**Walchand College of Engineering, Sangli**
**Department of Computer Science and Engineering**

# Practical No. 8
# Study and implementation of node.js

**https://github.com/themonstersd13/WddlExp8**
**Problem Statement 1: Introduction to Node.js**

1. **What is Node.js, and how does it differ from traditional server-side platforms like Apache or PHP?**
   Node.js is an asynchronous, event-driven JavaScript runtime designed for building scalable network applications Node.js — Run JavaScript Everywhere. It differs from platforms like Apache or PHP—which use a thread-per-request and blocking I/O model—by employing a single-threaded, non-blocking architecture that can handle many concurrent connections within one process zibtek.comNode.js — Run JavaScript Everywhere.

2. **What is the purpose of the V8 engine in Node.js?**
   Node.js uses Google's V8 engine to compile JavaScript into machine code at runtime, providing high-performance execution outside of the browser GeeksforGeeks. V8 is integral to Node.js and cannot be replaced, as it underlies all server-side JavaScript execution GeeksforGeeks.

3. **Explain the single-threaded, event-driven architecture of Node.js.**
   Node.js executes JavaScript on a single main thread via an event loop that delegates I/O operations to the system kernel or an internal thread pool and invokes callbacks when those operations complete Medium.

4. **Why is Node.js considered non-blocking?**
   All I/O operations in Node.js are non-blocking: when an operation starts, Node.js registers a callback and immediately continues processing other events, running the callback only after the operation finishes Node.js — Run JavaScript Everywhere.

5. **What is npm, and how is it used in Node.js?**
   npm is the default package manager for Node.js, used to install, update, and manage dependencies from the npm registry via commands like `npm install package-name` Node.js — Run JavaScript Everywhere. It hosts over two million packages, making it the largest code repository of its kind Node.js — Run JavaScript Everywhere.

6. **What is a module in Node.js? How do you export and import modules?**
   In Node.js, a module is any file or directory that encapsulates code. Developers export functionality using `module.exports` or `exports.foo` in CommonJS or `export` in ES Modules, and they import modules via `require('./module')` or `import` statements SitePoint. Modules are defined as files exposing public APIs and are consumed by other files via these mechanisms npm Docs.

7. **What is the difference between require() and import in Node.js?**
   `require()` is the CommonJS loader that synchronously loads modules based on `module.exports`, while `import`/`export` are the ECMAScript Module syntax, which supports static analysis and may be asynchronous; using `import` requires setting `"type": "module"` in `package.json` or using the `.mjs` extension Medium.

8. **How can you create a custom module in Node.js?**
   Create a file (e.g., `greet.js`) and assign your functionality to `module.exports`, for example:
   function greet(name) {
   return `Hello, ${name}!`;
   }
   module.exports = greet
   Then load it elsewhere with:
   const greet = require('./greet')
   console.log(greet('Alice')) npm Docs.

9. **What is the role of the package.json file in a Node.js project?**
   The `package.json` file defines project metadata (name, version, description), lists `dependencies` and `devDependencies`, and declares `scripts` (such as `"start"` and `"test"`), guiding npm on how to install and run the project npm Docs.

10. **How do you install a package globally and locally using npm?**
    To install locally (for a project), run `npm install package-name`, which adds it to `node_modules` and updates `package.json`. To install globally (system-wide executables), run `npm install -g package-name` Mediumnpm Docs.

11. **What is the difference between asynchronous and synchronous programming in Node.js?**
    Synchronous functions block the event loop until they complete, whereas

asynchronous functions initiate operations and return immediately, allowing the event loop to continue processing other tasks and invoking callbacks upon completion Stack OverflowGeeksforGeeks.

12. **How do you create an HTTP server in Node.js?**
    Use the built-in `http` module:
    ```
    const http = require('http')
    const server = http.createServer((req, res) => {
    res.writeHead(200, {'Content-Type': 'text/plain'})
    res.end('Hello World')
    })
    server.listen(3000)
    ```
    Node.js — Run JavaScript Everywhere.

13. **What is the difference between http.createServer() and using frameworks like Express.js?**
    `http.createServer()` provides a minimal, low-level HTTP server requiring manual routing and parsing of requests, whereas Express.js builds on it to offer declarative routing, middleware support, built-in body parsing, and many utilities out of the box The freeCodeCamp Forum.

14. **How do you handle GET and POST requests in Node.js?**
    In a raw HTTP server, check `req.method === 'GET'` and `req.url` to handle GET. For POST, check `req.method === 'POST'`, listen for `req.on('data', chunk)` to collect body data, concatenate the chunks, then process in `req.on('end')` before sending a response

**Problem Statement 2: Middleware (Express.js)**

### 1.What Is Middleware in Node.js / Express.js?

Middleware functions are functions that have access to the request object (`req`), the response object (`res`), and the next middleware function in the application's request–response cycle Express.

