### **Video 1**

In this lesson, I learned how to get started building an application with a Vue instance. First, I understood that a Vue instance is the foundation of every Vue application, acting as a data manager and connecting to elements in the DOM via the el property. Vue also allows displaying data on the page using the {{ }} (expression) syntax. Additionally, Vue is reactive, meaning any changes to the data are reflected on the UI, enhancing user experience.

A screenshot of a computer

Description automatically generated

**Figure 1. Challenge for video 1**

### **Video 2**

In this video, data in Vue.js can be tied to HTML attributes using the v-bind: directive or its shorthand :. The attribute to which the data is bound is specified by the attribute name that follows the :. The data to be bound is placed inside the quotations. This method allows the properties to be dynamically updated based on the data.A screenshot of a computer screen

Description automatically generated

**Figure 2. Challenge for video 2**

### **Video 3**

In this lesson, Vue provides directives like v-if, v-else-if, v-else, and v-show to conditionally display elements. When the condition inside the parentheses is true, the element will be shown. Expressions can be used within these parentheses. The key difference between v-if and v-show is that v-show only toggles the visibility of the element by changing its display style without removing it from the DOM. On the other hand, v-if completely adds or removes the element from the DOM based on the condition, which can help optimize performance when necessary.A screenshot of a computer screen

Description automatically generated

**Figure 3. Challenge for video 3**

### **Video 4**

Video 4 talk about that the v-for directive in Vue.js allows us to iterate over an array, using an alias to represent each item, such as v-for="content in contents". It can loop through objects, displaying their properties with dot notation. Providing a unique key for each element improves performance.

A screenshot of a computer

Description automatically generated

**Figure 4. Challenge for video 4**

## **Appendices 2**

### **Video 5**

In video 5, it talks about elements in Vue.js may listen for events like clicks and mouseovers thanks to the v-on directive. @ is the shorthand for v-on. A method that takes parameters can be called when an event happens. This allows for dynamic interaction with the element by referencing the data and methods of the current Vue instance inside the method.

A screenshot of a computer

Description automatically generated

**Figure 5. Challenge for video 5**

### **Video 6**

In this lesson, it is shown that data can be bound to an element's style or class attributes, allowing for dynamic styling based on data. Additionally, expressions can be used within class bindings to conditionally apply certain classes, depending on the evaluated result. This feature enables a more interactive and responsive design that adapts to changing data.A screenshot of a computer

Description automatically generated

**Figure 6. Challenge for video 6**

### **Video 7**

In Vue.js, computed properties are used to compute values based on application data rather than storing them. They allow dynamic values to be generated from existing data, enhancing data processing and presentation by automatically updating whenever the underlying data changes.

A screenshot of a computer

Description automatically generated

**Figure 7. Challenge for video 7**

## **Appendices 3**

### **Video 8**

In this lesson, it is explained that Vue.js components break down programs into reusable parts, each with its own structure and behavior, making applications easier to manage. A component's data must be defined as a function, but props allow data to flow from parent to child components. Additionally, props can be validated and dynamically bound. The Vue DevTools provide valuable insights into the behavior and structure of components.A screenshot of a computer

Description automatically generated

**Figure 8. Challenges for video 8**

### **Video 9**

In this lesson, components can notify their parent of events by using $emit, which allows the parent to listen for these events using the v-on directive (or its shorthand @). When an event is emitted, the component can send data along with it. This enables the parent to respond to the event and use the emitted data to trigger methods or update the state.A screenshot of a computer

Description automatically generated

**Figure 9. Challenge for video 9**

### **Video 10**

In video 10, the v-model directive in Vue.js enables two-way data binding for form elements. Although it might have some issues, the. number modifier ensures that values are rendered as numbers. To prevent page reloads during form submission, the. prevent modifier is used. Additionally, Vue provides basic custom form validation.A screenshot of a computer

Description automatically generated

**Figure 10. Challenge for video 10**

## **Appendices 4**

A screenshot of a computer

Description automatically generated

**Figure 11. Lesson 4 exercise 1**

A screenshot of a computer

Description automatically generated

**Figure 12. Lesson 4 exercise 3**

A screenshot of a computer

Description automatically generated

**Figure 13. Lesson 5 exercise 3**

A screenshot of a computer

Description automatically generated

**Figure 14. Lesson 6 exercise 2**

A sort of database architecture that deviates from conventional relational databases is the NoSQL database. In contrast to relational databases, which arrange data using tables, rows, and columns, NoSQL databases provide a versatile paradigm for storing semi-structured or unstructured data. Large data volumes, heavy user loads, and quickly evolving data architectures are all things that these databases are designed to manage. They are perfect for real-time online applications, big data analytics, and cloud computing since they are frequently employed in scenarios where scalability and performance are crucial. Key-value stores, document stores, column-family stores, and graph databases are examples of common NoSQL database types.

