Question 1

Source control (or version control) refers to tracking and managing changes to code. This ensures that developers are always working on the right version of source code. The codebase is usually stored in a SCM (Source Control Management) system which will help to track and manage code that is uploaded/pushed from multiple sources/developers. The SCM system will help to track code changes, historical changes/revision, different versions of the codebase at different points in the development process and enable collaboration between multiple developers – each with their own versions of the codebase which can later be merged together after detectable conflicts are resolved. Developers will typically work on their own forks and branches and commit their code for review by senior developers.

Question 2

Amongst what would be an endless list of aspects that encompasses “quality software”, here is a list of 5 general aspects that would be of high priority when it comes to quality software.

- Testing

Making sure that the application (and/or individual components/functions of the application) works as expected involves the use of a testing framework such as Jest. This ensures that function output and behaviour is as expected during and after the development process. User testing would also help to catch bugs, logic errors and facilitate User Interface adjustments based on user feedback and product alpha/beta testing.

- Scalability

The application needs to be engineered to be capable of growing to support the size of its increasing userbase. The foundation architecture of the application may have an effect in this outcome. Is it a Monolith application? ... or does the application use microservices? - perhaps both server side and client side. Database architecture is also an important consideration in terms of scalability. A document-based database (such as Mondo DB) is considered to be more easily and quickly deployable and scalable than an table/relational-based database. Mongo DB requires less planning and it is not as rigid in the way that table/relational-based database are to implement.

- Maintainability

The application needs to be easily maintainable by its engineers. The code base needs to be refactorable and re-implemented without causing significant interruption to users. This may perhaps come down to how the application was architecturally implemented. This is also one of many reasons that microservice architecture is generally used with larger applications, as Microservices allow parts of an application to be totally offline without effecting the rest of the application or its users. Another factor would be workflow, source control process and internal company procedures.

- Performance

As the userbase increases and/or new features implemented, the application needs to have been engineered for maximum speed, no matter how large the userbase. Server processing power will also play a major role here, as well as server bandwidth and also whether the application has been deployed using CDN technology (depending if global access is required).

- Security

Any software application needs to be secure against cyber-attacks and service disruption, information theft and identity theft. Keeping application dependency package versions current and up to date helps in maintaining security vulnerabilities. Also, maintaining server operating systems and security utilities can help to prevent malware and spyware script injections into hosted software applications.

Question 3

A MERN stack application is comprised of four main technologies – Mongo, Express, React and Node.

features - All four configurations can be used free of charge as an open source. - All four configurations are based on JavaScript.

JavaScript was a language created for front-end development, so back-end used languages such as Java and C#, but later back-end development programs based on JavaScript such as MongoDB and Node.js came out, allowing both front-end and back-end to work in one language.

From the developer's point of view, it's good to learn all of the back-end and front-end development just by learning one language (of course, just knowing the language doesn't make it easier, but it's easier for companies to use manpower more efficiently). Recently, even smartphone applications can be developed with JavaScript/HTML (to be precise, websites created with React can be converted into apps), so the use of JavaScript continues to increase.

MongoDB Mongo is a document-orientated “NoSQL” database program. It is used to store information in documents (in JSON format) – as opposed to a traditional SQL database which uses relational tables.

Node & Express Node is a JavaScript environment with libraries that make it easier to write software. Express extends Node specifically to make it easier to create webservers. Therefore, Express is a server-side framework which is used for building web/mobile applications and API’s.

React React is a JavaScript library/framework for building client-side user interfaces. This is the part of the application that is rendered in the user web browser and is how the user interacts with the application.

As stated, the user (client) interacts with the application via the user interface which is rendered to the users screen via the React framework. When the user/client triggers certain events, React will send or request data to/from the applications Server/Backend (Node/Express) API/s which are listening for incoming requests. The Server will then send a response/s back to the client – perhaps querying and saving data to the Mongo database first.