They can execute any code, make changes to `req` and `res`, end the request–response cycle, or call `next()` to pass control to the next matching middleware Express.

### 2.How to Create Custom Middleware in Express.js

A custom middleware is simply a function defined with three parameters: `(req, res, next)` DigitalOcean.

Within the function you can inspect or modify `req` and `res`, then either end the cycle (e.g., `res.send()`) or call `next()` to hand off control DigitalOcean.

Example (`logger.js`):

```
// logger.js
module.exports = function logger(req, res, next) {
  console.log(`${req.method} ${req.url}`);
  next();
};
```

Register it in your app with `app.use()` (application-level) or `router.use()` (router-level) GeeksforGeeks.

### 3.Execution Order of Middleware in Express.js

Express evaluates middleware and routes in the order they are registered with `app.use()`, `app.METHOD()`, or `router.use()`, so placement in your code determines the execution sequence Stack Overflow.

Once a middleware calls `next()`, control moves to the very next matching middleware; if it sends a response without calling `next()`, no subsequent handlers run Stack Overflow.

before more specific ones (e.g., `app.use('/api', ...)`) can preempt them, since `/:id` will match `/api` first Reddit.

**Problem Statement 3: File System (fs) Module**

**1.How do you read and write files using the fs module in Node.js?**

Asynchronous reading:

Use `fs.readFile(path, [encoding], callback)` to read a file without blocking. The callback receives `(err, data)`, where `data` is a `Buffer` or string. Node.js — Run JavaScript Everywhere

```
const fs = require('fs');
fs.readFile('example.txt', 'utf8', (err, data) => {
  if (err) throw err;
  console.log(data);
});
```

Asynchronous writing:

Use `fs.writeFile(path, data[, options], callback)` to write data to a file. If the file does not exist, it is created. Node.js — Run JavaScript Everywhere

```
const fs = require('fs');
const content = 'Hello, World!';
fs.writeFile('output.txt', content, err => {
  if (err) throw err;
  console.log('File written successfully');
});
```

- Synchronous methods:
  The module also offers `fs.readFileSync(path, [encoding])` and `fs.writeFileSync(path, data[, options])`, which return results directly but block the event loop until finished. Stack Overflow

**2.What is the difference between fs.readFile() and fs.readFileSync()?**

- `fs.readFile()` is asynchronous: it issues the I/O operation, registers a callback, and immediately returns control to the event loop; the callback is

invoked when data is ready. Stack Overflow

- `fs.readFileSync()` is synchronous: it reads and returns the file's contents before moving to the next line of code, blocking the entire thread until the read completes. Stack Overflow

**3.How can you check if a file or directory exists in Node.js?**

- Deprecated: `fs.exists(path, callback)` and `fs.existsSync(path)` exist but are discouraged due to race-condition issues. Node.js — Run JavaScript Everywhere

- Recommended:

  - `fs.access(path, fs.constants.F_OK, callback)` to asynchronously check existence/permissions. Node.js — Run JavaScript Everywhere

  - `fs.stat(path, callback)` to retrieve stats (and detect "not found" errors).

```
const fs = require('fs');
fs.access('myfile.txt', fs.constants.F_OK, err => {
  console.log(err ? 'Not found' : 'Exists');
});
```

**4.How do you handle file operations in an asynchronous manner?**

- Callbacks: Pass a callback to the async API; errors are handled inside the callback.

Promises: Use the `fs.promises` API or wrap callback methods with `util.promisify()`. Stack Overflow

```
const fs = require('fs').promises;
async function load() {
  try {
```

```
    const data = await fs.readFile('example.txt', 'utf8');
    console.log(data);
  } catch (err) {
    console.error(err);
  }
}
load();
```

- Streams: For large files, use `fs.createReadStream()` and `fs.createWriteStream()` to process data in chunks without loading the entire file into memory. Honeybadger

- Error handling: Always handle errors either via callback checks, `.catch()`, or `try/catch` in async functions.

**Problem Statement 4: Database Connectivity**

**Q1: How do you connect to a SQL or Oracle database from a Node.js application?**

A:
To connect to a SQL or Oracle database from a Node.js application, you can use specific libraries:

- MySQL: Use the `mysql2` library.

  Install: `npm install mysql2`

Example:

```
const mysql = require('mysql2');
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'your_username',
  password: 'your_password',
  database: 'your_database'
});

connection.connect((err) => {
  if (err) throw err;
  console.log('Connected to MySQL database.');
});
```

- Oracle: Use the `oracledb` library.

