**#### Section B: Short Answer Questions**

**\*\*6. Explain the Purpose of `express.Router()` in the Code Above.\*\***

- Why is `express.Router()` used in Express.js applications, and how does it benefit the code structure?

In Express.js applications, express.Router() is used to create modular, mountable route handlers.

Here are some key benefits and reasons for using it:

1. **Modularity**: Using Router allows you to define routes in separate files
2. **Reusability**: You can create reusable router modules
3. **Middleware Support**: Routers can use middleware specific to a group of routes.
4. **Cleaner Code**

**\*\*7. Error Handling in Express.js\*\***

- How would you implement error handling in the Express routes to ensure that any issues (such as file not found or server errors) are appropriately handled? Provide an example.

---

I would use Middleware for Error Handling

Example:

const express = require('express');

const app = express();

const fs = require('fs');

// Sample route to read a file

app.get('/file/:filename', (req, res, next) => {

const fileName = req.params.filename;

fs.readFile(fileName, 'utf8', (err, data) => {

if (err) {

// Pass the error to the error handling middleware

return next(err);

}

res.send(data);

});

});

// Sample route that simulates a server error

app.get('/error', (req, res, next) => {

// Simulate an unexpected server error

const error = new Error('Unexpected server error!');

error.status = 500; // Custom error status code

next(error); // Pass the error to the error handler

});

// Error handling middleware

app.use((err, req, res, next) => {

console.error(err.stack); // Log the error stack for debugging

// Set the response status and send a JSON response

res.status(err.status || 500); // Default to 500 if no specific status

res.json({

status: 'error',

message: err.message || 'Internal Server Error',

});

});

// Start the server

app.listen(3000, () => {

console.log('Server is running on port 3000');

});

**#### Section C: Bonus**

**\*\*7. Dynamic Port Binding in Express.js\*\***

- Explain how the `app.listen(process.env.port || 8081)` line works and why it's useful in production environments.

process.env.PORT used to store configuration settings outside of the code

The expression process.env.PORT || 8081 uses the logical OR (||) operator. If process.env.PORT is undefined or falsey, it falls back to 8081. This is a default port that can be used for local development.

app.listen(...) starts the Express application and makes it listen for incoming requests on the specified port.

**Why It’s Useful in Production Environments**

1. **Flexibility**
2. **Consistency**
3. **Ease of Deployment**
4. **Avoiding Conflicts**

**# Submission Guideline**

Process of creating a ZIP file, GitHub repository, and gathering screenshots as per your instructions. Here’s the step-by-step guide to complete the exercise: