SIT774 Task9.2

This project focuses on the development of a local server that connects to a MySQL database using Node.js, with TypeScript as the underlying layer for code quality and type safety. The main goal was to build a functional backend system capable of handling database post operation, dynamic HTML rendering, and server-side form validation.

2. Project Implementation

To achieve the project objectives, the following technologies were employed:

- **Node.js**: Node.js was chosen as the runtime environment for server-side development due to its efficiency in handling asynchronous operations and its extensive ecosystem of packages.
- **TypeScript**: TypeScript was used as a layer over JavaScript to ensure type safety and code consistency. This allowed for better debugging and readability during development. The TypeScript code was compiled into JavaScript, which is then served to the client.
- **Express.js**: Express.js provided a fast and flexible framework for building the web server and handling HTTP requests.
- **MySQL**: MySQL was used as the relational database management system to store and manage the data. The connection between Node.js and MySQL was established using the MySQL driver for Node.js.
- **EJS (Embedded JavaScript Templates)**: EJS was used to create dynamic HTML by injecting server-side data directly into the HTML templates. This allowed for seamless integration of data into the frontend.
- **Express Validator**: To ensure data integrity and security, Express Validator was employed to perform server-side validation on form inputs before any data was inserted into the MySQL database. This reduced the risk of SQL injection and other potential threats.

3. Server-Side Validation

One of the critical aspects of the project was the implementation of server-side validation. By using **Express Validator**, all user input fields were validated to ensure correctness and compliance with predefined rules before being processed by the server.

4. Dynamic HTML Rendering

The use of **EJS** allowed for easy integration of data into HTML templates, enabling dynamic content generation. This was particularly useful in displaying data fetched from the MySQL database to users in real-time. Through EJS, components could be updated efficiently without requiring extensive front-end rework.

5. Key Learnings

Throughout the course of this project, several key learnings emerged:

- **Integration of TypeScript and Node.js**: The use of TypeScript improved the quality of the JavaScript code, providing type safety and making debugging easier. This enhanced the overall robustness of the project.
- **Server-Side Validation**: Implementing server-side validation was crucial for preventing malicious data from entering the database. It emphasized the importance of security practices in web development.
- **Dynamic Content Management**: Working with EJS made it clear how useful server-side templating engines are for rendering dynamic content based on backend data. This improved the website's responsiveness and adaptability.
- **Database Management**: The integration of MySQL with Node.js showcased the power of relational databases in managing large sets of structured data. This experience emphasized the importance of database optimization and query handling for better performance.

App.ts

```
import express, {Express, Request, Response} from "express";
import { initializeCache, getCachedHosts } from "./cache";
import { body, validationResult } from 'express-validator';
import { insertProperty } from './database';

const path = require('path');
const app : Express = express();
const PORT = process.env.PORT || 3000;


initializeCache().then(() => {
  console.log('Cache initialized');
}).catch((error) => {
  console.error('Error initializing cache:', error);
});

app.use((req: Request, res: Response, next: Function) => {
  res.locals.hosts = getCachedHosts();
  next();
});
```

```
app.use(express.static(path.join(__dirname, '../public')));
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

app.set('view engine', 'ejs');
app.set('views', './views');

app.get('/', (req: Request, res: Response) => {

  res.render('index', {title: 'Home', hosts: res.locals.hosts});
});

app.post('/response', [
  body('host_id').notEmpty().withMessage('Host ID is required'),
  body('address')
    .matches(/^[0-9]+\s+[a-zA-Z\s]+$/).withMessage('Invalid address format')
    .isLength({ max: 50 }).withMessage('Address should be max 50 characters'),
  body('city')
    .matches(/^[a-zA-Z\s]+$/).withMessage('City should only contain letters and spaces')
    .isLength({ max: 50 }).withMessage('city should be max 50 characters'),
  body('state').isIn(['NSW', 'VIC', 'QLD', 'WA', 'SA', 'TAS', 'ACT', 'NT']).withMessage('Invalid state'),
  body('zip_code').matches(/^\d{4}$/).withMessage('Invalid postcode'),
  body('property_type').notEmpty().isLength({ max: 20 }).withMessage('Property type is required and should be max 20 characters'),
  body('description').isLength({ max: 100 }).withMessage('Description should be max 100 characters'),
  body('max_guests').notEmpty().isInt({ min: 1 }).withMessage('Max guests is required and should be at least 1'),
  body('price_per_night').notEmpty().isFloat({ min: 0 }).withMessage('Price per night is required and should be a positive number'),
], async (req: Request, res: Response) => {
```

```
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(400).json({ errors: errors.array() });
  }
  const formData = req.body;
  try {
    const propertyId = await insertProperty({
      host_user_id: parseInt(formData.host_id),
      address: formData.address,
      city: formData.city,
      state: formData.state,
      zip_code: formData.zip_code,
      property_type: formData.property_type,
      description: formData.description,
      max_guests: parseInt(formData.max_guests),
      price_per_night: parseFloat(formData.price_per_night)
    });
    console.log(`New property inserted with ID: ${propertyId}`);
    res.render('response', { title: 'Response', formData: formData, propertyId: propertyId });
  } catch (error) {
    console.error('Error inserting property:', error);
    res.status(500).json({ error: 'An error occurred while creating the property' });
  }
});

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

Database.ts