  Install: `npm install oracledb`

Example:

```
const oracledb = require('oracledb');
```

```
async function connect() {
  try {
    const connection = await oracledb.getConnection({
      user: 'your_username',
      password: 'your_password',
      connectString: 'localhost/XEPDB1'
    });
    console.log('Connected to Oracle database.');
    await connection.close();
  } catch (err) {
    console.error(err);
  }
}

connect();
```

---

**Q2: What is the purpose of the `mysql2` library in Node.js?**

A:
The `mysql2` library is a Node.js driver for MySQL that provides:

- Improved performance over the older `mysql` library.
- Support for Promises and async/await syntax.
- Prepared statements to prevent SQL injection.
- Connection pooling for efficient resource management.

---

**Q3: How would you perform basic CRUD operations (Create, Read, Update, Delete) using MySQL and Node.js?**

A:
Using the `mysql2` library:

Create (INSERT):

```
connection.query(
  'INSERT INTO users (name, email) VALUES (?, ?)',
  ['John Doe', 'john@example.com'],
  (err, results) => {
    if (err) throw err;
    console.log('User added:', results.insertId);
  }
);
```

Read (SELECT):

```
connection.query(
  'SELECT * FROM users WHERE id = ?',
  [1],
  (err, results) => {
    if (err) throw err;
    console.log('User details:', results);
  }
);
```

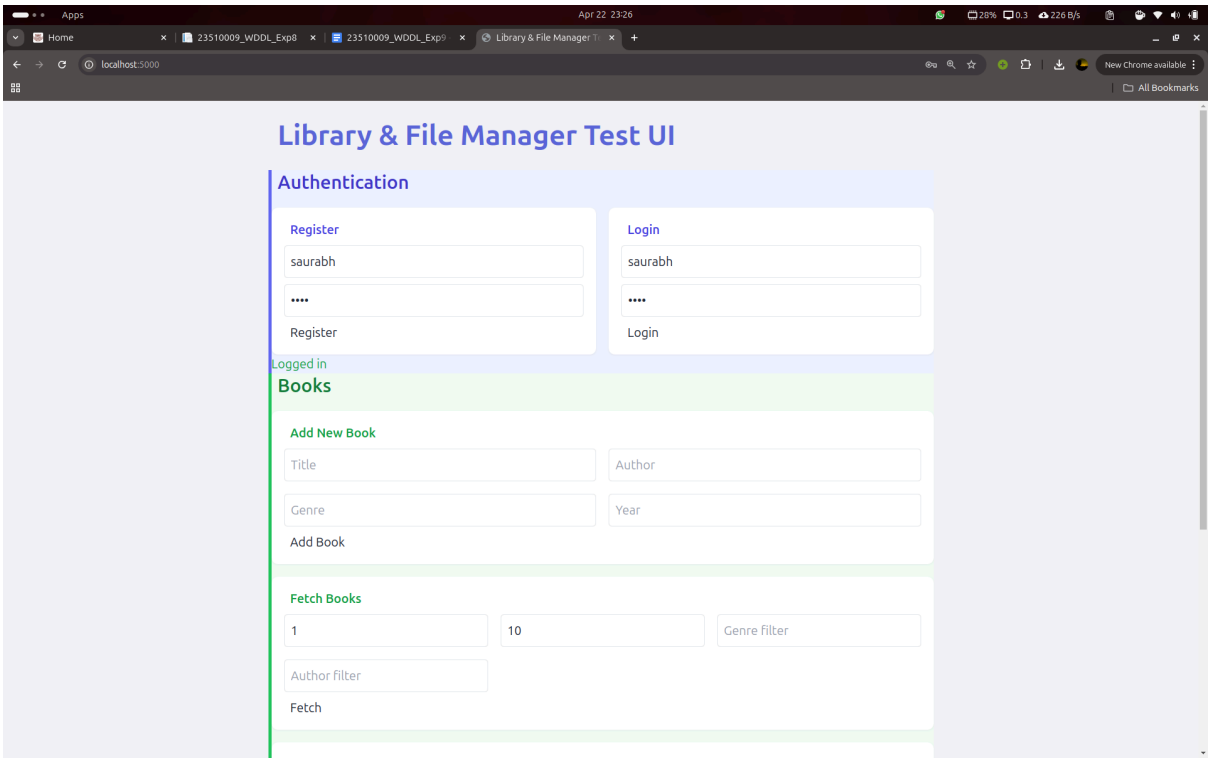
Update (UPDATE):

