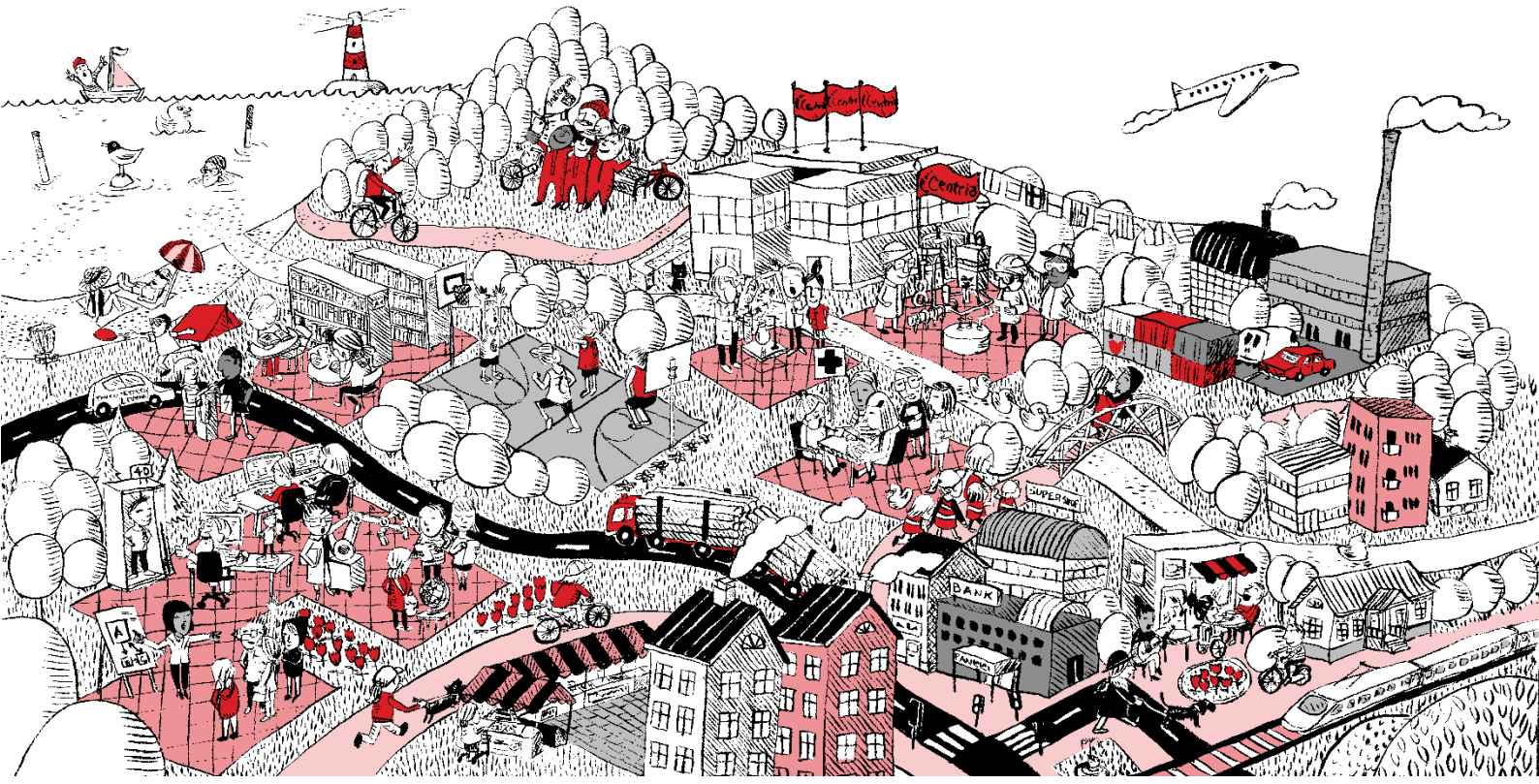
Quan Nguyen

IONIC FRAMEWORK AS STANDARD TO HANDLE WEB DESIGN EVOLUTION

Thesis
CENTRIA UNIVERSITY OF APPLIED SCIENCES
Degree Programme
June 2021



ABSTRACT

Centria University of Applied Sciences	**Date** June 2021	**Author** Quan Nguyen
Degree programme Bachelor of Information Technology		
Name of thesis IONIC FRAMEWORK AS STANDARD TO HANDLE WEB DESIGN EVOLUTION		
Centria supervisor Jari Isohanni		**Pages** 44 + 1

The aim of the thesis is to provide the history of web design and Ionic Framework as standard to catch up with modern design's requirements. The thesis consists of many technologies, from basic ones such as HTML and JavaScript to complex ones such as Angular and Typescript.

In modern days, organizations require efficient tools to develop applications that can scale as they grow. Taking this into consideration, three versions of 'To-do' app will be created and demonstrated in the thesis to show standard requirement for a full stack application.

The thesis comprises two parts. The first part will list the technologies and designs mostly used in each decade, and the second part will use Ionic Framework to show the design of many websites in those decades. For the front-end, HTML, CSS, Angular and JavaScript will be used while NodeJS and Firebase will handle the app's back-end. As a beginner guide, this thesis is for students or companies who only have basic web programming knowledge.

Key words
Ionic Framework, Angular, TypeScript, JavaScript, FireBase.

CONCEPT DEFINITIONS

List of Abbreviations

CDN	Content Delivery Network
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
DOM	Document Object Model
SQL	Structured Query Language
XML	Extensible Markup Language
CSRF	Cross-site Request Forgery
DDOS	Distributed Denial-of-Service
IDE	Integrated Development Environment
CDN	Content Delivery Network
API	Application Programming Interface
JSON	JavaScript Object Notation
CPU	Central Processing Unit
CLI	Command Line Interface
SRC	Source
HTTP	Hypertext Transfer Protocol
RAM	Random-access memory

ABSTRACT
CONCEPT DEFINITIONS
CONTENTS

1 INTRODUCTION

With the recent wave of Industry 4.0, technologies are growing with incredible speed and scalability. Since 2010, Internet has become a necessity which provides many tools and intermediary for all people around the world. Now, a person can play video games, make purchases, work online, watch streaming concerts with only a mobile phone and a good website to help them satisfy those needs. This creates many opportunities while requires standards and competitions for developers and organizations. There will always be aspects of web design that are never going away such as user experience, data security and fast load time. However, new features and elements will keep the site at the forefront of design and search engines, making scalability becomes a standard requirement for a front-end web developer (Lara, 2020). With this, companies must come up with creative improvements from time to time, without forgetting the stability and work efficiency in the web development process.

To show how the web design progresses over the time, this thesis also provides a study case of a simple ‘To-do list website’ that collects user input. In college education, students are commonly taught to use traditional HTML, CSS, and JavaScript. Although this does not require high experience in web development, it can quickly become inefficient when there is a need to update the website or collaborate between web developers. Therefore, the thesis will use Ionic Framework to create a showcase for designs in later years since it also works with the traditional languages listed above.

The purpose of the thesis is to demonstrate the progress of web design. The first chapter explains the changes in web design and detailed research about important technologies used in each era. The second chapter provides three study cases to demonstrate the growth in web designing using Ionic Framework. Finally, there will be a conclusion to decide if Ionic or many other Frameworks should be used as a requirement for web development or not.

2 EVOLUTION OF WEB PAGE

The Internet has progressed many times to be widely used today with the help of many updates and open-sources services, bringing more devices connected. In the past, web development was only known for the designing of web pages and websites on the intranet and internet. However, nowadays it is more like creating a web application that providing complex features for both businesses and customers. With the new era incoming, it is worth looking back at the history of web development to learn the rapid growth of technology and make preparation to catch up with modern improvements.

2.1 In the early 1990

In August 1991, the first website was published by Tim Berners-Lee containing a dozen links and basic HTML technologies. However, at that time, the available public web browser was not graphical but text only (Figure 1). In 1992, the first image was posted online, and in 1994, other graphical features were introduced, including GIF image, text formatting and table (TK, 2020).

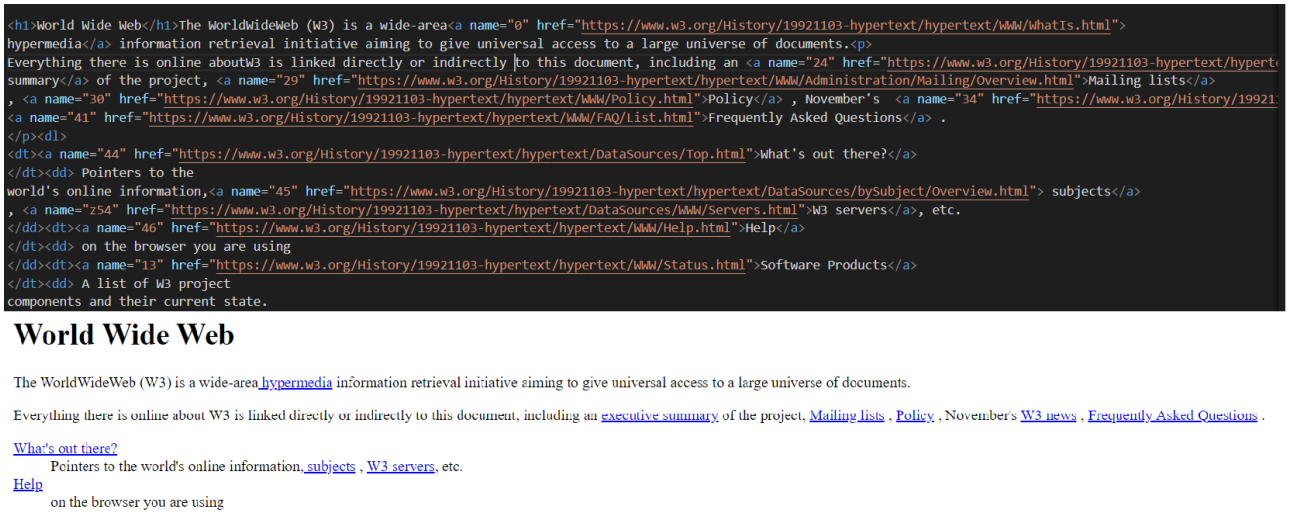


FIGURE 1. The First Website that only contains text and hyperlinks (CERN Info, 1991).

HTML stands for Hypertext Markup Language, which is a standard markup language to create websites. HTML files end with a `.html` or `.htm` extension and can be created to render the front-end of the website. With front-end, the user can have a visual view of the website to interact with them instead of lines of codes. Each HTML page consists of a set of elements on top of each other. Each element structures the website's content into a header, footer, paragraphs, heading, etc. To use them, they need

to be declared using <*tag*> syntax. If the element has content, an opening and closing tag will be needed. Additional attributes can also be placed to opening tags to give them more functions (Curtis, 2021). The figure below features a Heading element that change the font and size of content.

