**Instructions**:

Please answer the following questions and complete the tasks. You may use any libraries or frameworks you prefer for authentication and authorization, but be prepared to explain your choices.

**Section 1: General Node.js Knowledge**

1.1. What is Node.js, and how does it differ from traditional server-side technologies like PHP or Java?

Ans: Node js is serverside scripting language, which is use to write java script code on server side. Node.js uses an event-driven, non-blocking I/O model, which is different from the multi-threaded model like java.

Due to its non-blocking I/O model and asynchronous programming, Node.js can handle a large number of concurrent connections efficiently, making it well-suited for building real-time applications such as chat applications, streaming services, and APIs  
it is differ from other tech like it asynchronous,

1.2. Explain the concept of event-driven programming in Node.js and provide an example.

Ans: nodejs is single threaded ,non-blocking I/o model,

So whenever a request comes to nodejs server it send to event queue   
after that if it is synchoruns request then then it will put it event loop or if it is async task then it directly process that data.  
synchronous tasks are put into thread pool and assign worker to it.  
who work on the task and return result.  
can explain more on call

1.3. Describe the purpose of the package.json file in a Node.js project.  
ANS: it is a heart on nodejs project which contains all the list of packages which are used to run a nodejs project.also we can write some scripts to run project.

1.4. What is callback hell, and how can it be mitigated in Node.js applications?

ANS: we are calling callback within a callback is a callback hell just like nested loops,  
we can use async and await and promises to avoid callback hell.

1.5. What is npm, and why is it commonly used in Node.js projects?  
ANS: it is a node package manager, commonly used to install packages for the nodejs project  
npm have wast number of packages on internet, and it is continously growing more,  
there is another package manager yarn.

**Section 2: CRUD Operations**

2.1. Create a simple Node.js server using Express that listens on port 3000.

2.2. Set up routes for creating, reading, updating, and deleting resources (CRUD) for a hypothetical "Task" entity.

2.3. Implement in-memory storage for tasks and handle the CRUD operations using routes.

**Section 3: Multiple File Uploading**

3.1. Create an API endpoint to handle file uploads. The endpoint should accept multiple files in a single request.

3.2. Explain how you would handle file storage and organization on the server.

3.3. Provide an example of a request using a tool like curl to upload multiple files to your API.

//curl example

curl -X POST \

-H "Authorization: Bearer YOUR\_ACCESS\_TOKEN" \

-F "files=@/path/to/file1.txt" \

-F "files=@/path/to/file2.jpg" \

<http://localhost:3000/upload>

**Section 4: Authentication**

4.1. Implement user authentication using a library of your choice (e.g., Passport.js, jsonwebtoken).

4.2. Create a registration endpoint that allows users to sign up with a username and password.

4.3. Implement login functionality using JSON Web Tokens (JWT).

4.4. Describe how you would securely store user passwords in a database.  
Ans: I have use bcrypt to to encrypt password string into encrypted data   
it is one way method so we can not decrypt it once it is encrypted, we have to use brut force to decrypt it. So we can store this encrypted data into db.

4.5. Provide a sample API request that demonstrates the user registration and login process.

Video recording provided

**Section 5: Authorization**

5.1. Define the concept of authorization in the context of a Node.js application.  
ANS: authorization means user have that particular role/ authority to access that data.  
in nodejs we are assigning some role to user so that they can have limited accessibility to app or functionality of product

5.2. Implement role-based authorization for your "Task" entity, allowing only authenticated users to create and update tasks while restricting the deletion of tasks to administrators.

5.3. Provide an example of how you would check and enforce authorization in your routes.  
we can add middle for it   
but in this code I have added in only single api for admin role  
if it has admin role then you can add task.

**Section 6: Bonus Task**

6.1. Implement user authentication using a different library or method than the one you used in Section 4. Compare and contrast the two approaches, highlighting their advantages and disadvantages.  
for this I have shared one code base in this folder

**Section 7: Conclusion**

7.1. Summarize the key takeaways and challenges you encountered while completing this technical test paper.  
we can use any database to store tasks and user but as per instruction I have to use in storage as a variable in the code.

So if the file stop code it will erase all data in that variables.   
To overcome this problem we can store data in to json file and can access that file as variable and use it as storage

7.2. Provide any additional comments or suggestions for improving the Node.js application you've developed during this test.

we can use any database to store tasks and user but as per instruction I have to use in storage as a variable in the code.So if the file stop code it will erase all data in that variables.   
To overcome this problem we can store data in to json file and can access that file as variable and use it as storage.  
we can convert this code into MVC structure for better view and accessibility for code which is mentioned in that zip code file. All the code is in single file so we can define routes separetly in another folder same for models and controllers also.  
we can enable cors when we deploy this code any cloud provide aws/azure/google.

**Submission**: Please submit your test paper along with the code for your Node.js application. Ensure that your application is well-documented, and any dependencies are clearly listed in the package.json file. Include instructions on how to run your application.