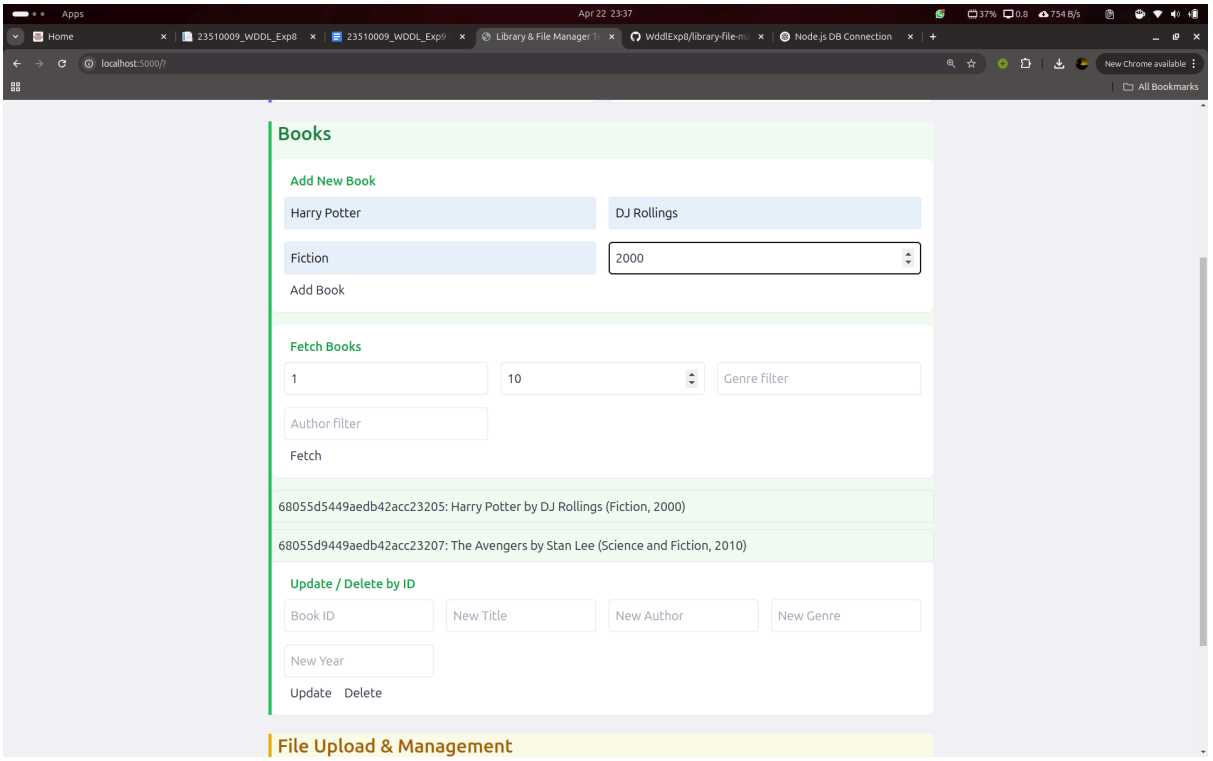
```
connection.query(
  'UPDATE users SET email = ? WHERE id = ?',
  ['newemail@example.com', 1],
  (err, results) => {
    if (err) throw err;
    console.log('User updated:', results.affectedRows);
  }
);
```

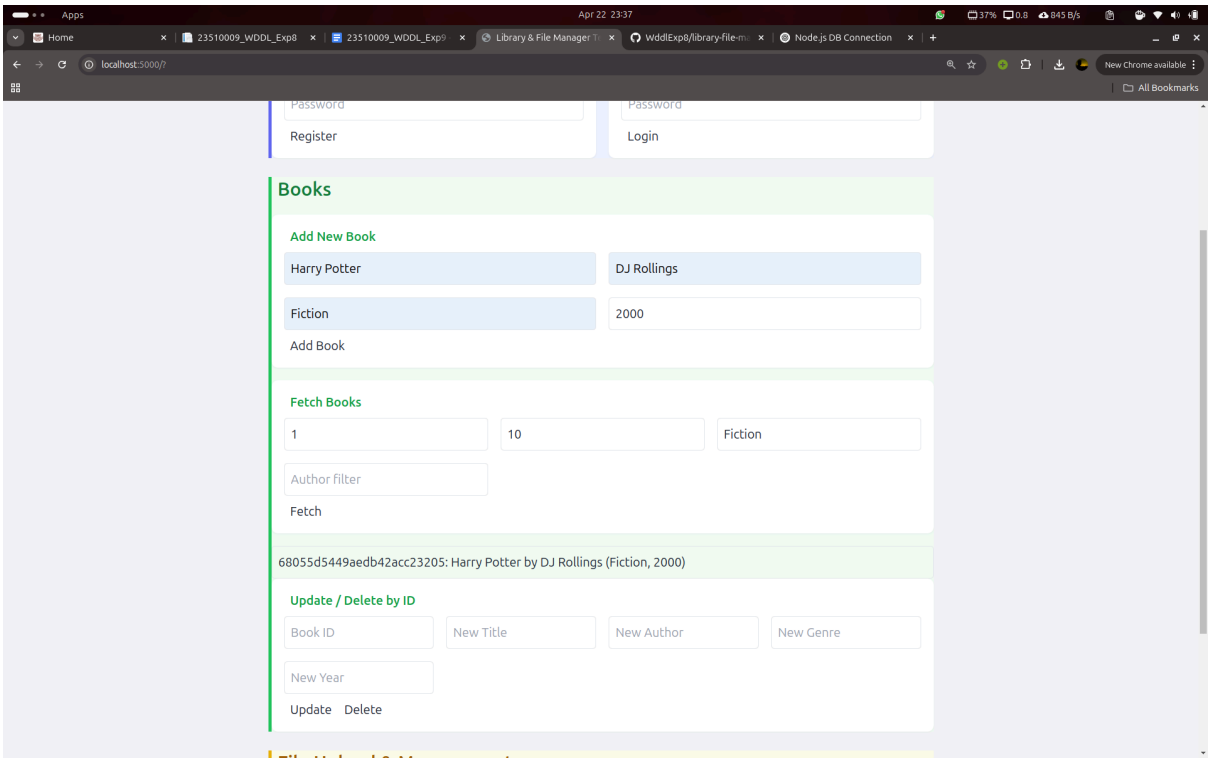
Delete (DELETE):