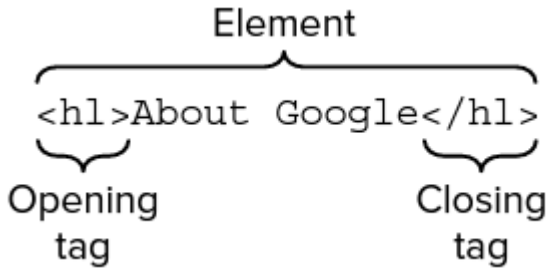


FIGURE 2. An element (Larsen, 2013)

In 1995, web hosting changed everything. World Wide Web was not only for the company that owned a computer or server but also for individuals or businesses that could not meet on the requirements of hosting a website themselves. Throughout the decades, there were many reliable services like Angel-Fire, GeoCity, Tripod, that offered 35kB, 1MB and 2MB respectively. At the end of decades, two unique types of web hosting service called Blogger and LiveJournal was launched which popularised blog-publishing format and changed the way people thought about personal website (MicroStartups, 2019).

2.2 In the 2000s

In 1996, CSS was released. However, it only gained popularity after the dot-com bubble in 2000, marking the whole change for website development. With that technology, HTML elements can have their look changed, results in a cleaner and semantic web layout (Figure 3). This gives the website a creative aspect that inspires many trends at that time, such as blogging or news.

FIGURE 3. Hewlett-Packard with CSS in 2000 (Hewlett Packard in 2000)

CSS stands for Cascading Style Sheets, a language to describe how HTML elements should be shown on screen. When the page is rendered, selectors in the CSS file will point to HTML elements. The CSS files will be loaded using <style> syntax, and a DOM tree (Document Object Model) will be created to appoint each property to a specific element. In order to be loaded, CSS files are required to have `.css` as their extension. The figure below demonstrates an example of using CSS. The selector will have a declaration block containing the style. Inside those blocks are CSS properties to define the name and value of the style which are separated by semicolons In Figure 4 below, the property 'change the colour to blue' will be given to all <*p*> elements in the HTML document.

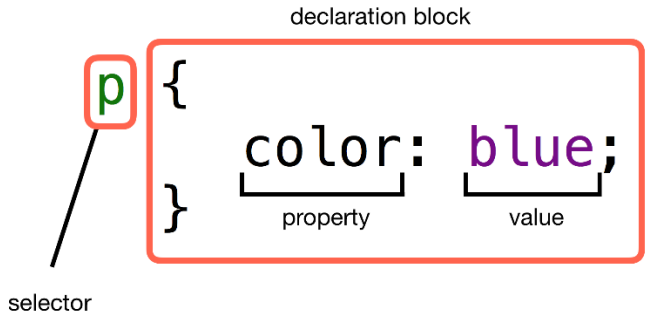


FIGURE 4. CSS property (Bhargava, 2018)

In this period, databases are widely used to collect, store and gain data accessibility for specific users. This data can be further analyzed or manipulated for many additional purposes. There are also two types of databases, Structured Query Language (SQL) and NoSQL. SQL databases define and manipulate data as tables using structured query language, making it versatile and popular. To make SQL databases, a schema must be predefined to determine the structure of data. On the other hand, the NoSQL database has a dynamic schema, allowing it to become document-oriented, column-oriented, graph-based, or organized as a KeyValue store (Sharma, 2020). While SQL is suitable for multi-row transactions applications, NoSQL is flexible and has better scalability.

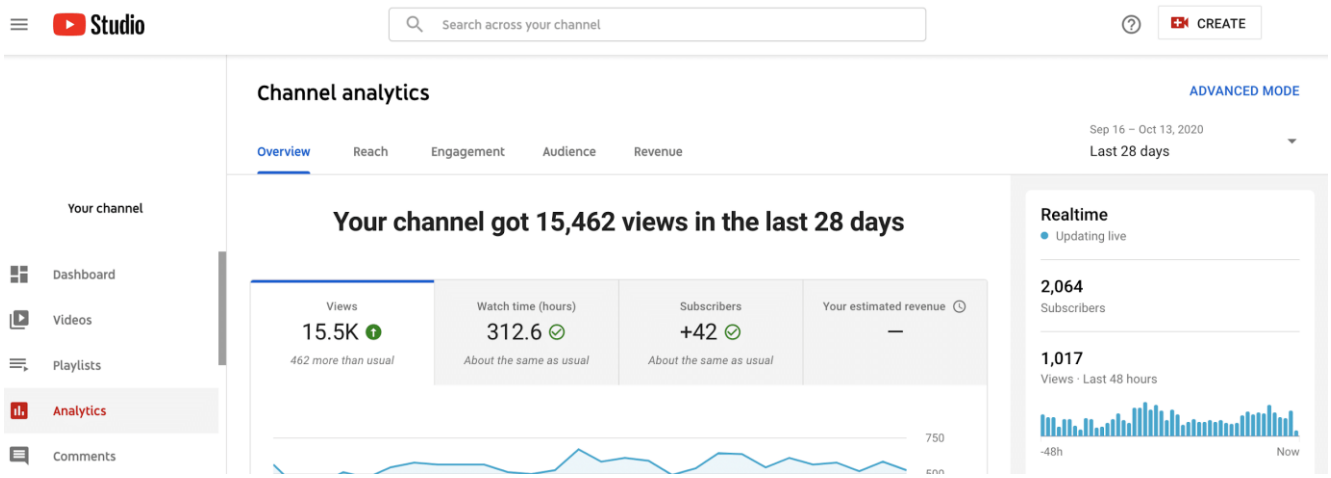


FIGURE 5. A channel use Youtube's database to check on data and analyze them (Fergus Baird, 2020)

In 2000s, JavaScript was widely used and can be implemented using files with `.js` extension. With the new technology, HTML pages have been sorted into hierarchical objects that can be accessed using any other language. JavaScript is a client-side language that allows the server to send the files to the user's computer machine when rendering. While HTML markup language allows web developers to format content, JavaScript makes webpages more dynamic by allowing users to interact with web

pages, click on the elements, and change the pages (Blumenthal, 2017). Figure 6 below demonstrates the hierarchical tree. At the root, there must be `<html>`, `<head>` and `<body>` objects to declare a container. Other element objects can be wrapped inside those containers to provide more containers for other usage, such as `<a>` container to declare a link, or `<title>` container to declare a title. Some containers also include text object to render those text on the screen.

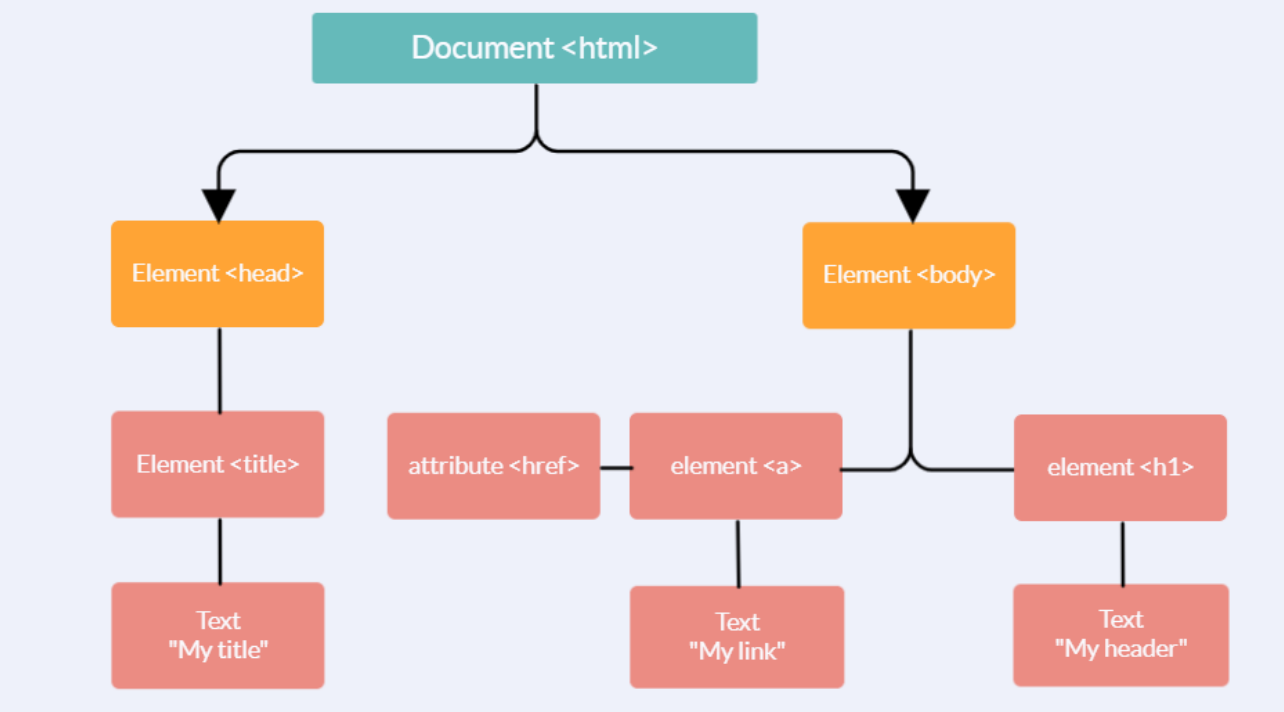


FIGURE 6. HTML DOM Tree of Object

2.3 In 2010s

2010 marks the time web design evolves out of the web. HTML5 and CSS3 opened the door for designing and developing dynamic, interactive, and most importantly, responsive websites. Instead of having two separate sites (one for mobile and one for desktop), website owners could have their pages completely optimized for many different devices (Brill, 2020). This allowed WordPress to start a tradition of building web using a different theme to express the creativity of the web programmers. One of the popular themes at that time was TwentyTen which provides a stylish, customizable, simple and readable layout (Wordpress.org, 2021).

