Assignment – 3

User Management System Using Express.js

Tasks:

1. Setup and Initialization:

Create a New Node.js Project

1. Open your terminal or command prompt.
2. Navigate to the directory where you want to create your project.
3. Run the following command to initialize a new Node.js project:

bash

Copy code

```
npm init -y
```

Install Express.js

4. Install Express.js using npm:

bash

Copy code

```
npm install express
```

Ensure the Application Starts Without Errors

5. Create a new file named app.js:

bash

Copy code

```
touch app.js
```

2. Create the Express Server:

6. Open app.js and set up a basic Express server:

javascript

Copy code

```
// app.js
const express = require('express');
const app = express();
const port = 3000;


// Middleware to parse JSON bodies
app.use(express.json());


// Root route
app.get('/', (req, res) => {
  res.send('Welcome to the User Management System');
});


// Start the server
app.listen(port, () => {
  console.log(Server is running on http://localhost:${port});
});
```

7. Run the server:

bash

Copy code

```
node app.js
```

Ensure you see the message Server is running on http://localhost:3000 and visit http://localhost:3000 in your browser to see the welcome message.

3. Define User Routes:

Create a Routes Directory with user.js

8. Create a routes directory:

bash

Copy code

```
mkdir routes
```

9. Create a user.js file inside the routes directory:

bash

Copy code

touch routes/user.js

Define Routes in user.js

10. Open routes/user.js and define the routes:

javascript

Copy code

```
// routes/user.js
const express = require('express');
const router = express.Router();
const fs = require('fs');
const path = require('path');
const usersFilePath = path.join(__dirname, '../data/users.json');

// Helper function to read users from the JSON file
const readUsers = () => {
  const data = fs.readFileSync(usersFilePath, 'utf8');
  return JSON.parse(data);
};

// Helper function to write users to the JSON file
const writeUsers = (users) => {
  fs.writeFileSync(usersFilePath, JSON.stringify(users, null, 2));
};

// GET /users - Retrieve all users
router.get('/users', (req, res) => {
  const users = readUsers();
  res.json(users);
});
```

```
// GET /users/:id - Retrieve a user by ID
router.get('/users/:id', (req, res) => {
  const users = readUsers();
  const user = users.find(u => u.id === parseInt(req.params.id));
  if (!user) {
    return res.status(404).send('User not found');
  }
  res.json(user);
});

// POST /users - Create a new user
router.post('/users', (req, res) => {
  const users = readUsers();
  const newUser = { id: users.length + 1, ...req.body };
  users.push(newUser);
  writeUsers(users);
  res.status(201).json(newUser);
});

// PUT /users/:id - Update a user by ID
router.put('/users/:id', (req, res) => {
  const users = readUsers();
  const userIndex = users.findIndex(u => u.id === parseInt(req.params.id));
  if (userIndex === -1) {
    return res.status(404).send('User not found');
  }
  const updatedUser = { ...users[userIndex], ...req.body };
  users[userIndex] = updatedUser;
  writeUsers(users);
  res.json(updatedUser);
});

// DELETE /users/:id - Delete a user by ID
router.delete('/users/:id', (req, res) => {
  const users = readUsers();
  const newUsers = users.filter(u => u.id !== parseInt(req.params.id));
  writeUsers(newUsers);
  res.status(204).send();
});

module.exports = router;
```

4. User Data Management:

Create a Data Directory with users.json

11. Create a data directory:

bash

Copy code

```
mkdir data
```

12. Create a users.json file inside the data directory and add an empty array:

json

Copy code

```
// data/users.json
[]
```

5. Request Handling and Middleware:

Use Body-Parser Middleware

13.	Body-parser is now built into Express, so it is already included in the setup. Ensure you have the line app.use(express.json()); in your app.js file.

Validate User Input

14.	Update the POST and PUT routes in routes/user.js to validate user input:

javascript

Copy code

```
// routes/user.js


// POST /users - Create a new user
router.post('/users', (req, res) => {
  const { name, email, bio } = req.body;
  if (!name || !email || !bio) {
    return res.status(400).send('Name, email, and bio are required');
  }
  const users = readUsers();
  const newUser = { id: users.length + 1, name, email, bio };
  users.push(newUser);
  writeUsers(users);
  res.status(201).json(newUser);
});


// PUT /users/:id - Update a user by ID
router.put('/users/:id', (req, res) => {
  const { name, email, bio } = req.body;
  if (!name || !email || !bio) {
    return res.status(400).send('Name, email, and bio are required');
  }
  const users = readUsers();
  const userIndex = users.findIndex(u => u.id === parseInt(req.params.id));
  if (userIndex === -1) {
    return res.status(404).send('User not found');
  }
  const updatedUser = { ...users[userIndex], name, email, bio };
  users[userIndex] = updatedUser;
  writeUsers(users);
  res.json(updatedUser);
});
```

6. Integrate Routes with the Express Server

15. Open app.js and integrate the user routes:

javascript

Copy code

```
// app.js
const express = require('express');
const app = express();
const port = 3000;
const userRoutes = require('./routes/user');

// Middleware to parse JSON bodies
app.use(express.json());

// Root route
app.get('/', (req, res) => {
  res.send('Welcome to the User Management System');
});

// User routes
app.use('/api', userRoutes);
```

```
// Start the server

app.listen(port, () => {

  console.log(Server is running on http://localhost:${port});

});
```