.StudentForm.controls['Email'].setValue(row.Email);

this.StudentForm.controls['MobileNumber'].setValue(row.MobileNumb
er);
 }

 updateStudent(){
 this.Student.FirstName=this.StudentForm.value.FirstName;
 this.Student.LastName=this.StudentForm.value.LastName;
 this.Student.Address=this.StudentForm.value.Address;
 this.Student.Email=this.StudentForm.value.Email;
```

```
this.Student.MobileNumber=this.StudentForm.value.MobileNum
ber;



this.educationService.updateStudent(this.Student,this.Student.id)
.subscribe((result:any) => {
   console.log(result);
   alert("Student updated sucessfully");
   let ref=document.getElementById('cancel');
   ref?.click();
   this.StudentForm.reset();
   window.location.reload();
 })
 }


}
```

- **app-routing.module.ts**

```
import { NgModule } from '@angular/core';

@NgModule({

imports:[],
exports:[]
})
export class AppRoutingModule { }
```