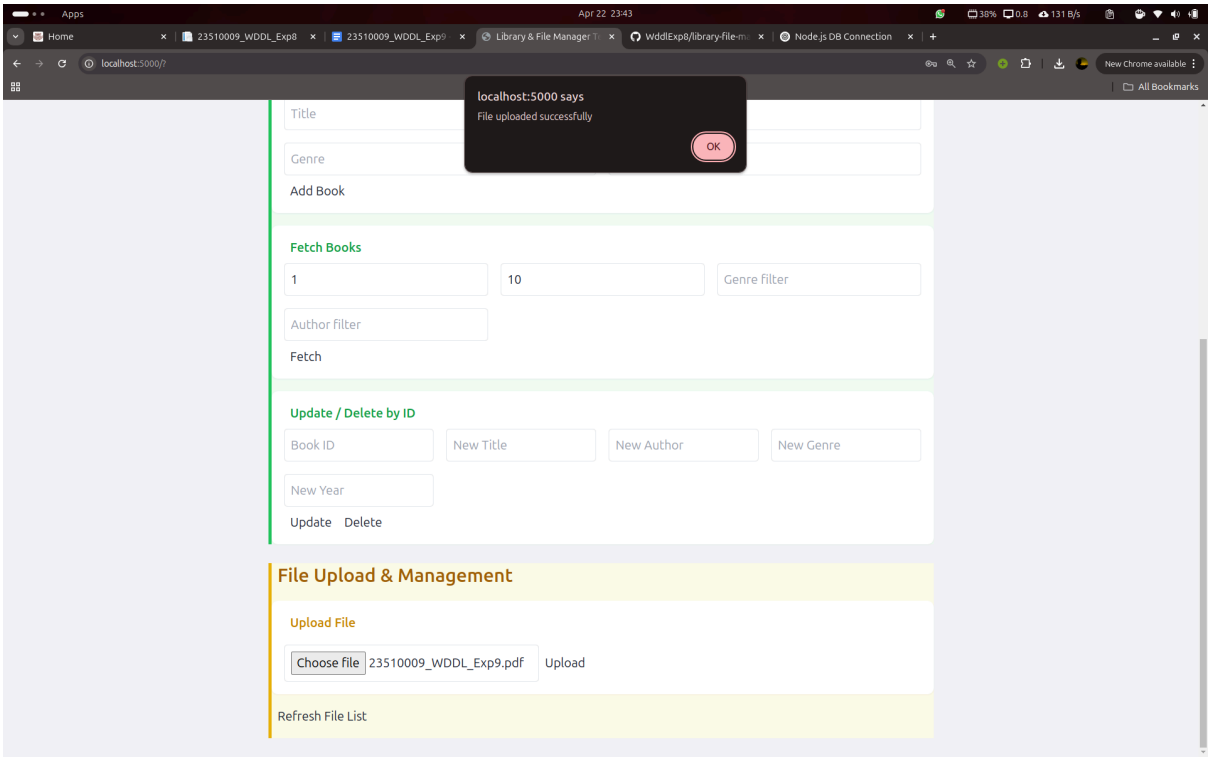
```
connection.query(
  'DELETE FROM users WHERE id = ?',
  [1],
  (err, results) => {
    if (err) throw err;
    console.log('User deleted:', results.affectedRows);
  }
);
```