Question 4

Firstly, the dev team (or project manager or business analyst) would need to meet with the client in order to assess the business requirements of the website or web application.

This would help determine what technologies need to be used to develop a website or application so that it would meet the businesses requirements - now and also perhaps into the future (referring to future scalability).

Depending on what the website or web application requirements are – a static website could easily be done with HTML, CSS and JavaScript, perhaps even using a Framework/Library such as React.

If a web application is required, full knowledge of MERN stack or similar server-side / client-side application design would be required. The team would need to produce technical documentation such as ERD charts, User-flow diagrams as well as UI and UX designs wireframes and low or high-fidelity designs.

Working as a team, a Source Control Process and Source Control Management solution would need to be implemented so as to keep the development process and codebase managed.

Also, depending on the size/scope of the project, the team would also need to determine what development methodology to be suitable for the project. A smaller project might benefit more from 'Rapid Application Development' methodology as opposed to a 'DevOps Deployment' methodology.

Question 5

A project that I worked on earlier in the course was a web application built with Ruby On Rails. Of course, the biggest aspect of this project was learning the Object Orientated language 'Ruby'. Being my first programming language also it was hard to find resources compare to Javascript and Python

Apart from Ruby, learning Rails was also quite an undertaking and a challenge. Having only ever worked with HTML and CSS, I really found the concepts of object-oriented programming and MVC difficult and making sense of how it all fit into the Rails Framework.

I believe that It was a great experience to deal with backend databases such as Postgres, AWS, and ERD.

Question 6

One of the very first projects that I worked on during the Fast-track was the Personal Portfolio website. This was primarily a personal website which was written using only HTML and CSS.

I found that I excelled during this project. I received a lot of affirmations that the way I was coding HTML and CSS was indeed how it was supposed to be done by professional developers. I found this to be very rewarding after spending time learning by myself.

One improvement I have made – and definitely something I have changed – is that I now code SCSS and strict HTML5. I find this to be much more efficient than using older HTML syntax and plain CSS. Also now I always do mobile first when I make a frame work.

Another improvement I have made is the addition of using more JavaScript in the client browser for small things that I previously had tried to do with CSS. I have been able to make my static websites much more visually dynamic.

Question 7

In most programming languages, including Javascript, the code can be concisely expressed by changing the flow of codes that run sequentially, called "control flow." The control flow, if simply understood, is a conditional sentence and a repetition sentence, and is an important part of programming. it is the order in which the computer executes statements in a script. A script in JavaScript may include several/many control structures such as conditionals, loops and functions.

The below If/Else Statement can change the flow of an application based on if a condition that is passed into the statement is determined to be true or false:

```
if (age > 20) {
  goAdultPage()
} else if (age > 14) {
  goTeenPage()
} else {
  goKidPage()
}
```

Question 8

Type coercion is the process of converting value from one type to another (such as string to number, object to boolean, and so on). There are two different types of 'Type Coercion' – Explicit and Implicit.

Explicit type conversion is literally stated! Number(), parseInt(), parseFloat(), String(), Boolean(), and so one. 'Explicit' Type Coercion is when type coercion is done on purpose by writing the appropriate code. The following example will coerce the number 123 into a string (using the JavaScript built-in String function):

```
const explicit = String(123)
console.log(explicit);

Number(null) // 0
Number(undefined) // NaN
Number("-12.34") // -12.34
String(-12.3) // "-12.3"
String(false) // "false"
Boolean(0) // false
Boolean([]) // true
```

'Implicit' Type Coercion is also a strength of JavaScript, but it is also the most avoided function. JavaScript is a loose language, so var automatically changes any type, so it is called double edge sword; it's a great source of frustration and defects, but also a useful mechanism that allows us to write less code without losing the readability. In the following example, the Number 123 will be coerced into a String that reads as "123".

```
const implicit = 123 + ''
console.log(implicit);

1 + 'kyu' // 1kyu : string
1 + '2' + 3 // '123' : string
1 + 2 + '3' // '33' :string
2 * "2" // 4 : number
1 + true // 2 : number
1 + false // 1 : number
```

Thus, there is one operator that does not trigger implicit type coercion is `===`, which is called the strict equality operator.