```
import express, {Express, Request, Response} from "express";
```

```
import { initializeCache, getCachedHosts } from "./cache";
import { body, validationResult } from 'express-validator';
import { insertProperty } from './database';

const path = require('path');
const app : Express = express();
const PORT = process.env.PORT || 3000;


initializeCache().then(() => {
  console.log('Cache initialized');
}).catch((error) => {
  console.error('Error initializing cache:', error);
});

app.use((req: Request, res: Response, next: Function) => {
  res.locals.hosts = getCachedHosts();
  next();
});

app.use(express.static(path.join(__dirname, '../public')));
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

app.set('view engine', 'ejs');
app.set('views', './views');

app.get('/', (req: Request, res: Response) => {

  res.render('index', {title: 'Home', hosts: res.locals.hosts});
});
```

```
app.post('/response', [
  body('host_id').notEmpty().withMessage('Host ID is required'),
  body('address')
    .matches(/^[0-9]+\s+[a-zA-Z\s]+$/).withMessage('Invalid address format')
    .isLength({ max: 50 }).withMessage('Address should be max 50 characters'),
  body('city')
    .matches(/^[a-zA-Z\s]+$/).withMessage('City should only contain letters and spaces')
    .isLength({ max: 50 }).withMessage('city should be max 50 characters'),
  body('state').isIn(['NSW', 'VIC', 'QLD', 'WA', 'SA', 'TAS', 'ACT', 'NT']).withMessage('Invalid state'),
  body('zip_code').matches(/^\d{4}$/).withMessage('Invalid postcode'),
  body('property_type').notEmpty().isLength({ max: 20 }).withMessage('Property type is required and should be max 20 characters'),
  body('description').isLength({ max: 100 }).withMessage('Description should be max 100 characters'),
  body('max_guests').notEmpty().isInt({ min: 1 }).withMessage('Max guests is required and should be at least 1'),
  body('price_per_night').notEmpty().isFloat({ min: 0 }).withMessage('Price per night is required and should be a positive number'),
], async (req: Request, res: Response) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(400).json({ errors: errors.array() });
  }
  const formData = req.body;
  try {
    const propertyId = await insertProperty({
      host_user_id: parseInt(formData.host_id),
      address: formData.address,
      city: formData.city,
      state: formData.state,
      zip_code: formData.zip_code,
```

```
        property_type: formData.property_type,
        description: formData.description,
        max_guests: parseInt(formData.max_guests),
        price_per_night: parseFloat(formData.price_per_night)
      });
      console.log(`New property inserted with ID: ${propertyId}`);
      res.render('response', { title: 'Response', formData: formData, propertyId: propertyId });
    } catch (error) {
      console.error('Error inserting property:', error);
      res.status(500).json({ error: 'An error occurred while creating the property' });
    }
});