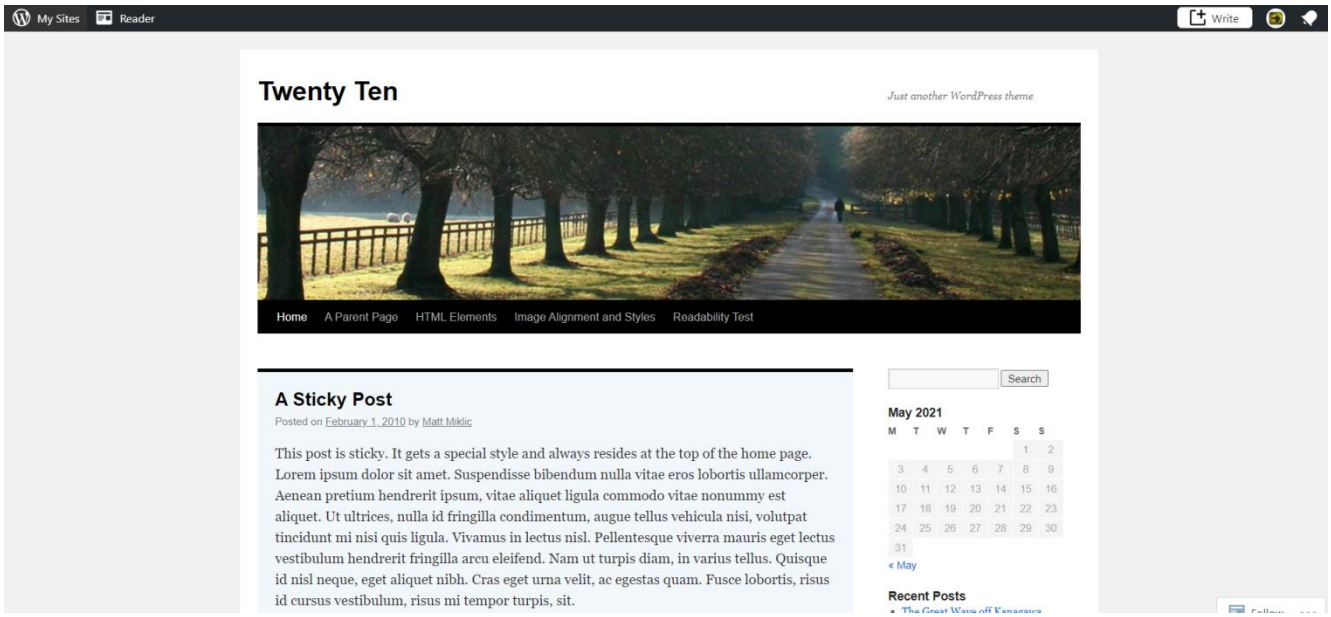


FIGURE 7. WordPress theme (Twenty Ten) with minimalistic design

In the past, API (Application Programming Interface) served as an interface that enables applications to communicate with each other, but this technology had more uses in the 2000s. This is important since API works well with Internet and opens its potential. Internal API can enhance the productivity of the development team by maximizing reusability and enforcing consistency in new applications. Public API allows third-party developers to enhance any company's services or bring in more customers. Using those improvements as the base, REST architecture was coined to distribute hypermedia systems (Patni, 2017). With this, applications and services can connect and exchange data.

A few years later, E-commerce became wildly popular. Since JavaScript provided the services but easily to be hacked with XML injection, the usage of this technology was questioned. This demanded website to have various services outside of its data content, including security and encryptions. One of the popular security features is CSRF Protection, which prevents the site from using HTTP requests to direct users to other servers for illegal actions. CSRF provides an Origin header to identify which website generate the API request (Barth, 2010).

In the middle of the 2010s, mobile phones became widely popular. For the first time, people viewed the website on mobile devices more than personal computers. This led to "mobile-first" development, which is a trend that companies create the product from the mobile end then expand its features to tablet or desktop version. Because of various smartphone designs, the website is put under restrictions like bandwidth, screen size, etc. and developers must head to a lean and neat product with prioritized features (Xia, 2017). Since then, websites and databases are coded inside a framework and then built

into deployment. Then, all the source codes will be configured and go through a publish process to transfer data from framework structure in development tools to hosting service, reducing cyber-attacks and DDOS (Figure 9).

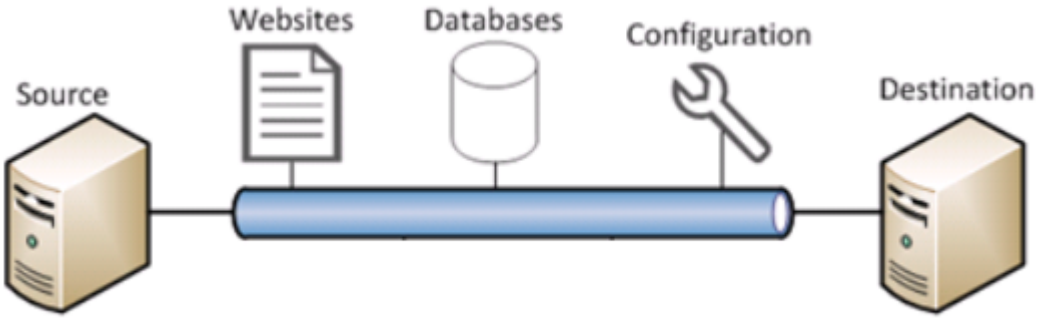


FIGURE 8. Web Deploy (Mittal, 2011)

With different services created in this era, many intellectual challenges have changed the web development process, and programmers are requested to solve problems and work with situations they have never faced before. Nowadays, web developers use framework technology to build, test, and optimize in an efficient, robust and versatile way. With these frameworks, the developers can focus on the high functionality of the product while the rest will be taken care of using the framework (Singh, 2020). One of the popular open-source frameworks is Ionic. Using hybrid application technology, HTML, CSS, and JavaScript can run on an internal browser (WebView) that is wrapped in a native app. JavaScript can access native APIs through the wrapper (Cheng, 2018).

Ionic framework is one of many powerful tools to build website or web app. Based on Web Components standards and flexibility, Ionic components are built as custom elements compatible with many other popular frameworks, including Angular, React and Vue. Furthermore, its UI components work and perform stably across different platforms (Cheng, 2018). Included inside Ionic is another framework called Angular which can be used to build single-page web applications using HTML and TypeScript. With Angular as Ionic's core, the project can be to be structured into building blocks. Each block can become components that provide many features, such as views or services.

3 CASE STUDY 1: TO DO LIST WITH HTML, CSS AND JAVASCRIPT

Three study cases are made to show the changes in website development in each era. The first case will focus on the requirements of a website at the end of the 20th century. During those years, developers only used HTML, CSS, JavaScript to make websites so the front page would be extremely generic without many features. Therefore, the main attraction of the websites were layouts and decorations made with CSS. During the development, the structure of the project and the codes will be clearly explained alongside many other coding techniques that many programmers frequently use. In the end of each case study, there will be a conclusion and a comparison to show how complex the web development can be after many years of evolution.

3.1 Visual Studio Code

To compile the codes, the programmers are required to install IDE. IDE stands for Integrated Development Environment, a software application that provides tools to make, execute, check the codes, etc. One of the commonly used IDE is Visual Studio Code (Figure 10), which can be downloaded and installed from Microsoft. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to ease the software development process. Moreover, there are more open-source features provided by Visual Studio Code using Extension.

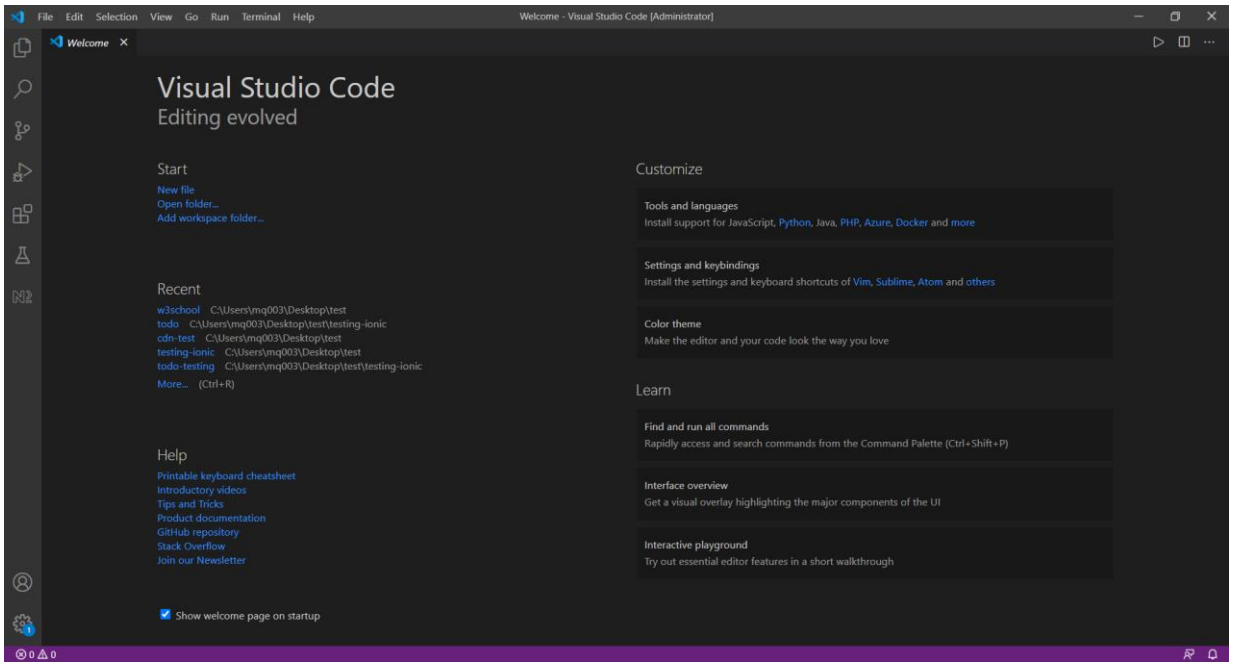


FIGURE 9. Visual Studio Code interface

Since JavaScript is also used in the project, the application can run synchronically. With this, when the codes are updated, the website will be updated as well. To do this, an extension called ‘Live Server’ needs to be installed to Visual Studio Code. Live Server is a web extension that open a local port to load elements and JavaScript feature for static and dynamic pages. The extension can be found in the Extension section of Visual Studio Code (Figure 11).

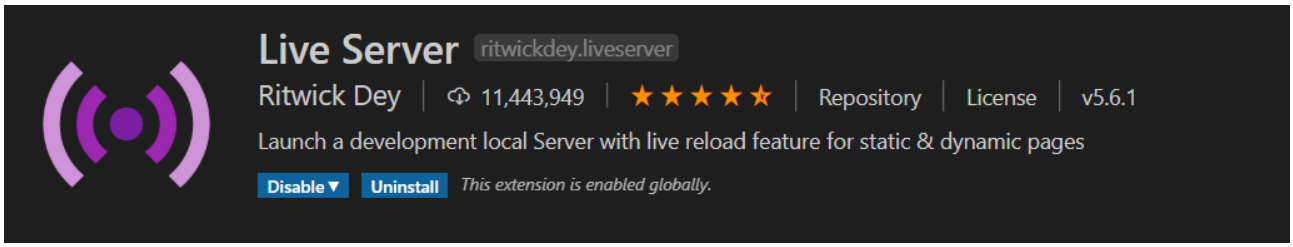


FIGURE 10. Live Server extension

Visual Studio Code also has its own directory structure to manage the codes. When the program runs, additional files will be added to provide additional configurations to the project. Figure 12 shows the structure generated by the IDE, including a `.vscode` file to control the compiler for debugging and the `.gitattributes` with `README.md` file to configure the Github to initialize the project to store all the codebase online. Since the first case study demonstrates traditional codes, it is not necessary to edit any of the files whose IDE generates.

