

Modules



Admin



Customer



Shopping



Vandor



Delivery



NoSQL
Indexing
Caching



Admin



Vandor



Customer



Shopping



Delivery



Firestore

Firestore



Location Service

Social Login



Admin



Vandor



Customer



Shopping



Delivery



OTP Model



Admin



Vandor



Customer



Shopping



Delivery



Send Notification
Send Email



Admin



Vandor



Customer



Shopping



Delivery

API Gateway



Message Queue



Admin



Vandor



Customer



Shopping



Delivery





Container Orchestratio



kubernetes

Routing Endpoints





Deploy on Cloud





node

mongoDB. redis

PM2

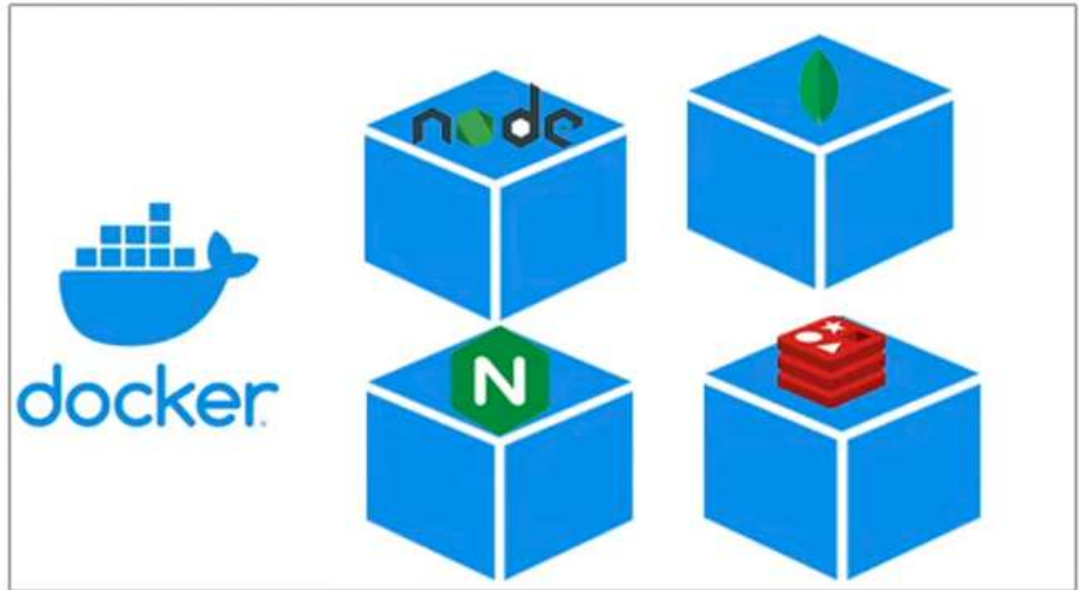


PORT 8000

NGINX



Google Cloud










kubernetes










Summary

In this video, we expand our Node.js food order app by enhancing the admin controller functionality for managing restaurant vendors, including creating, retrieving, and validating vendors.

Highlights

-  Enhanced admin controller functionality for vendor management.
-  Implemented password encryption using bcrypt for security.
-  Successfully retrieved and displayed all vendors in the database.
-  Cleared duplicate vendor entries from the database for accurate results.
-  Added vendor retrieval by ID with error handling for better user experience.
-  Implemented login functionality for vendors with validation checks.
-  Introduced JWT for secure authentication and authorization mechanisms.