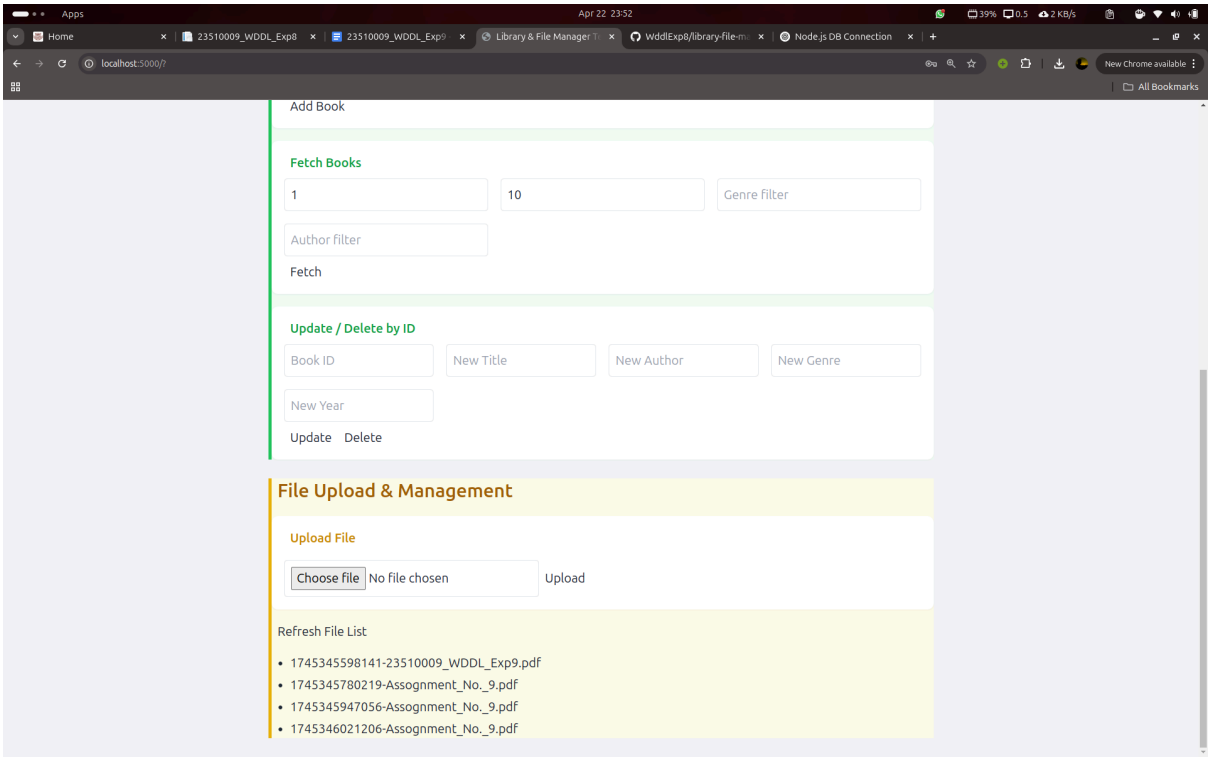
# Problem Statement 5: Building a RESTful API