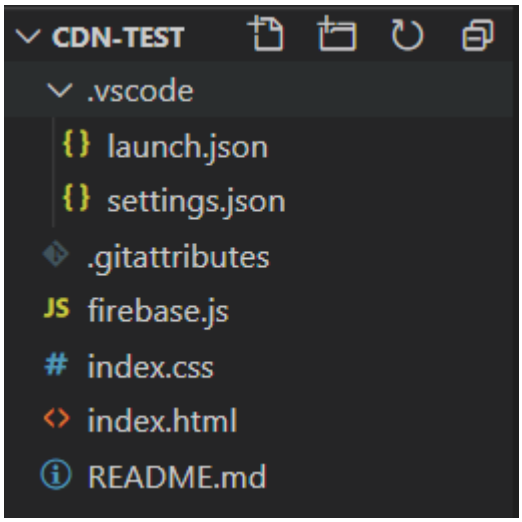


FIGURE 11. Visual Studio Code Structure

3.2 Adding the codes

The files can be created inside Visual Studio Codes using Ctrl + N. After that, the codes can be copied into those files and save with the correct extension. The codes added in this case study are a modified version of the one on W3School (W3School, 2021). For this thesis, many redundant aspects of the codes have been removed or modified to fit with the coding styles in 1990 while improving CSS design (Nguyen, 2021). After adding the codes, the programmer can run the project with Live Server to get the result in the following Figure.

FIGURE 12. Base To-do list app

To render this web page, the compiler reads the `.html` files and scales the website with the `<meta>` tag. Then the CSS is added to the documents with the `<link>` tag. The content of the page is declared in `<body>`. After rendering the content, the JavaScript file is loaded using the `<script>` tag to add controls to the web page.

In the HTML page, `<input>` is used to create a space to type, `<span>` to add the Add button and `<ul>` is to create the list of notes. In the CSS file, these elements are styled to fit in. In the JavaScript file, there is a new function and a new event. The additional function will create a new item in the list when the user presses the Add button. The additional event will occur when the user clicks on the item, which changes the selector of CSS.

Since 1996, there have been many web hosting services that allow websites to become available on the Internet, so website in this case study should be hosted online as well. Web hosting is the process of renting or buying servers to house a website on the World Wide Web. The files that make up a website are uploaded from a local computer to a web server. That server will allocate its resources (RAM, hard drive space, bandwidth, etc) to allow users to view the websites from anywhere in the world (What is Web Hosting, 2021). With this, the chance of inaccessible to the web due to hardware failure is reduced significantly. Although there are many web hosting services, this thesis will use InifinityFree since it is stable and does not require a fee. After creating an account, the website can be uploaded using Online File Manager. When the user goes to the site, the index.html and other imported files will be rendered as the main page. Although InfinityFree also has database services, this study case will not include that technology since they are not frequently used until many years later.

3.3 Summary of Study Case 1

Although the codes are compiled using advanced IDE and open-source extensions, the design structure is similar to those at the end of the 20th century. Furthermore, with HTML, CSS and JavaScript, it is possible to create a functional website. In this project, there are 23 lines in .html file, 66 lines in .css file and 25 lines in .js file. For a basic website that was used at the end of 1990, the most important part of the page mostly came from CSS file since the project utilizes that programming language to provide a good layout. At the end, a web service is used to host the website straightforwardly and provide a link to grant access for public uses.

4 CASE STUDY 2: TO DO LIST WITH IONIC CDN

There are two ways of adding a framework to the project: using CDN or recode in that framework. The former way can be used to maintain a simple, small project but the latter is necessary for big projects in companies or organizations. Therefore, the 2nd case study will be used to demonstrate how to integrate a framework into the project and the coding style of the 2000s. The project will also have a database as a standard requirement of any websites containing info in that time and use the same hosting service as the 1st case study (Nguyen, 2021).

4.1 CDN - Content Delivery Network

CDN is used to quickly transfer the assets needed to load the content of an Internet site, including HTML, JavaScript files, stylesheets, images, and videos (Cloudflare, 2009). When a user access from a webpage, In this case, the Ionic CDN is added to import Ionic. These contents include theme, image, components, etc. stored on a network server. Since these servers are linked together, the website can be loaded very quickly compared to being recorded in that framework (Figure 14) . Since CDN can be used to import most of the assets, they can be used to increase the scale of it or reduce the number of codes required without modifying old codes. For example, Bootstrap CDN can be imported to reduce the number of CSS lines for a button or use Bootstrap's button.

FIGURE 13. Server storing assets are placed around the world for fast accessibility (Cloudflare, 2009)

4.2 Adding Ionic components to project

In the second era, many themes are available so many websites reuse those themes to have a suitable layout. Therefore, in this case study, the project will utilize the import function to load the assets and CDN of the framework. Since Ionic team already packaged their framework into many files, the JavaScript and CSS files can be loaded using <script> and <link> tag respectively. The figure below provides the necessary codes for the importing.

```
<script type="module" src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionic/ionic.esm.js"></script>
<script nomodule src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionic/ionic.js"></script>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@ionic/core/css/ionic.bundle.css"/>
```

FIGURE 14. Ionic CDN import codes

```
<ion-input type="text" placeholder="Title..." id='myInput'></ion-input>
<ion-button color="light" expand="block" onclick="newElement()" >Add</ion-button>
```

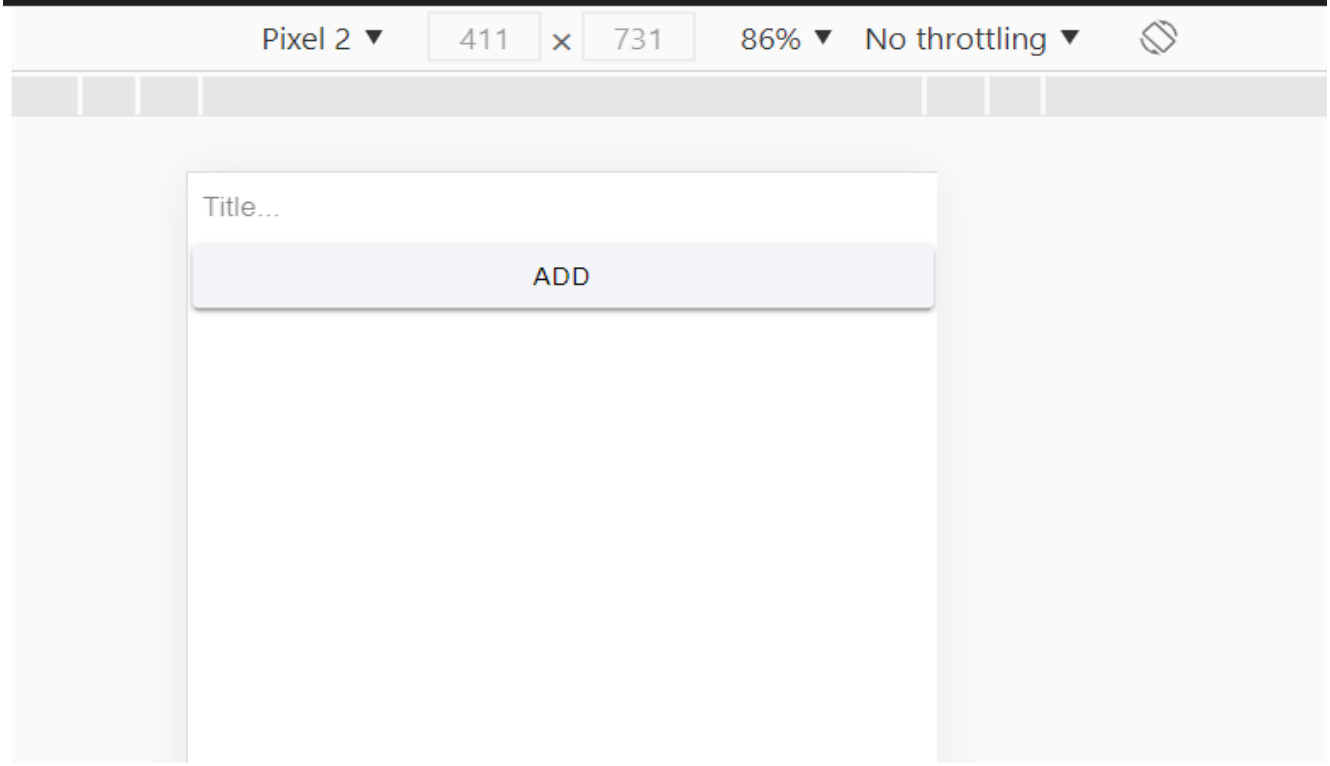


FIGURE 15. Using Ionic components without CSS

In the figure above (Figure 16), there is no CSS or JavaScript file, but the program can render the input field and the Add button adequately. Each is margined into the page and the text holder is scaled like how it is viewed on mobile devices. The Add button uses a different font, has colour and shader with each letter placed beautifully. The page also works on other devices, such as Pixel 2 (Figure 17).

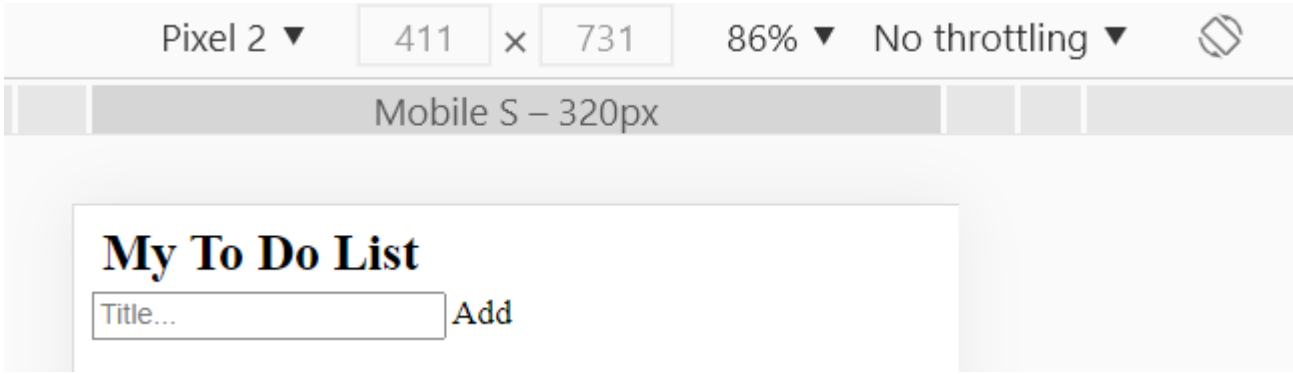


FIGURE 16. The codes in Case 1 without CSS

However, to replicate the same website like in Case Study 1, there are more codes to be added. For the HTML files, `<ion-app>` needs to be declared to use Ionic components. `<ion-header>` is used to declare the header part, which is our banner 'My To do list' imported as an `<ion-toolbar>`. The content is the main part of the project is stored in `<ion-content>` are separated by `<ion-item>` tags instead of `<p>`. To create a list, `<ion-list>` and `<ion-item>` replaces the `<ul>` and `<li>`. In the end, the .html file contains 47 lines.