Key Insights

-  **Code Reusability:** The refactoring of vendor retrieval functions highlights the importance of reusing code to reduce redundancy, making the application more maintainable.
-  **Security Focus:** Integrating bcrypt for password encryption emphasizes the need for security in applications handling sensitive user data, ensuring safer logins.
-  **Database Management:** Using tools like Studio 3T for MongoDB management streamlines the process of database manipulation, enhancing developer efficiency.
-  **Error Handling:** Adding a structured error handling system improves application robustness and user experience by providing meaningful feedback.
-  **Authentication:** Implementing JWT for managing user sessions introduces a scalable authentication mechanism, crucial for applications with multiple user roles.
-  **Vendor Profile Management:** The ability to update vendor profiles dynamically facilitates better management of vendor services, enhancing user interactions.
-  **Future Scalability:** The groundwork laid for vendor management systems and authentication opens pathways for future features like food listings and profile picture

Application Pipeline and Vendor Functionality

Episode Summary

- 👋 Welcome to the channel and recap of previous episode
- 📖 Discussion on vendor creation and password encryption

Admin Controller Enhancements

- 📄 Adding functionality to list all vendors
- 🔍 Finding vendors by ID

Database Management

- 🗄 Managing vendor data through MongoDB
- 🧹 Deleting duplicate vendor entries

Vendor Routes Implementation

- 🌐 Setting up routes for getting vendors
- 🛡 Adding error handling for vendor retrieval

Vendor Functionality Development

- 🔑 Implementing vendor login functionality
- ✅ Validating user credentials with bcrypt

Signature Generation

- 🔒 Creating a signature for authenticated sessions
- 📄 Using the signature for subsequent requests

Middleware Integration

- 🛡 Adding middleware for user authentication
- 📖 Validating requests using the signature

Update and Edit Vendor Profiles

- ✍ Implementing profile update functionality
- ✅ Ensuring fields are editable based on vendor's choice

Next Steps

- 🗂 Planning to add food options and profile picture uploads
- 👋 Thanking viewers and previewing the next episode

Summary

In this episode, we expand the online order app by adding food functionality and integrating image uploads for vendors and food items.

Highlights

- 🚀 Added food functionality to the vendor profile.
- 🍲 Created a food model to associate food items with specific vendors.
- 📁 Integrated image upload functionality using Multer.
- 🔧 Updated the vendor controller for food item management.
- 🔄 Successfully tested food creation via Postman.
- 🖼️ Implemented cover image upload for vendor profiles.
- ✅ Ensured seamless data handling between MongoDB and the application.

Key Insights

- 🏠 **Dynamic Food Management:** The integration of food functionality allows vendors to manage their offerings effectively, enhancing user experience. This modular approach benefits scalability as more features can be added later.
- 🔗 **Vendor-Food Association:** Linking food items to specific vendors aids in organizing the menu, making it easier for customers to find what they want, thereby improving order accuracy and satisfaction.

Episode Summary: Vendor and Food Functionality

Vendor Functionality Recap

- ❓❓ Previous episode covered vendor functionality
- 🔄 Continuing with banner and food functionality in this episode

Adding Food Functionality

- 🏗️ Create a model for food similar to vendor
- 🔧 Implement food model in vendor controller

Food Creation Process

- 🔍 Find vendor before adding food
- 📁 Create food and link to vendor's food array

Testing Food Creation

- 🖋️ Use Postman to test food addition
- ✅ Add multiple food items successfully

Image Upload Functionality

- 🖼️ Implement image upload for food items
- 📁 Use Multer library for file handling

Updating Vendor Profile with Images

- 👤 Add functionality to upload vendor profile images
- 🔄 Ensure correct data handling for images

Next Steps

- 🛒 Upcoming episode to work on shopping section
- 🚀 Focus on user module for food and cart access

Summary

In this episode, we refactor the code and implement the shopping section, enabling users to browse food catalogs from vendors based on their location.

Highlights

- 🔍 Refactored code for better organization and maintainability.
- 📦 Created separate service files for Express and MongoDB connections.
- 🗺️ Implemented routes for the shopping section to access food catalogs.
- 🇮🇳 Developed a controller to handle food availability queries based on pin codes.
- ⌚ Added functionality to filter foods available within 30 minutes.
- 🍽️ Retrieved top restaurants based on user location and availability.
- ✅ Successfully tested API calls using Postman for functionality validation.