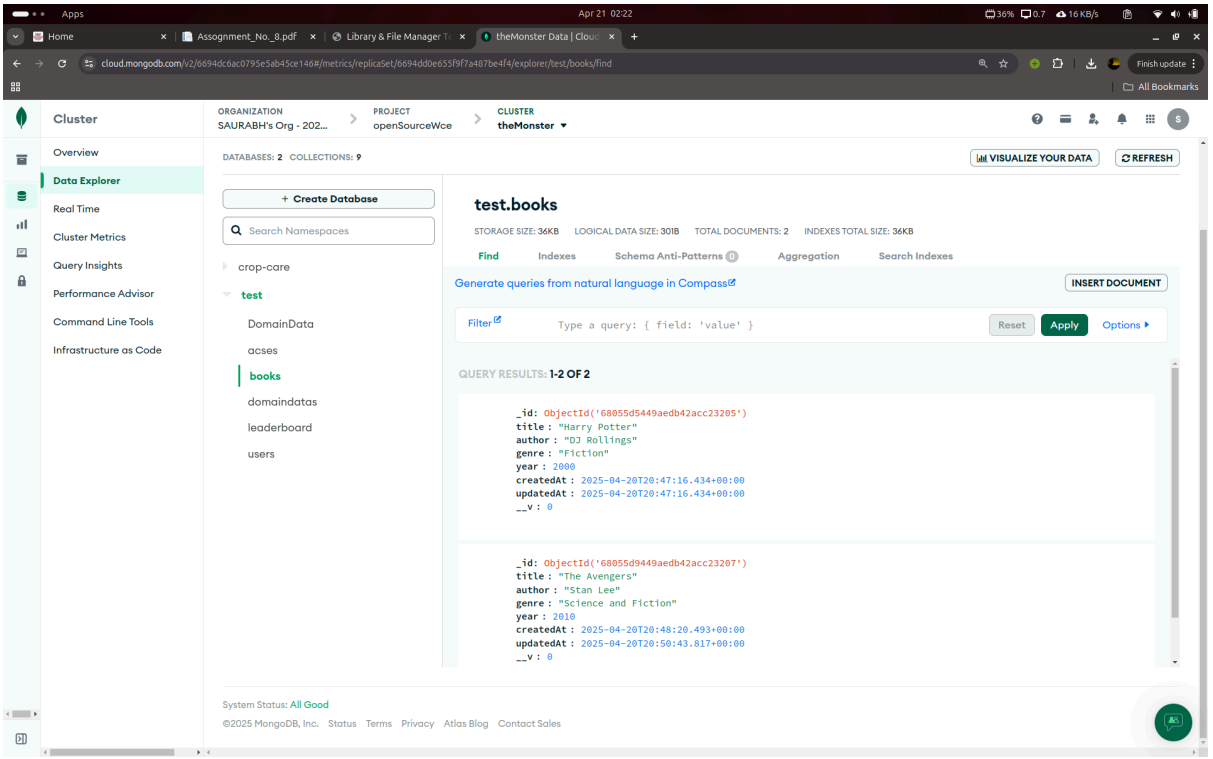


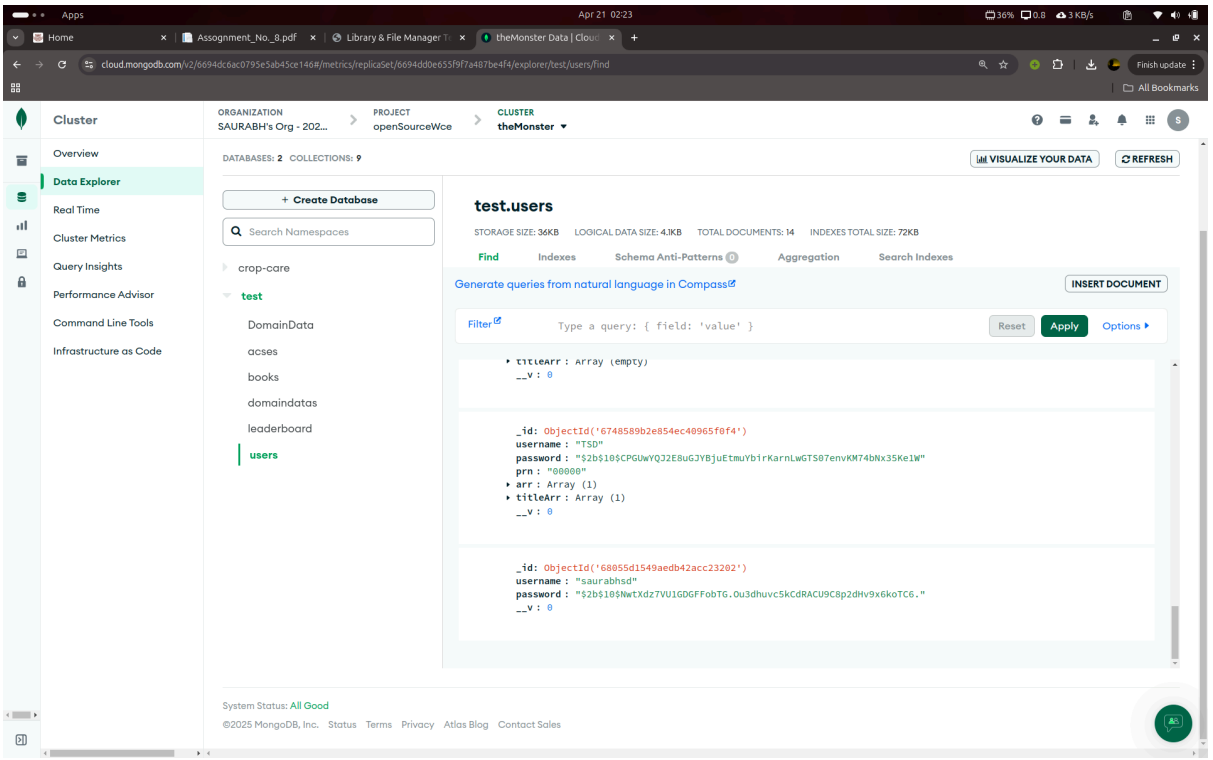

Apps
Apr 22 23:37
Home
23510009_WDDL_Exp8
23510009_WDDL_Exp9
Library & File Manager
WddlExp8/library-file-ma
Node.js DB Connection
localhost:5000/?
New Chrome available
All Bookmarks
Books
Add New Book
Harry Potter
DJ Rollings
Fiction
2000
Add Book
Fetch Books
1
10
Genre filter
Author filter
Fetch
68055d5449aedb42acc23205: Harry Potter by DJ Rollings (Fiction, 2000)
68055d9449aedb42acc23207: The Avengers by Stan Lee (Science and Fiction, 2010)
Update / Delete by ID
Book ID
New Title
New Author
New Genre
New Year
Update
Delete
File Upload & Management


Apps
Apr 22 23:37
Home
23510009_WDDL_Exp8
23510009_WDDL_Exp9
Library & File Manager
WddlExp8/library-file-ma
Node.js DB Connection
localhost:5000/?
New Chrome available
All Bookmarks
Password
Password
Register
Login
Books
Add New Book
Harry Potter
DJ Rollings
Fiction
2000
Add Book
Fetch Books
1
10
Fiction
Author filter
Fetch
68055d5449aedb42acc23205: Harry Potter by DJ Rollings (Fiction, 2000)
Update / Delete by ID
Book ID
New Title
New Author
New Genre
New Year
Update
Delete


localhost:5000 says
File uploaded successfully
OK
Title
Genre
Add Book
Fetch Books
1
10
Genre filter
Author filter
Fetch
Update / Delete by ID
Book ID
New Title
New Author
New Genre
New Year
Update
Delete
File Upload & Management
Upload File
Choose file
23510009_WDDL_Exp9.pdf
Upload
Refresh File List


Add Book
Fetch Books
1
10
Genre filter
Author filter
Fetch
Update / Delete by ID
Book ID
New Title
New Author
New Genre
New Year
Update
Delete
File Upload & Management
Upload File
Choose file
No file chosen
Upload
Refresh File List
1745345598141-23510009_WDDL_Exp9.pdf
1745345780219-Assognment_No._9.pdf
1745345947056-Assognment_No._9.pdf
1745346021206-Assognment_No._9.pdf


Cluster
ORGANIZATION
SAURABH's Org - 202...
PROJECT
openSourceWce
CLUSTER
theMonster
Overview
Data Explorer