With this, many codes are deleted from the CSS files. The rest of the codes are used to add features to the above elements, such as: check the list, or make the cursor becomes a pointer when hovering on, etc. Because of this, the CSS file now only has 28 lines.

The most important part is the JavaScript file. Since the CDN is imported instead of the whole framework, many codes need to be changed. Items in `<ion-list>` have shadow-DOM inside them so it is very dangerous to manipulate DOM in that element and getting the elements in the list impractical. This results in a new `strikeIt()` function that toggles the class which is added to every item when they are created. In the end, the new JavaScript still has 28 lines, nearly the same as Case Study 1 since it is possible to code the JavaScript this way in the previous Case.

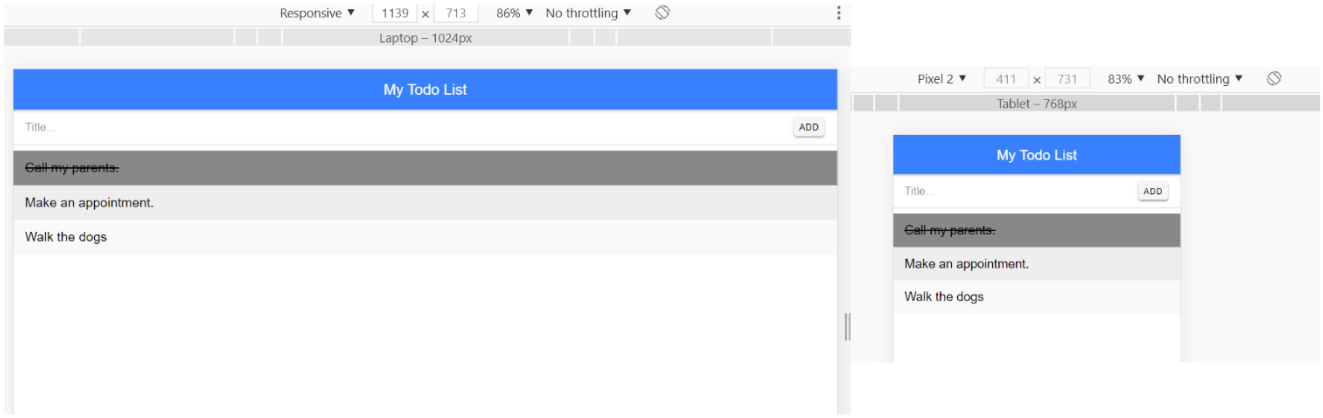


FIGURE 17. Webpage viewed in PC (left) and Pixel 2 (right)

Finally, many Ionic scripts can be added to the website as well. The figure below shows the custom configuration for the alert message when the user did not type anything in the input, but still press the 'Add' button.

```
const alert = document.createElement('ion-alert');
alert.header = 'Warning!';
alert.message = 'You forgot to type in the Todo note. Please take a look back.';
alert.buttons = ['OK!'];

document.body.appendChild(alert);
alert.present();
```

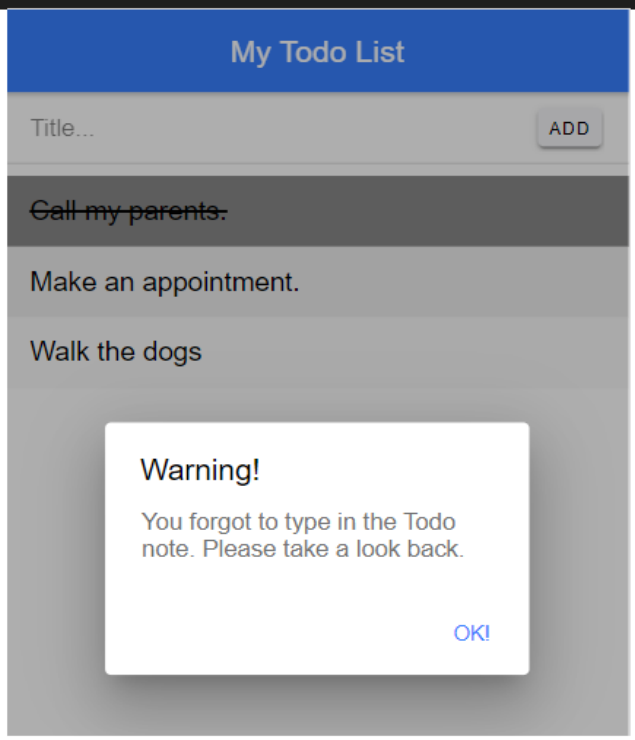


FIGURE 18. The codes to modify the alert with the mobile's phone standard

4.3 Adding Database to the Project

Since many services were created when the Internet growing, this study case will provide a service to the project as well. By utilizing those services, many features will be added into the web page and the coding time can be reduced tremendously. With that in mind, Firebase will be chosen to provide services for this project. The CDN of Firebase can be loaded using the following codes from Figure 20.

```
<script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-firestore.js"></script>
```

FIGURE 19. Firebase CDN

The next step is to create the database on Firebase's server through their website. After that, the key is given to connect to the database. Since Firebase uses NoSQL as a database language, the schema for the project is unnecessary. In order to initialize the database and provide encryptions for this project, many configurations are required beforehand using the codes in Figure 21. These configurations can be put inside the main JavaScript file, or created separately as environment depending on the scale of the project.

```
var firebaseConfig = {
  apiKey: "[apiKey]",
  authDomain: "[projectName].firebaseapp.com",
  projectId: "[projectName]",
  storageBucket: "[projectName].appspot.com",
  messagingSenderId: "47909479809",
  appId: "1:47909479809:web:14a422c90417babb67f480",
  measurementId: "G-T72CG336Z5",
};
```

FIGURE 20. The key used in the thesis. Information in orange is generated automatically by Firebase

Since Firebase also uses REST API, there are many configurations to authenticate the project before accessing the database. Depending on what services the website provides, each property can be added, removed or modified. Here is what service each property gives.

1. ProjectID: This contains the name of the database stored on Firebase, which is required to make a request to the database.
2. APIKey: This is the project's ID, which is used when there are many websites or applications connecting to the same database.
3. AuthDomain: This includes Firebase's Authentication service to identify the user of the project.
4. StorageBucket: This is the URL directing to the database.
5. MessengingSenderID: This is the service to send notifications to user's phones.
6. AppID: This is the identifier for cross-platform app.
7. MeasurementID: This is the ID of Analytic service for Firebase's project.

After declaring the key for service, the project needs to makes a connection to the database using the function `initializeApp()` in Figure 22. Then, from the link, the reference to the database can be retrieved using the function `firestore()` inside the link. Without this step, those keys above cannot be initialized.

```
firebase.initializeApp(firebaseConfig);

const db = firebase.firestore();
```

FIGURE 21. Creating object db to store reference to the database

After that, the database needs to be initialized in the server for further connection. This can be done by going to the Project on Firebase website and creating the first document with data inside there. Then, the next thing that needs to be done is sending the HTTP request to manipulate the data.

```
db.collection('notes').get().then((snapshot) => {
    snapshot.docs.forEach(doc => {
        renderNotes(doc);
    })
});
```

FIGURE 22. A GET HTTP request.

In the figure above the website will send GET requests using ‘db’ reference to retrieve information in the ‘notes’ collection. Then the notes at that moment will be sent back to the website. Since the website does not need to have its URL updated after the GET request, the snapshot parameter will be used instead of a subscription. Also, this request only occurs once, the asynchrony function is used to perform the next function after receiving the data requested. Lastly, for each data inside the document data received, the website will render it using the function `renderNotes()`.

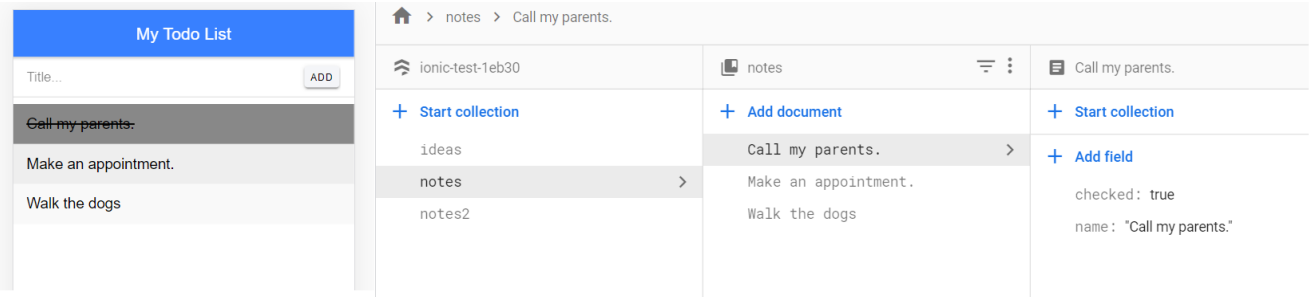


FIGURE 23. The data is displayed as content (left) based on the data in Firebase (right)

Based on the GET method, other requests can also be added to the website for further manipulation. The figure below demonstrates ADD method to send data to Firebase. The message will be a JSON object, which has its name that comes from the value user input (a new to-do item) and it will start without the ‘checked’ class. If the data is written to the database successfully, the page will be reloaded. Otherwise, an error will be returned to the console log.

```
db.collection('notes').doc(inputValue).set({
    name: inputValue,
    checked: false
}).then(() => {location.reload(); return false;})
.catch((error) => console.error(error));
```

FIGURE 24. ADD method

For the hosting part, it will use the same method as the case above. The website can be uploaded into File Manager of InfinityFree. With many advancement in hosting services, it is possible to host the project alongside CDN.

4.4 Summary of Study Case 2

When switching from pure basic codes to Ionic Framework, there has been many changes. First, many assets and templates have been added to reduce the number of codes the programmer has to write. Second, those assets come with their own script as well, so companies can change their behavior right in JavaScript files. Finally, the framework is very compatible with Firebase wrapped inside the project. By utilizing this, time spent on coding can be reduced to be used in developing more features or debugging.

While the first Case can be compared to a website in the early 2000s since it only contains pure HTML, CSS, and JavaScript codes, the second Case can be the innovated ones in the late 2000s. The use of Ionic syntax gives a cohesive minimalistic theme that is easy to view on both computer and mobile devices. Furthermore, the usage of databases leads to easy and maintainable data access. The JavaScript codes are reduced can also decrease the chance of XML attacks.

5 CASE STUDY 3: TO-DO LIST WITH FULL IONIC FRAMEWORK

In practice, Ionic Framework can do more when it is not imported as a CDN component. As it has been stated above, the framework also comes with an Angular framework built inside it with many common NodeJS modules and components. When importing Ionic as CDN, many Angular functions are lost, reducing many features the framework provides. Therefore, this case study will recode the project from scratch with the framework imported directly from Github. Moreover, by coding the project this way, many modules and services such as Firebase can be optimized and deployed into the server. The codebase for this case study can be found in the references, like other Case Studies (Nguyen, 2021).