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

Types.ts

```
export interface HostFromDB {
  user_id: number;
  host_rating: number;
};

export interface PropertyToInsert {
  host_user_id: number;
  address: string;
  city: string;
  state: string;
  zip_code: string;
  property_type: string;
```

```
  description: string;
  max_guests: number;
  price_per_night: number;
}
```

Index.ejs

```
<%- include('header'); -%>
<main>
  <div class="container mt-5">
    <h3 class="mb-4">Create a New Property</h3>

    <form action="/response" method="post" id="propertyForm">
      <div class="mb-3">
        <label for="host_id" class="form-label">Select Host:</label>
        <select id="host_id" name="host_id" class="form-select" required>
          <option value="">Choose a host</option>
          <% hosts.forEach(function(host) { %>
            <option value="<%= host.user_id %>">Host ID: <%= host.user_id %>, Rating: <%= host.host_rating %></option>
          <% }) %>
        </select>
      </div>

      <div class="mb-3">
        <label for="address" class="form-label">Address:</label>
        <input type="text" id="address" name="address" class="form-control" required maxlength="50" pattern="^[0-9]+\s+[a-zA-Z\s]+$">
        <small class="form-text text-muted">Format: Number followed by street name (e.g., 123 Main Street)</small>
      </div>
      <div class="mb-3">
        <label for="city" class="form-label">City:</label>
```

```
            <input type="text" id="city" name="city" class="form-control" required maxlength="50" pattern="^[a-zA-Z\s]+$">
        </div>
        <div class="mb-3">
            <label for="state" class="form-label">State:</label>
            <select id="state" name="state" class="form-select" required>
                <option value="">Select a state</option>
                <option value="NSW">New South Wales</option>
                <option value="VIC">Victoria</option>
                <option value="QLD">Queensland</option>
                <option value="WA">Western Australia</option>
                <option value="SA">South Australia</option>
                <option value="TAS">Tasmania</option>
                <option value="ACT">Australian Capital Territory</option>
                <option value="NT">Northern Territory</option>
            </select>
        </div>
        <div class="mb-3">
            <label for="zip_code" class="form-label">Postcode:</label>
            <input type="text" id="zip_code" name="zip_code" class="form-control" required pattern="^\d{4}$">
            <small class="form-text text-muted">4-digit Australian postcode</small>
        </div>
        <div class="mb-3">
            <label for="property_type" class="form-label">Property Type:</label>
            <input type="text" id="property_type" name="property_type" class="form-control" required maxlength="20">
        </div>
        <div class="mb-3">
            <label for="description" class="form-label">Description:</label>
            <textarea id="description" name="description" class="form-control" maxlength="100" rows="3"></textarea>
```

```
        <small class="form-text text-muted"><span id="charCount">0</span>/100 characters</small>
      </div>
      <div class="mb-3">
        <label for="max_guests" class="form-label">Max Guests:</label>
        <input type="number" id="max_guests" name="max_guests" class="form-control" required min="1">
      </div>
      <div class="mb-3">
        <label for="price_per_night" class="form-label">Price per Night:</label>
        <input type="number" id="price_per_night" name="price_per_night" class="form-control" required min="0" step="0.01">
      </div>
      <button type="submit" class="btn btn-primary">Create Property</button>
    </form>
  </div>
</main>
<%- include('footer'); -%>
```

Response.ejs

```
<%- include('header'); -%>
<main class="container">
  <h1>Property Created</h1>
  <p>Host ID: <%= formData.host_id %></p>
  <p>Address: <%= formData.address %></p>
  <p>City: <%= formData.city %></p>
  <p>State: <%= formData.state %></p>
  <p>Country: Australia</p>
  <p>Postcode: <%= formData.zip_code %></p>
  <p>Property Type: <%= formData.property_type %></p>
  <p>Description: <%= formData.description %></p>
  <p>Max Guests: <%= formData.max_guests %></p>
```

```
</main>

<%- include('footer'); -%>
```