Task: Log Request Information to a File Using fs Module in Express.js

Objective: Create an Express.js application that logs essential request information (timestamp, IP, URL, protocol, HTTP method, and hostname) to a file using the fs (File System) module. This task will help you understand middleware in Express.js and how to interact with the file system in Node.js.

Problem 1: Set Up the Express.js Application

1. **Install Dependencies:**
   - Initialize a new Node.js project:
   - npm init -y
   - Install Express:
   - npm install express
2. **Create the Main Server File:**
   - Create a file named server.js:
   - const express = require('express');
   -
   - const app = express();
   -
   - // Start the server on port 3000
   - const PORT = 3000;
   - app.listen(PORT, () => {
   -     console.log(`Server is running on http://localhost:${PORT}`);
   - });
3. **Run the Server:**
   - Start the server using:
   - node server.js
   - Verify it runs by visiting http://localhost:3000 in your browser.

Problem 2: Implement Middleware to Capture Request Details

Add middleware to log request details:

const fs = require('fs'); // Import the File System module

const path = require('path');

```
// Middleware to log request details
app.use((req, res, next) => {
  const logDetails = {
    timestamp: new Date().toISOString(),
    ip: req.ip,
    url: req.originalUrl,
    protocol: req.protocol,
    method: req.method,
    hostname: req.hostname
  };

  // Convert log details to a JSON string
  const logEntry = JSON.stringify(logDetails);

  // Call the next middleware in the chain
  next();
});
```

Problem 3: Use the fs Module to Write Request Details to a File

Update the middleware to write log details to requests.log using fs.appendFile:

```
app.use((req, res, next) => {
  const logDetails = {
    timestamp: new Date().toISOString(),
    ip: req.ip,
    url: req.originalUrl,
    protocol: req.protocol,
    method: req.method,
    hostname: req.hostname
  };

  const logEntry = JSON.stringify(logDetails);
  // Append the log entry to the file
  fs.appendFile(
    path.join(__dirname, 'requests.log'),
    logEntry + '\n',
    (err) => {
      if (err) {
        console.error('Failed to write log entry:', err);
      }
    }
  );

  next(); // Call the next middleware
});
```

Problem 4: Test the Logging Functionality

1. **Start the Server:**
2. node server.js
3. **Make Test Requests:**
   - Use a browser, Postman, or cURL to send requests to your server. Example:
   - curl http://localhost:3000/test
4. **Verify the Log File:**
   - Open the requests.log file and verify the log entries. Each entry should look like this:
   - {"timestamp":"2025-01-21T12:34:56.789Z","ip":"::1","url":"/test","protocol":"http","method":"GET","hostname":"localhost"}

Problem 5: Optional Advanced Features

Log Rotation:

Rotate the log file when it reaches a certain size:

1. Install the fs-extra package:
2. npm install fs-extra

3. Modify the logging middleware to check file size and rotate logs:

```
const fse = require('fs-extra');
const MAX_LOG_SIZE = 1024 * 1024; // 1MB

app.use((req, res, next) => {
  const logDetails = {
    timestamp: new Date().toISOString(),
    ip: req.ip,
    url: req.originalUrl,
    protocol: req.protocol,
    method: req.method,
    hostname: req.hostname
  };

  const logEntry = JSON.stringify(logDetails) + '\n';
  const logPath = path.join(__dirname, 'requests.log');

  fs.stat(logPath, (err, stats) => {
    if (!err && stats.size >= MAX_LOG_SIZE) {
      const timestamp = new Date().toISOString().replace(/[:.]/g, '-');
      const archivePath = path.join(__dirname, `requests-${timestamp}.log`);
      fse.move(logPath, archivePath, (err) => {
        if (err) {
          console.error('Failed to rotate logs:', err);
        } else {
          fs.writeFile(logPath, logEntry, (err) => {
            if (err) console.error('Failed to write new log entry:', err);
          });
        }
      });
    } else {
            fs.appendFile(logPath, logEntry, (err) => {
                if (err) console.error('Failed to append log entry:', err);
            });
        }
    });

    next();
});
```

Enhanced Logging:

Add more details to the log, such as:

- **Query Parameters:**
- logDetails.query = req.query;
- **Request Headers:**
- logDetails.headers = req.headers;
- **User-Agent:**
- logDetails.userAgent = req.get('User-Agent');

Summary: This task guides you through setting up an Express.js application, implementing middleware to log request details, and writing logs to a file using the fs module. Optional features like log rotation and enhanced logging provide additional functionality for a more robust logging system.