5.1 NodeJS

Thanks to Visual Studio Code's Terminal, the framework can be installed directly to the programmer's computer. However, before installing Ionic, NodeJS must be installed first. It is a JavaScript runtime environment that store many libraries to perform I/O operations, such as: reading from network, accessing database or filesystem. Instead of blocking the thread and wasting CPU cycles, NodeJS will resume the operations when the response comeback (Heller, 2020). NodeJS can also be used to run the live server, just as the extension 'Live Server' stated above. The Figure below demonstrates using NodeJS to open a local port to run a server.

FIGURE 25. NodeJS Server (NodeJS, 2021)

To install NodeJS, the programmer can download it from its website and run it as administrator. After that, the programmer may have to set up their System Environment Variable by going to Windows System, then proceeding to Advance System Setting and add the PATH to direct it to npm's bin and NodeJS's directory (Figure 27).

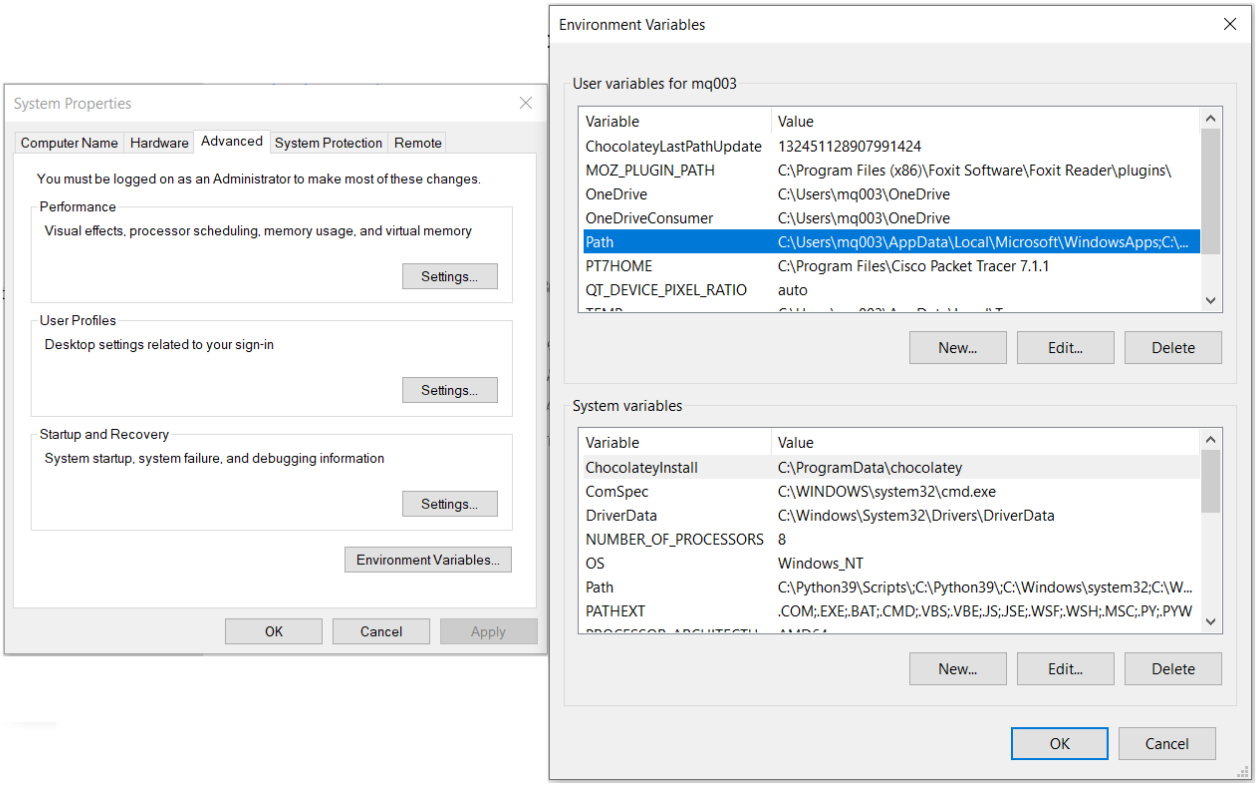


FIGURE 26. Setting up System Environment Variables

After installing NodeJS, programmers can use Visual Studio Code's Terminal to access its CLI (command line interface) to execute the operating system's command. With this, Ionic can be installed to the programmer's operating system by using a command line:

$ npm install -g @ionic/cli.

The 'npm' stands for the NodeJS' syntax, and '-g' means the framework will be installed as a set of modules inside the `node_modules` directory. If everything is correct, the Terminal will start loading the components like in the Figure below.

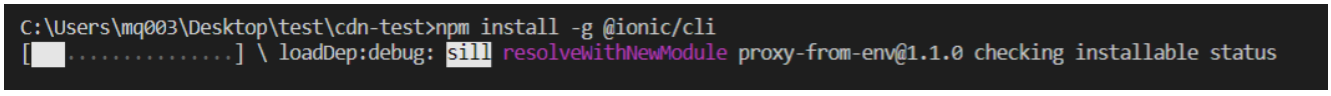


Figure 27. NodeJS fetching Ionic Framework

An Ionic project can be created using the following command line.

$ ionic start <name> <template> [option]

This command-line fetches the repository of Ionic from Github depending on the template chosen. Those templates are the framework that Ionic can be built on top of, including Angular, Vue or React. During the initializing, the name of the project will be put into a configuration JavaScript file. There are also options to install dependencies automatically and link to other Github as well.

5.2 Ionic Structure

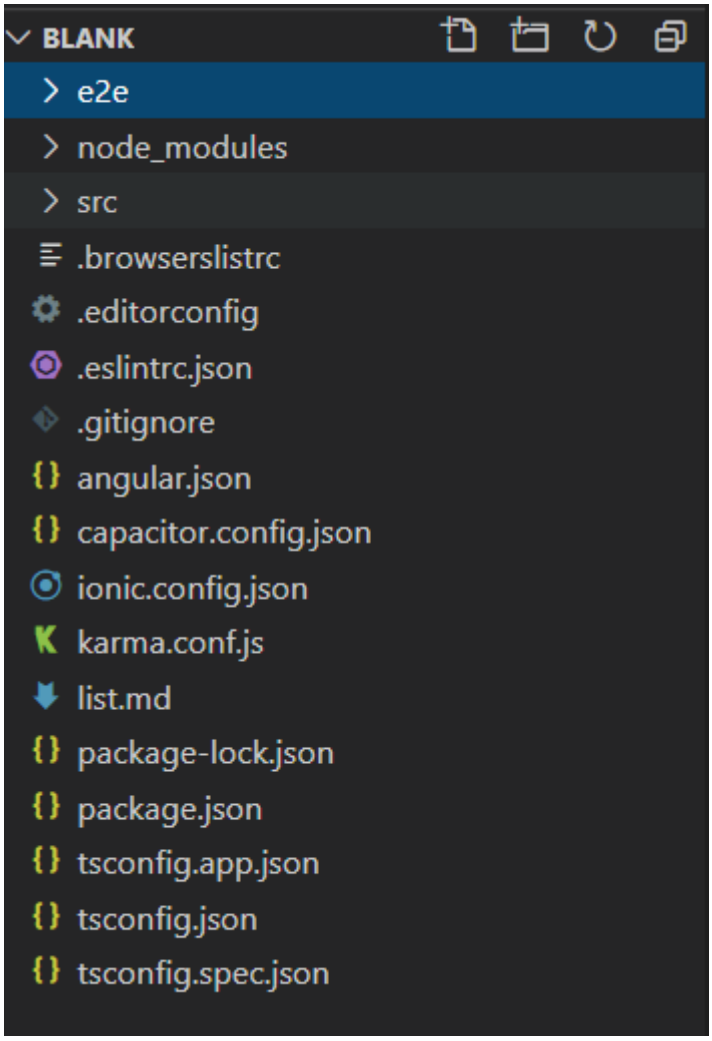


FIGURE 28. Ionic project structure

From Figure 29, it is easy to recognize that the Ionic project has three directories: e2e, src, and node_modules. The /e2e directory contains the components required for the E2E test and uses the Karma file to render it on the browser. This is a testing unit made for Angular framework so it is usable in an Ionic project as well.

To prevent confusing structure, all the NodeJS modules are stored in `/node_modules` directory. This includes frameworks, tools, functions, etc. that can be imported into the scripting files. This directory contains most of the library Ionic project needs, so it is the heaviest directory of all three.

The last directory is `/src` and that is where Ionic stores its source code. Anything that belongs to the website, including front-end, back-end, environment, assets, etc. will be stored here. The rest of the files are configuration files for external services like Github and Tslint. Github is the only database to store codes and Tslint is an extension to check code syntax. The list of dependencies (other modules) can be shown through package.json.

The theme template of Ionic comes from the `/theme` directory and the global `.scss`. The `.scss` file imports styling files from the Ionic server while the one in `/theme` initializes the classes used for Ionic syntax. Furthermore, the `/assets` directory contains pictures and video assets necessary for the website. The `/environment` directory contains global variables that are accessible anywhere in `/src`. The `/app` directory contains the pages of the website and the routing.

To preview the project, the programmer can use the following command line.

```
$ionic serve
```

When the CLI executes, the terminal will open the default browser and use a local port to host the page server in real-time. Any changes in the codes will also change how the website works.

5.3 Angular Architecture

The pages in the Ionic project are made using Angular Architecture and they are called app components. Instead of using a View Template like Laravel or any other frameworks, Angular's app is made from coding blocks called modules. The app will contain one module to enable bootstrapping and other modules for different features. Each module will have two type of components. One is views, a set of elements that Angular can choose among and modify according to programming logic and data. The other one is service, a set of functionalities injected into views as dependencies. There are also Router services to define the navigation between views (Introduction to Angular Concept, 2021).

To create a web page, a directory should be made in the `/src` directory. That directory will store all the modules required for the view and the services. A typical page includes an HTML and CSS module for the view, a root module, a module for routing, a root component to provide more functions and a testing module. The command line below can also be used to generate a web page.

```
$ ionic generate page <name>
```

When the page is initialized, the `index.html` file will be called to direct to the root module. The root module will then create a bootstrap array and insert it into the browser's DOM tree. The root module also declares which framework module is used for the page to load and export necessary data outside. In Figure below, the root module imports the module for front-end (FormsModule, IonicModule), navigation (HiPageRoutingModule) and Angular features (CommonModule). Then, it declares the directory of its component, which is HiPage (Figure 30).