Key Insights

- 🔄 Refactoring is crucial for maintaining a clean codebase, improving the project's scalability. By organizing code into services, we enhance readability and make future updates easier.
- 📍 Location-based services play a significant role in food delivery apps, allowing users to find nearby options quickly, thereby improving user experience.
- ⚙️ Implementing controllers for different sections of the application helps manage the complexity, making it easier to handle various functionalities like food availability and restaurant listings.
- ⌚ The feature to filter food based on preparation time enhances convenience for users,

Episode 4 Overview

Introduction

- 👋 Welcome to the channel
- 📺 Continuing with series, focusing on the shopping section

Project Structure

- 📁 Refactoring code to separate services
- 📁 Creating a services folder for organization

Shopping Section Development

- 🛒 Adding routes for the shopping section
- 🔧 Creating a controller to handle shopping routes

Food Availability

- 📍 Using pin codes to find food availability
- 🔍 Checking service status of vendors

Top Restaurants

- ⭐ Limiting results to top-rated restaurants
- 🇮🇳 Ensuring correct pin code input for accurate results

Food Preparation Time

- 🕒 Filtering foods based on preparation time
- 🍲 Returning foods ready in less than 30 minutes

Source Food Functionality

- 🌐 Fetching available foods by pin code
- 🔄 Implementing caching techniques for efficiency

Final Steps

- ✅ Testing all functionalities through Postman
- 📦 Preparing for the next episode on user module development

Conclusion

- 🎉 Summary of features covered in the shopping module
- 📅 Preview of upcoming user account and OTP system functionality

Summary

In this episode, we dive into the customer module, focusing on account creation, OTP verification, and refining validations to enhance the food order app's backend.

Highlights

- 🚀 Customer Module Focus: We're building the customer onboarding process.
- 🛒 Account Creation: Implementing customer account creation with validations.
- 🔑 OTP Verification: Adding OTP mechanisms for secure account verification.
- 📝 Code Refactoring: Improving code structure for better validation handling.
- 🛡️ Authentication: Ensuring secure access to routes with authentication.
- 📦 Profile Management: Enabling customer profile retrieval and updates.
- ✅ Future Enhancements: Next episode will cover cart and order functionalities.

Key Insights

- 📈 Customer Onboarding: A seamless onboarding process is crucial for user retention and satisfaction, enhancing the overall user experience.
- 🛡️ Security Measures: Implementing OTP verification adds a layer of security, preventing unauthorized access to user accounts.
- 🛠️ Code Structure: Refactoring code not only improves readability but also makes it easier to maintain and scale the application in the future.
- 📊 User Data Management: Efficient profile management helps in personalizing user experiences and streamlining interactions.

Mind Map Structure

Introduction

- 👋 Welcome to the channel
- 📺 Series continuation on shopping module

Previous Episode Recap

- 🛒 Covered basic functionality of the shopping module
- 📍 Foods available as per location
- 🛒 Prepared to add to cart

Current Focus: Customer Module

- 👤 Working on the customer module
- ⌚ Good amount of time needed for this module

Structure Overview

- 🏠 Monolithic approach for basic understanding
- 🌀 Transition to microservices for professional implementation

Customer Account Creation

- ✍️ Create customer account functionality
- 🔒 OTP verification for account confirmation
- 🔑 Logging into customer profile

Code Refactoring

- 🔄 Basic validation implementation
- 🔧 Shopping controller adjustments

Routes Setup

- 🚦 Adding customer/user routes
- 🔒 Authentication required for specific routes

Customer Controller

- 👤 Naming conventions for the customer controller
- 🖥️ Sign up function implementation
- 📄 Input validation using DTO (Data Transfer Object)

Mind Map Structure

Customer Controller

- 👤 Naming conventions for the customer controller
- 🖨️ Sign up function implementation
- 📄 Input validation using DTO (Data Transfer Object)

Libraries Installation

- 📦 Added necessary libraries for validation
- 🔧 Utilizing class-transformer for input handling

