Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Professional Practice in IT

Website to find out Trades men in your area.

Student: Damien McDonagh

Student number: g00386999

Supervisor: Brian McGinley

Table of Contents

Github

https://github.com/damienmcdonagh2020/Professional-Practise

Introduction

This documentation covers the Professional Practice in IT solo repeat project, undertaken by Damien McDonagh, under the supervision of Brian McGinley. The project involves enhancing and fixing issues in a website designed to help users find tradesmen in Ireland, the project built in semester two as part of the group. The primary focus is on getting the website fully functional and improving its user interface. This document addresses various aspects of the project, including requirements, technologies used, design methodologies, architecture, limitations, test plans, conclusions, and recommendations.

Project Requirements

The project aims were to achieve the following requirements:

- **Improved User Interface**: Enhancing the website's user interface to make it more user-friendly and presentable, for users on the site.
- **Database Integration**: Implement a database system to store and manage data related to tradesmen and their locations, name, and contact details. Database system to store users' email, user name and password
- **Search Functionality**: Develop features that allow users to search for specific trades and locations effectively.

Architecture

1. MongoDB

- MongoDB is our NoSQL database management system. MongoDB stores and manages our application's data. It's extremely scalable and schema-less, therefor making it suitable for accommodating a wide range of data structures which we use in this project.

2. Express.js

- Express.js serves as our back-end framework. Express.js is responsible for handling HTTP requests, routing, and serving as the API layer. It communicates with the database, performs data validation, and responds to client requests. Express.js follows RESTful principles, providing a structured and organized approach to handling CRUD operations.

3. React

- React is our front-end library React handles the user interface and client-side rendering. It creates dynamic, interactive, and responsive user interfaces that enhance the user experience. React components are reusable, making it efficient for building complex UIs.

4. Node.js

- Node.js is our server-side runtime environment. Node.js powers our server, facilitating real-time communication and data flow between the client and server. It uses an event-driven, non-blocking I/O model, which is ideal for handling asynchronous operations and scaling applications.

Architecture Flow

- User Interaction: Users interact with the React-based front-end, initiating requests.
- Front-end Components: React components process user input, display data, and trigger HTTP requests to the Express.js back end.
- Express.js Routing: Express.js routes incoming requests, handling CRUD operations. It communicates with the MongoDB database to retrieve or modify data.
- MongoDB Database: MongoDB stores and manages data in a flexible, JSON-like format.
- Data Response: Express.js responds to the front end with data retrieved from the database or confirmation of CRUD operations.
- User Interface Updates: React updates the user interface based on the responses received from the back end, providing a real-time, dynamic user experience.

Benefits of this Architecture

- Scalability: MongoDB and Node.js enable horizontal scaling, allowing the application to handle increased traffic and data as needed which is great for my project.
- Responsiveness: Reacts component-based structure ensures responsive and interactive user interfaces.
- Efficiency: Express.js simplifies routing and API development, streamlining the back-end logic.
- Real-time Updates: Node.js communicate in real-time between clients and the server.

Design Methodologies Implemented

The project follows a modular and component-based design approach:

- **Modular Design**: Components and modules are used to create a structured and organized application.
- **Component Reusability**: React components are employed to ensure the reusability of UI elements.
- **RESTful Principles**: Express.js adheres to RESTful principles, allowing for systematic handling of CRUD operations.

Limitations

The project scope does not cover advanced user authentication and authorization mechanisms. Security measures such as password hashing are not implemented in this version.

Limited error handling and validation are in place. Login and authentication have not been completed.

Test Plans

Tested the project on Postman, the browsers inspector, and the terminal for problems. Problems encountered were searched online for how to guide on where to fix these issues.

Also created users and used the website like a user.

Conclusions and Recommendations

Conclusions

In conclusion, the MERN CRUD application combines the strengths of MongoDB, Express.js, React, and Node.js to deliver a scalable, flexible, and responsive web application. Showing how the foundation provided can build dynamic and efficient applications for handing extremely complex data operations while not overly confusing and great for user experience.

Recommendations

I wish to implement more advanced user authentication and authorization for log in security. Expanding the database schema to include more information and make a review portal. Adding a calculator for users so people can get a better price on jobs. Iv learned a lot from this experience, aside from the code to be more patient with work and to present it in a more professional manner. That is my biggest take home from the project.