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

import { IonicModule } from '@ionic/angular';

import { HiPageRoutingModule } from './hi-routing.module';

import { HiPage } from './hi.page';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    HiPageRoutingModule
  ],
  declarations: [HiPage]
})
export class HiPageModule {}
```

FIGURE 29. Root module

After that, the root component will be imported (Figure 31). The browser then reads the template of the page (HTML file) and adds the other components, in this case, the SCSS style. Then it declares it-self outside as selectors so the programmer can make a reference to it from the HTML page.

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

@Component({
  selector: 'app-hi',
  templateUrl: './hi.page.html',
  styleUrls: ['./hi.page.scss'],
})
export class HiPage implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

FIGURE 30. Root app component

With this, the Ionic web page does not require a header. The header of the page will come from the root module and root component instead (Figure 32).

```
<ion-header>
  <ion-toolbar>
    <ion-title>Hi</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>

</ion-content>
```

FIGURE 31. HTML component

5.4 Importing old projects into Ionic

As Ionic uses Angular structure, a lot of the codes from Case Study 2 will have to be rewritten. Since Angular's architecture relies on building blocks using modules, JavaScript is replaced with the TypeScript library, allowing more advanced features. HTML file also needs to be rewritten to adapt to modern web design. While there are many improvements that can be made to CSS file, it is not necessary to rewrite the codes due to the simplicity of this project.

5.5 Rewriting JavaScript codes into TypeScript modules

Thanks to NodeJS, Firebase can be integrated into the project as modules. When CDN provides a fast way to implement databases, NodeJS allows programmers to install them as dependencies. This means programmers can use the Firebase module to do HTTP Request to the server, utilizing all functions the database provides. The next line is the command line to install the dependencies for the Angular framework.

$ npm install firebase @angular/fire

After the installation, the reference to the database can be added to the project using the key. However, since this is an Ionic framework and the key mostly never change, it is better to add that key to the `/environment` directory to make it globally accessible. Therefore, the codes in Figure 21 can be added to '`environment.ts`' file. Programmers also have to let the app access to /environment directory and Firebase module through '`app.module.ts`'.

As shown in Figure below, three modules 'AngularFireModule', 'environment', and 'AngularFireStoreModule' are imported. '@NgModule' also needs to contain them in the import section so that when the website is available, the modules are also loaded alongside.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouteReuseStrategy } from '@angular/router';

import { IonicModule, IonicRouteStrategy } from '@ionic/angular';

import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';

import { AngularFireModule } from '@angular/fire';
import { environment } from '../environments/environment';
import { AngularFirestoreModule } from '@angular/fire/firestore';

@NgModule({
  declarations: [AppComponent],
  entryComponents: [],
  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule,
    AngularFireModule.initializeApp(environment.firebase),
    AngularFirestoreModule],
  providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy }],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

FIGURE 32. App Module

Then, just like in CDN Case, the link between the website and the database must be made using the key. Since components should not fetch or save data directly from outside, a service must be established to get the data from Firebase. After handling the data, components of the pages will present it. The code below can be used to generate a new service. For this Study Case, the service will be put inside /service directory.

$ ionic generate service <service-name>

The Figure below shows the codes required to make the reference. Instead of using JSON object this time, the reference is stored in an Observable. They are patterns used to handle data whose states change frequently. With this, the website can subscribe to the database and listen to any changes. The todo is an Observable, meaning it will listen to the reference `todoCollection` and when there is a change in the database, it will be used to contain the new data. The rest of the codes are the same as in CDN. The programmer only needs to import function ‘map’ and ‘take’ from module `rxjs` since they are not available by default.

```
export class TodoService {

  private todo: Observable<Todo[]>;
  private todoCollection: AngularFirestoreCollection<Todo>;

  constructor(private collectionRef: AngularFirestore) {
    this.todoCollection = this.collectionRef.collection<Todo>('notes2');
    this.todo = this.todoCollection.snapshotChanges().pipe(
      map(actions => {
        return actions.map(a => {
          const data = a.payload.doc.data();
          const id = a.payload.doc.id;
          return { id, ...data };
        });
      })
    );
  }
```

FIGURE 33. New database reference

Furthermore, since this is a service, it can hold an interface for the JSON object. The Figure below contains the codes to declare that interface. Since ‘id’ is generated and taken from the database server, a question mark is put next to it so the program can understand.

```
export interface Todo {
  id?: string,
  name: string,
  checked: boolean
}
```

FIGURE 34. Todo Interface

Next, HTTP Request is created here. The method is likely the same as JavaScript. To get the data from the database, the function `getTodo()` must be called. Then it will return the Observable, which is listening to the server all the time. Then, `getTodo()` will hold the state of that data at that time (Figure 36).

```
// Return all document's data
getTodos(): Observable<Todo[]> {
  return this.todo;
}
```

Figure 35. GET Request

The other two requests will be a little bit different. The POST Request use Promise instead of Observable. A Promise is an object of an asynchronous operation to produce a single value sometime in the future to show the resolve state. A promise maybe in one of three possible states: fulfill, rejected, or pending. With this, when the program sends a POST Request to add more to-do notes, `addTodo()` will show if the process success or not.

Contrary to `addTodo()`, `updateTodo()` looks for the note in the database with the specific id. Then, it gets the value of the checked and assigns the opposite value to the database. This is used to toggle the checked status of each note. After completing, it does not return anything (Figure 37).

```
  addTodo(todo: Todo): Promise<DocumentReference> {
    return this.todoCollection.add(todo);
  }

  // Update the list to be checked or not
  updateTodo(todo: Todo): Promise<void> {
    return this.todoCollection.doc(todo.id).update({ checked: !(todo.checked) });
  }
}
```

FIGURE 36. POST and UPDATE Request

After this, the service will be imported into the root component of the page. In the file `home.page.ts`, Observable and Router modules are also imported. There is a `todo` object using the interface of the service and the `todos` Observable here. When the page is loaded, `ngOnIt()` will run and initialize the list, putting data received into the `todos` Observable. This action always repeats whenever the page reloads as well. Next, when `addTodo()` is called, the POST Request in service will be called to push data to the database and refresh the page. When `updateTodo()` is called, it gets the id and the `checked` status and runs the service to toggle it on the database (Figure 38).

```
export class HomePage implements OnInit{

  todo: Todo = {
    name: '',
    checked: false
  };

  public todos: Observable<Todo[]>;

  constructor(private todoService: TodoService, private router: Router) {}

  ngOnInit() {
    this.todos = this.todoService.getTodos();
  }



  addTodo() {
    this.todoService.addTodo(this.todo).then(() => { this.router.navigateByUrl('/')})
  }

  updateTodo(id, isChecked) {
    this.todo.id = id;
    this.todo.checked = isChecked;
    this.todoService.updateTodo(this.todo).then(() => { this.router.navigateByUrl('/')})

  }
```

FIGURE 37. Root component with HTTP Request

Therefore, the HTML page needs to be rewritten a little bit also. First, `ngModel` is a function on Angular which tracks the value and user interaction to keep the view in sync with the model established in TypeScript. When something is written into the ion-input, functions in the root component will track and put it in the `todo` object model. Then, press the Add button, that same `todo` will be uploaded to the database.

First, in the list of notes, the `*ngFor` is a loop function that will run until the number of `todo` equal to the number of `todos`, which has been initialized when the page is created. Here, async is used to make the GET Request keeps going until there is no data left. Second, after getting all of the data, each Observable will be assigned to them, become an ion-item and having their class depending on the checked variable. Each also has a function that toggles their own class with a mouse click which updates the database in real-time. Third, Ionic will assign the checked icon depending on the item's class. To make it happen, Angular uses another conditional statement called `*ngIf` that checks the model's checked variable. The icon comes from the Ionic framework that is free to use and can be put into the file using the attribute name. Lastly, the description of the note will be rendered to be viewed on the page (Figure 39). Figure 40 below will demonstrate the end result.

```
<ion-item>
    <ion-input type="text"  placeholder="Title..." [(ngModel)]="todo.name">> </ion-input>
    <ion-button color="light" expand="block" (click) = "addTodo()">Add</ion-button>
</ion-item>
<ion-list id="myUL" lines="none">
  <ion-item *ngFor="let todo of (todos | async)"
  [ngClass] = "todo.checked ? 'checked' : 'nothing' "
  (click) = "updateTodo(this.todo.id, this.todo.checked)">
    <ion-icon *ngIf="todo.checked"name="checkmark-outline"></ion-icon> {{ todo.name }}
  </ion-item>
```

FIGURE 38. The list written in Ionic

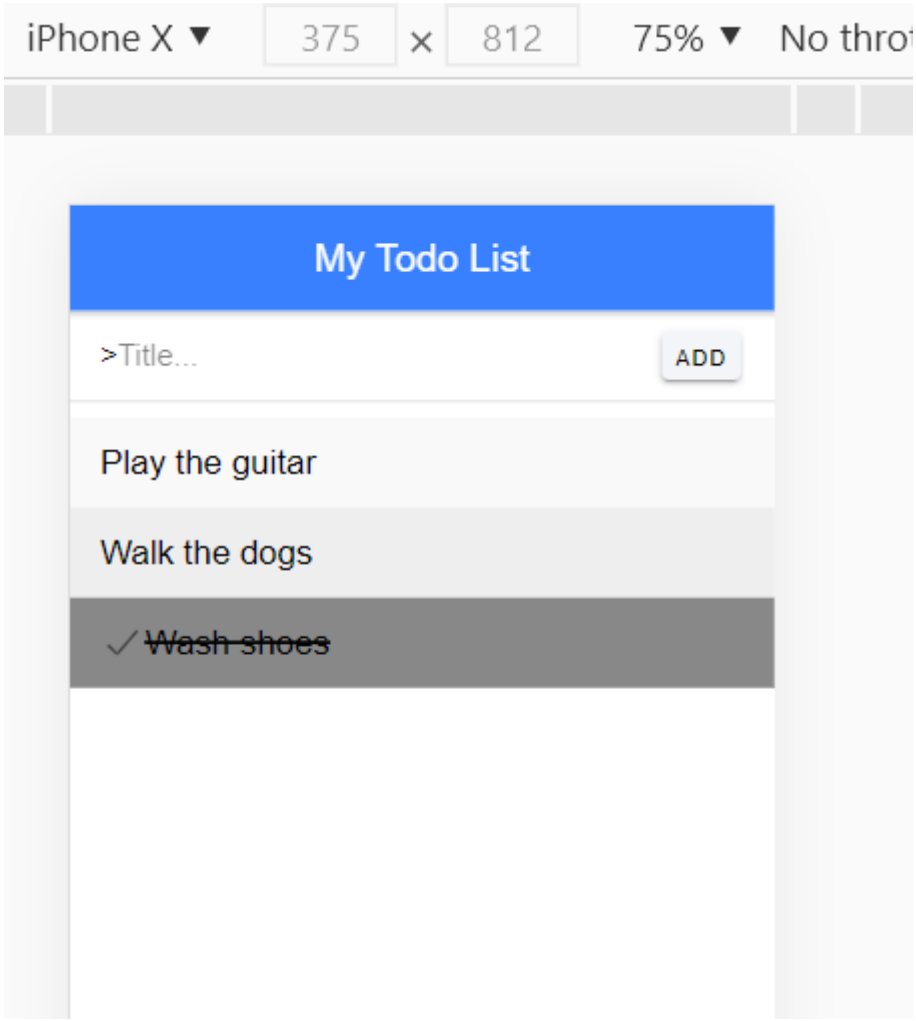


FIGURE 39. Todo list with Ionic Framework

5.6 Hosting Ionic project

Firebase comes along with its web hosting services. By integrated Firebase Hosting into the project, files from local directories can be deployed as dynamic content. Firebase service is also free and provides SSL certificates with impressive speed across multiple geographic locations without the need for a sperate CDN on top (Uddin, 2021)

Firebase Hosting can be imported into this Ionic project using its CLI. The tool can be installed with the command line.