Real Time
Cluster Metrics
Query Insights
Performance Advisor
Command Line Tools
Infrastructure as Code
DATABASES: 2 COLLECTIONS: 9
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
crop-care
test
DomainData
acses
books
domaindatas
leaderboard
users
test.books
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 301B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
QUERY RESULTS: 1-2 OF 2
_id: ObjectId('68055d5449aedb42acc23205')
title : "Harry Potter"
author : "DJ Rollings"
genre : "Fiction"
year : 2000
createdAt : 2025-04-20T20:47:16.434+00:00
updatedAt : 2025-04-20T20:47:16.434+00:00
__v : 0
_id: ObjectId('68055d9449aedb42acc23207')
title : "The Avengers"
author : "Stan Lee"
genre : "Science and Fiction"
year : 2010
createdAt : 2025-04-20T20:48:20.493+00:00
updatedAt : 2025-04-20T20:50:43.817+00:00
__v : 0
System Status: All Good
©2025 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales


Cluster
ORGANIZATION
SAURABH's Org - 202...
PROJECT
openSourceWce
CLUSTER
theMonster
Overview
Data Explorer
Real Time
Cluster Metrics
Query Insights
Performance Advisor
Command Line Tools
Infrastructure as Code
DATABASES: 2 COLLECTIONS: 9
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
crop-care
test
DomainData
acses
books
domaindatas
leaderboard
users
test.users
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 4.1KB TOTAL DOCUMENTS: 14 INDEXES TOTAL SIZE: 72KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
titleArr : Array (empty)
__v : 0
_id: ObjectId('6748589b2e854ec40965f0f4')
username : "TSD"
password : "$2b$10$CPGUwYQJ2E8uGJYBjuEtmuYbirKarnLwGTS07envKM74bNx35Ke1W"
prn : "00000"
arr : Array (1)
titleArr : Array (1)
__v : 0
_id: ObjectId('68055d1549aedb42acc23202')
username : "saurabhsd"
password : "$2b$10$NwtXdz7VU1GDGFFobTG.Ou3dhuvc5kCdRACU9C8p2dHv9x6koTC6."
__v : 0
System Status: All Good
©2025 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales