Assignments

## 1) Mini File Management Tool (CRUD via CLI)

**Goal:** CLI tool to create, read, update (append/overwrite), list, and delete text files in a `notes/` folder.

Save as `file-manager.js`. Run with `node file-manager.js <command> [args]`.

```
async function main() {
  await ensureNotesDir();
  const [,, cmd, name, ...rest] = process.argv;

  try {
    switch (cmd) {
      case 'create':
        await createNote(name, rest.join(' ') || '');
        break;
      case 'read':
        await readNote(name);
        break;
      case 'append':
        await appendNote(name, rest.join(' '));
        break;
      case 'overwrite':
        await overwriteNote(name, rest.join(' '));
        break;
      case 'delete':
        await deleteNote(name);
        break;
      case 'list':
        await listNotes();
        break;
      default:
        console.log('Usage: node file-manager.js
<create|read|append|overwrite|delete|list> <name> [content]');
    }
  } catch (err) {
    console.error('Error:', err.message);
  }
}

main();
```

**Examples:**

```
node file-manager.js create meeting "Notes from stand-up"
node file-manager.js append meeting "Added action items"
node file-manager.js read meeting
node file-manager.js list
node file-manager.js delete meeting
```

## 2) Node.js Server that Serves JSON Data from File

**Goal:** HTTP server that reads a JSON file and returns JSON for an API endpoint. Use `fs.promises` and simple routing.

Save as `json-server.js` and create `data/users.json`.

`data/users.json`:

```
[
  { "id": 1, "name": "Alice" },
  { "id": 2, "name": "Bob" }
]
```

`json-server.js`:

**Run & test:**

```
node json-server.js
curl http://localhost:3000/api/users
```

## 3) CLI: Explore `path` and `os` Modules (System Information)

**Goal:** Small CLI that prints path utilities and OS info.

Save as `sys-info.js` and run `node sys-info.js`.

## 4) File Operations with Promises — Notes App Mini-API (CLI + JSON DB)

**Goal:** Use `fs.promises` to implement a JSON-based note storage with simple commands.

Save as `notes-api.js`. Usage examples follow.

```
// notes-api.js
const fs = require('fs').promises;
const path = require('path');

const DB = path.join(__dirname, 'notes-db.json');

async function loadDB() {
  try {
    const txt = await fs.readFile(DB, 'utf8');
    return JSON.parse(txt || '[]');
  } catch (err) {
    // If file doesn't exist, return empty array
    if (err.code === 'ENOENT') return [];
    throw err;
  }
}

// complete code

// CLI dispatch
(async function(){
  const [,, cmd, arg1, ...rest] = process.argv;
  try {
    switch(cmd) {
      case 'add':
        await addNote(arg1 || 'Untitled', rest.join(' ') || '');
        break;
      case 'list':
        await listNotes();
        break;
      case 'get':
        await getNote(arg1);
        break;
      case 'delete':
        await deleteNote(arg1);
        break;
      default:
        console.log('Usage: node notes-api.js <add|list|get|delete>
[args]');
    }
  } catch (err) {
    console.error('Error:', err.message);
  }
})();
```

**Examples:**

```
node notes-api.js add "Todo" "Finish Node.js workshop"
node notes-api.js list
node notes-api.js get 1
node notes-api.js delete 1
```

## 5) Exercise — Implement `/time` route

**Goal:** Extend the earlier HTTP server to include `GET /time` to return current time.

```
// time-server.js (extends basic routing)
const http = require('http');

//complete code

server.listen(3001, () => console.log('Time server on
http://localhost:3001'));
```

**Test:**

```
curl http://localhost:3001/time
# => {"now":"2025-10-24T10:12:34.000Z"}
```