```
$ npm install -g firebase-tools
```

The NodeJS will install Firebase dependency to a local computer for all users. However, before the hosting can be established, the website must be built into a deployment state. Deploy is a process of converting the codes in a framework into codes that servers can read and understand. Fortunately, Ionic also has a deploying service built alongside, and any programmer can use it using the following command.

```
$ ionic build
```

When the building process is completed, a new directory named '`www`' will appear in the project root directory. This contains the result of converting all the codes in the framework into the codes that the hosting server can understand. When the server renders the page, it will look for the specific `index.html` file use it to initialize the site.

The next step would be uploading the codes into the hosting server. To do that, the programmer needs to authorize by login to Firebase with the command.

```
$ firebase login
```

After that, the programmer can choose to import the local project to any Firebase project. Since a Firebase project to create a database is made already, it is possible to use that same project to host the server. The command used to connect to the hosting service is shown below.

```
$ firebase init
```

The public directory to push the codes will be 'www' directory. Since Ionic can be used to make a progressive web app to use on many devices, it is possible to host them as a single-page app. After that, the programmer can use `$ firebase serve` to check the codes or send the codes to the server using `$ firebase deploy`. Firebase then will return an accessible link that can be used to view the codes online on any devices. The graph below is the screenshot taken from a mobile device when previewing the project.

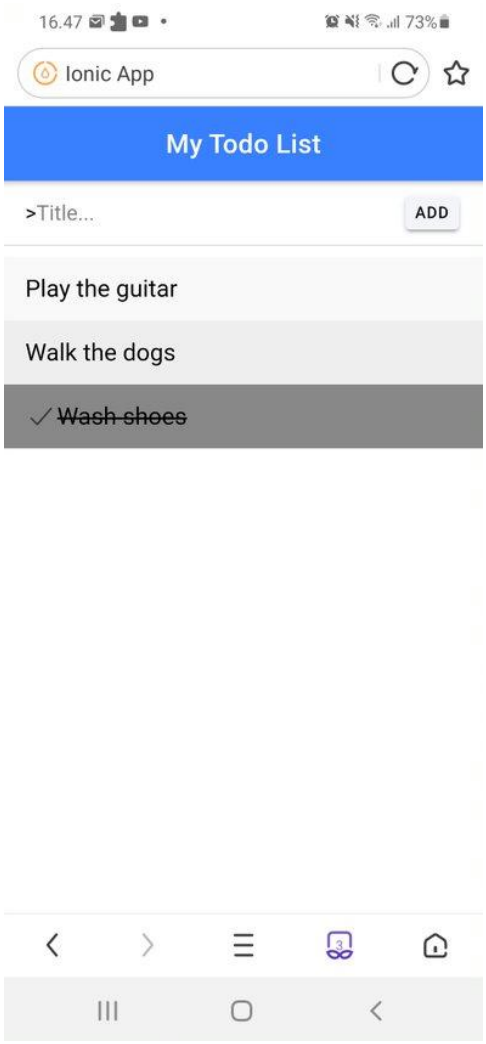


Figure 40. Android view

5.7 Summary of Case Study 3

The switch from pure HTML, CSS and JavaScript into Ionic makes many great changes. From the code perspective, while the CSS codes are reusable, half of HTML codes must be written due to the lack of DOM manipulation. All JavaScript codes are removed and changed into TypeScript to be compatible with the Angular framework built under Ionic. From the structure perspective, codes can be organized into separate parts of the project to be imported efficiently.

While those changes increase the complexity, it also gives many benefits. First, programmers can get access to many useful assets from the Ionic library. Second, there are many open-source modules or extensions that can be imported to gives the website more functions. Lastly, the scale of the project is greatly improved, leading to the fact that the coding time of developers is reduced, and the website will not be out-of-date for the next few decades. One great example of these benefits is Firebase, where Ionic developers can host, stores, makes database with many other analytic features in its modules.

6 CONCLUSIONS

Creating a good website has never been easy. That is the reason why choosing the right technology and catching up with the trends at the start are very important, especially at the beginning stage of the project. In addition, it is possible for companies to utilize a group of developers to build a low-cost website that can be released on multiple platforms and potentially reach many profitable consumers.

One of the popular frameworks is Ionic, which is built on top of Angular. Using three study cases, the thesis can establish the scalability of a 'To do list' website using Ionic and its dependency. Using modules as its core, Ionic provides many useful themes and services to shorten the codes needed. Furthermore, with the help of Angular, HTML has more functions for the DOM Manipulations, and TypeScript is used to link the functionalities into a cohesive block. This allows Ionic to have a stable structure to create build for each platform and deployment.

From the first case, it is proven that the websites in the 1990s were very simple, but functional. This explains the rise of blogging at the end of 1990 that made the Internet become a place to express creativity using CSS. In return, the websites lacked scalability and security. From the 2000s, the trend of blogging appeared with the help of other services, and with those services came websites that handle every human's needs. From then, websites got harder to make but were more functional. In the late 2010s, websites became too complex with security measures and functions, leading to a point that a web designer is required to make a good website. Although websites could run on multiple platforms and had many linked services, they required structure and many other frameworks to ensure scalability and stability.

In conclusion, as website development grows rapidly, more requirements are put onto developers, and new frameworks and modules come out every year to support the changes. For further research, studies about other frameworks can be useful to improve developer's coding techniques. Many of them are popular in the market, such as React Native, Adonis, Laravel. Furthermore, studies about open-source libraries and modules can ease the coding time and help the website achieve maximum performance. Those libraries and modules can be found on many platforms, such as GitHub, NodeJS, StackOverflow, etc. By mastering as much as possible, the developers can deliver solutions to every challenge the company faces, attracting customers effectively in the 4.0 industry revolution.

REFERENCES

B, A. (2021, 09 March). *What is CSS*. Retrieved from Hostinger: https://www.hostinger.com/tutorials/what-is-css

Barth, A. (2010, January 26). *Security in Depth: New Security Features*. Retrieved from Chromium Blog: https://blog.chromium.org/2010/01/security-in-depth-new-security-features.html

Bhargava, M. (2018). *Educative*. Retrieved April 2021, from Learn HTML, CSS and JS from scratch: https://www.educative.io/courses/learn-html-css-javascript-from-scratch

Blumenthal, S. (2017). *JavaScript: JavaScript For Beginners - Learn JavaScript with ease in HALF THE TIME - Everything about the Language, Coding, Programming and Web Pages that you need to know!* CreateSpace Independent Publishing Platform.

Brill, D. (2020, January 27). *Online Innovations*. Retrieved April 2021, from Recapping the 2010s: The biggest changes in Web Design & Development: https://www.onlineinnovations.com/2010s-web-design-development-changes

CERN Info. (1991). Retrieved April 2021, from WWW Project: http://info.cern.ch/hypertext/WWW/TheProject.html

Cheng, F. (2018). *Build Mobile Apps with Ionic 4 and Firebase: Hybrid Mobile App Development.* Apress.

Cloudflare. (2009). Retrieved April 2021, from What is a CDN: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/

Curtis, S. (2021, June 15). *What is HTML*. Retrieved from Hostinger: https://www.hostinger.com/tutorials/what-is-html

Fergus Baird, K. S. (2020, November 2). *Hootsuite*. Retrieved April 2021, from Youtube Analytics: https://blog.hootsuite.com/youtube-analytics/

Heller, M. (2020, April 6). *What is Node.js? The JavaScript runtime explained*. Retrieved April 2021, from InfoWorld: https://www.infoworld.com/article/3210589/what-is-nodejs-javascript-runtime-explained.html

Hewlett Packard in 2000. (n.d.). *Web Design Museum*. Retrieved Apirl 2021, from https://www.webdesignmuseum.org/gallery/hewlett-packard-2000

Introduction to Angular Concept. (2021, April). Retrieved from https://angular.io/guide/architecture

Lara, C. (2020, July 8). *The Digital*. Retrieved April 2021, from Web Design Trend: https://www.theedigital.com/blog/web-design-trends

Larsen, R. (2013). *Beginning of HTML and CSS.* Wrox.

MicroStartups. (2019, September 13). What You Need To Know About The History Of Web Hosting. Retrieved April 2021, from https://www.wpsupportspecialists.com/what-you-need-to-know-about-the-history-of-web-hosting/

Mittal, H. (2011, September 12). Microsoft Build. *Introduction to Web Deploy*. Retrieved April 2021, from https://docs.microsoft.com/en-us/iis/publish/using-web-deploy/introduction-to-web-deploy

Nguyen, Q. (2021). To Do List - Base Codes. GitHub Repository. Retrieved April 2021, from https://github.com/mq003at/html-test

Nguyen, Q. (2021). To-Do List - CDN. Retrieved April 2021, from https://github.com/mq003at/ionic-cdn-test/tree/with-database

Nguyen, Q. (2021). To-Do List - Ionic. Retrieved April 2021, from https://github.com/mq003at/ionic-todo

NodeJS. (2021). *Introduction to NodeJS*. Retrieved April 2021

Patni, S. (2017). *Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS.* Apress.

Sharma, A. (2020, December 23). *Difference between SQL and NoSQL*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/difference-between-sql-and-nosql/

Singh, V. (2020, April 9). *hackr.io.* Retrieved April 2021, from What is Frameworks? [Definition] Types of Frameworks: https://hackr.io/blog/what-is-frameworks

TK. (2020, December 28). *Evolution of Web Design and Development 1990-2019*. Retrieved from Red Stapler: https://redstapler.co/evolution-webdev-webdesign-1990-2019/

Uddin, A. (2021, May). How to host static website. Retrieved from Medium.

W3School. (2021). How TO - Create a To Do List. Retrieved April 2021, from https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_todo

What is Web Hosting. (2021, May). Retrieved from NameCheap: https://www.namecheap.com/hosting/what-is-web-hosting-definition/

Wordpress.org. (2021, June 17). Twenty Ten. *Wordpress*. Retrieved April 2021, from https://wordpress.org/themes/twentyten/

Xia, V. (2017, December 21). *Medium*. Retrieved April 2021, from What is Mobile First Design? Why It's Important & How To Make It?: https://medium.com/@Vincentxia77/what-is-mobile-first-design-why-its-important-how-to-make-it-7d3cf2e29d00