One of the most well-known NoSQL databases is MongoDB, which is distinguished by its document-oriented architecture. Unlike conventional relational databases, it stores data as flexible documents that resemble JSON, allowing for more dynamic schemas. With MongoDB, documents may have distinct sets of fields, giving developers a great deal of freedom. Because MongoDB is intended to be horizontally scalable, it can manage massive amounts of data by splitting them across several servers. It is commonly employed in contemporary applications where real-time processing, scalability, and agility are crucial. Because of its design, MongoDB is a flexible option for a wide range of businesses. It can manage many sorts of unstructured data, including texts, logs, and complex objects, and it can retrieve data more quickly.

## **Appendices 5**

**A screenshot of a computer

Description automatically generated**

**Figure 15. Home page**

**A screenshot of a computer

Description automatically generated**

**Figure 16. Student page**

**A screenshot of a computer

Description automatically generated**

**Figure 17. Admin page**

**A screenshot of a computer

Description automatically generated**

**Figure 18. Data page**

**A screenshot of a computer

Description automatically generated**

**Figure 19. Invalid request**

A strong platform for creating server and network applications is Node.js. Making a web server is one of its most popular uses. During the learning process, the user investigated how to utilize the built-in http module in Node.js to develop and run a basic server. With the help of this module, you may handle HTTP requests and react appropriately by using the provided URL.  
Because Node.js is event-driven and asynchronous, it can process numerous requests at once without being paused by other processes. Because of this, it's a useful tool for creating scalable online apps and real-time applications.  
Specifically, in this task, the user wrote a simple code that creates a Node.js server, which listens on port 8000. Depending on the different requested URLs, such as the home page, the student page, or the admin page, the server returns various responses ranging from HTML messages to JSON data. Additionally, the user learned that Node.js can return appropriate HTTP status codes to describe the state of the request, such as a 200 status code for success.  
A key takeaway is that Node.js is lightweight and fast, thanks to its asynchronous event-driven architecture. This helps optimize performance in applications with a large and continuous volume of requests, making it an ideal choice for handling scalable server-side operations efficiently.

## **Appendices 6**

**A screenshot of a computer

Description automatically generated**

**Figure 20. Model code**

**A screen shot of a computer

Description automatically generated**

**Figure 21. Server is running on port 3000**

**A screenshot of a computer

Description automatically generated**

**Figure 22. Api connected show data**

**A screenshot of a computer

Description automatically generated**

**Figure 23. Studio 3T show data**

This lesson offers a comprehensive insight into the development of a fundamental RESTful API utilizing Node.js, Express, and MongoDB. A RESTful API constitutes a well-defined framework that facilitates interaction between a client and a server through HTTP methods like GET, POST, PUT, and DELETE. This architectural pattern promotes the design of scalable and efficient web services.

The journey commences with the establishment of the environment, necessitating the installation of crucial modules, which include express for managing HTTP requests, mongoose for interacting with the MongoDB database, and body-parser for interpreting incoming request bodies. Furthermore, nodemon is incorporated to ensure the server restarts automatically whenever modifications are made, enhancing the development process.

Once the environment has been set up, essential project files are generated. The server.js file plays a crucial role in initializing the server and specifying the port for incoming connections. The model file utilizes Mongoose to define the data schema, detailing how the data will be organized within the MongoDB database. Subsequently, routes are established to manage the different HTTP methods, enabling CRUD (Create, Read, Update, Delete) operations to be executed on the data.

The API is evaluated using tools like Postman, which ensures that the communication between the client and server operates smoothly. The final stage entails linking the API to the MongoDB database, whether it be locally or via a university-hosted instance. By accomplishing these tasks, the lesson emphasizes the significance of grasping RESTful principles, data organization, and asynchronous communication in contemporary web development.