In fact, once you appreciate it with your eyes, you can see the rough rules. This is because you shouldn't write it like this in the end, and you won't have to write such a bizarre code if you write the code. However, when working with many developers, these bizarre codes come out, and developers who collaborate have to spend a lot of time figuring out the codes, so they use tighter typescripts in the first place.

Question 9

The 'data type' of JavaScript refers to the "type" of data to be expressed. There are several different types of data in JavaScript.

'Primitive' data types include type 'String', 'Number', 'Boolean', 'Null' and 'Undefined'. These data types are referred to as 'Primitive' because their values can contain only a single type (i.e.: a 'String' OR a 'Number', but NOT both combined).

The follow examples are 'primitive' types of data. Each individual data type can only be assigned to one variable:

```
let string = 'foo';
let number = 42;
let boolean = false;
let x = null;
let y = undefined;
```

A more complex data type in JavaScript would be an 'Object'. Objects are used to store collections of data and more complex entities. An example of a more complex entity might be an Object containing other data structures like an Array and even an Object containing other Objects.

The following example is an Object which contains 5 Key/Value pairs, one if which the value is another Object:

```
const kyu = {
  name: 'kyu',
  age: 37,
  gender: 'male',
  single: false,
  spouse: {sammi}
}
```

Question 10

Arrays are data structures that can store, access, manipulate/variable, and delete data. To accomplish this in JavaScript, the array needs to be iterated through while using the appropriate array method for the desired outcome.

```
let fruits = ['apple', 'banana', 'orange', 'pear', 'watermelon'];

// Example 1 – (mutates the original array) Add another fruit to the array :
fruits.push('Kiwi');
console.log(fruits);

// Example 2 – (mutates the original array) Delete 'Kiwi' from the array:
fruits.pop();
console.log(fruits);

// Example 3 – executes a provided function once for each array element:
fruits.forEach(printArray = (fruit) => {
  console.log(fruit);
});

// Example 4 – (does not mutate the original array) Creates a new array from the original array:
const newFruitsArrary = fruits.map(fruit => {
  return fruit
});
console.log(newFruitsArrary);
```

Question 11

The below Object is referred to as an 'Object Literal'. This example of a single Object is a data structure made up of key: value pairs. It has 3 properties – make, model, and year:

```
const myCar = {
    make: 'Hyundai',
    model: 'IX 30',
    year: 2014
};
```

To access the keys/values in this Object and log it to console using string interpolation, I would write the following code:

```
const make = myCar.make
const model = myCar.model
const year = myCar.year

// other way

const make = myCar['make']
const model = myCar['model']
const year = myCar['year']

console.log(`I bought a ${make} ${model} in ${year}`);
```

To create more instances of an Object, a Constructor function is required. Using ES6 syntax, I would do this using the Class function to create more instances of the 'Cars' object:

```
class Cars {
  constructor(make, model, year) {
    this.make = make;
    this.model = model;
    this.year = year;
  }
}

const kyu = new Cars('hyundai', 'IX 35', '2014');
console.log(kyu);

const sammi = new Cars('audi', 'A3', '2017');
console.log(sammi);
```

Question 12

JSON (JavaScript Object Notation), is a format in which data can be stored in an organized, easy-to-access manner.

The JSON.stringify() function converts a JavaScript Object into a JSON string. This is useful for when the data needs to be available via API to be accessible by other servers/clients.

In the below example, the “JSON.stringify()” function is used on a JavaScript Object to convert it to a JSON string, then stored in a variable:

```
const employee = {
  name: "kyu",
  age: 37,
  address: {
    street: "15 fuster street",
    city: "Brisbane"
  },
  interests: ["game", "exercise"]
}
const data = JSON.stringify(employee)
console.log(data);
```

To access data provided in JSON format via an API - or a file that has been “required” into a document, the JSON.parse() function is used. In effect, this is the reverse process, in that a JSON String is now being converted into a JSON Object.

In this example, the JSON String (stored in the variable 'data') is now being parsed back into an Object:

```
const newEmployee = JSON.parse(data);
console.log(newEmployee);
```

Question 13

```
class Car { // this is a 'Car' class that produce object
  constructor(brand) { // constructor that initiallize the object
    this.carname = brand; // make variable(assigne) brand to this.carname
  }
  present() {
    return 'I have a ' + this.carname; // returns 'I have a '(string) concatinated with the this.carname when function is called
  }
}

class Model extends Car { // this class inherits from the Car class
  constructor(brand, mod) { // constructor that initiallize the brand and mod
    super(brand); // call  'Car' class
    this.model = mod; // make variable(assigne) mod to this.model
  }
  show() { // returns this.present concatinated with string and this.model
    return this.present() + ', it was made in ' + this.model;
  }
}

let makes = ["Ford", "Holden", "Toyota"] // make variable in array containing 3 strings
let models = Array.from(new Array(40), (x,i) => i + 1980) // make variable in array containing numbers representing dates since 1980

function randomIntFromInterval(min,max) { // function with two parameters
    return Math.floor(Math.random()*(max-min+1)+min);
} // return ramdom numbers between min and max

for (model of models) { //

  make = makes[randomIntFromInterval(0,makes.length-1)] // make variable for taking out of random number from index of 0 to length -
  model = models[randomIntFromInterval(0,makes.length-1)] // make variable for taking out of random number from index of 0 to length

  mycar = new Model(make, model); // make variable that stored new Model object
  console.log(mycar.show()) // print out the consle randomely choes one with string
}
```

Reference

Q1

Gehman, C., 2021. What Is Source Control and Why Is It Important? | Perforce. [online] Perforce Software. Available at: https://www.perforce.com/blog/vcs/what-source-control [Accessed 30 August 2021].

Q2

dzone.com. 2021. 10 Software Quality Factors That Should Always Be Remembered - DZone Performance. [online] Available at: https://dzone.com/articles/10-groups-software-quality [Accessed 1 September 2021].

Software Testing Material. 2021. What are Quality Attributes in Software Architecture. [online] Available at: https://www.softwaretestingmaterial.com/quality-attributes-in-software-architecture/ [Accessed 1 September 2021].

Q3

MongoDB. 2021. What is the MERN Stack? Introduction & Examples. [online] Available at: https://www.mongodb.com/mern-stack [Accessed 1 September 2021].

Q4

Guru99. 2021. Agile Vs. DevOps: What's the difference?. [online] Available at: https://www.guru99.com/agile-vs-devops.html [Accessed 1 September 2021].

Q7

Developer.mozilla.org. 2021. Control flow - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Glossary/Control_flow [Accessed 1 September 2021].

Q8

Developer.mozilla.org. 2021. Type coercion - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Glossary/Type_coercion [Accessed 1 September 2021].

freeCodeCamp.org. 2021. JavaScript type coercion explained. [online] Available at: https://www.freecodecamp.org/news/js-type-coercion-explained-27ba3d9a2839/ [Accessed 13 September 2021].

Q9

Developer.mozilla.org. 2021. JavaScript data types and data structures - JavaScript | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures [Accessed 5 September 2021].

Q10

Developer.mozilla.org. 2021. Array - JavaScript | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array [Accessed 10 September 2021].

Q11

Developer.mozilla.org. 2021. Working with objects - JavaScript | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects [Accessed 10 September 2021].

Q12

Developer.mozilla.org. 2021. Working with JSON - Learn web development | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON [Accessed 10 September 2021].

Q13

Cs.utah.edu. 2021. Programming - Commenting. [online] Available at: https://www.cs.utah.edu/~germain/PPS/Topics/commenting.html [Accessed 10 September 2021].