OTP Generation

- 🔑 Unique OTP generation for account verification
- 📠 Sending OTP via Twilio

Customer Verification

- 🔍 Verification of customer OTP
- 📅 Checking OTP expiry and validity

Customer Login

- 🔑 Implementing customer login functionality
- 📠 Error handling for login issues

Customer Profile Management

- ⚙️ Requesting new OTP functionality
- ✎️ Editing customer profile details
- 📄 Fetching customer profile information

Upcoming Features

- 🛒 Add to cart functionality
- 📦 Order section implementation








Conclusion

- 👍 Encouragement to like and subscribe
- 📅 Teaser for the next episode








Summary

In this tutorial, we deploy a Node.js Food Order System on Heroku and work on the order section, preparing for future microservices architecture.

Highlights

-  **Deployment on Heroku:** Learn how to deploy a Node.js project on Heroku to make it accessible online.
-  **TypeScript Configuration:** Understand the importance of compiling TypeScript to ES5/ES6 for production.
-  **Dockerization:** Get introduced to Docker and how to create containers for your application and its dependencies.
-  **CI/CD Integration:** Overview of Continuous Integration and Deployment practices for efficient code updates.
-  **Order Functionality:** Implement create, get, and get by ID functionalities for managing food orders.
-  **Database Configuration:** Set up a MongoDB connection for storing order data securely.
-  **Next Steps:** Prepare for upcoming episodes focusing on microservices architecture.

Key Insights

-  **Heroku Deployment Process:** Deploying on Heroku simplifies the process of making applications accessible, though it's crucial to understand production configurations beforehand.
-  **TypeScript vs. JavaScript:** Compiling TypeScript into JavaScript is vital for performance and compatibility in production environments, ensuring smoother execution.
-  **Benefits of Docker:** Docker allows for consistent environments across development and production, making it easier to manage dependencies and application setup.
-  **DevOps Culture:** Understanding DevOps practices enhances collaboration between development and operations teams, leading to smoother deployments and updates.
-  **Order Management Logic:** Creating a robust order management system involves handling user requests, calculating totals, and ensuring data integrity.
-  **Secure Database Connections:** Using a cloud database like MongoDB ensures that your application can scale while maintaining security and performance.
-  **Microservices Transition:** Planning for a shift to microservices architecture can improve scalability and maintainability of applications in the long run.

Node Project Deployment and Order Management

Deployment Overview

- 🚀 Deploying current progress on Heroku
- 🔧 Transitioning to microservices architecture in future episodes

DevOps Workflow

- 🔑 Understanding production environment setup
- 🌐 Importance of Docker and CI/CD in deployment

Project Structure and Configuration

- 📁 Organizing project files into source directory
- 📦 Modifying package.json for Heroku compatibility

Database Configuration

- 🔗 Setting up MongoDB cloud database
- 🇩🇪 Managing environment variables for port configuration

Order Management Implementation

- 📝 Creating order routes and controller methods
- 🔍 Handling order creation and retrieval logic

Testing and Validation

- ✅ Successful deployment and testing of application on Heroku
- 🇩🇪 Validating order creation and retrieval functionalities

Future Directions

- 🚀 Preparing for microservices architecture in upcoming episodes
- 🌟 Exploring advanced topics and deployment strategies in DevOps

Summary

The video discusses building a powerful cart system for a food order app using Node.js, focusing on customer and vendor functionalities.

Highlights

- 🛠️ **Project Continuation:** The host returns after a health break to finish the food order app series.
- 🛒 **Cart Features:** Implementing cart functionalities for customers and vendors to enhance user experience.
- 🔄 **Module-wise Progress:** Completing features in modular steps to align backend and mobile app development.
- 📧 **Community Engagement:** Responding to viewer emails about project completion and addressing concerns.
- 🔧 **Technical Implementation:** Detailed coding demonstration for adding, retrieving, and deleting cart items.
- ✅ **Order Processing:** Integration of order processing features for vendors to manage customer orders effectively.
- 🚀 **Future Plans:** Upcoming episodes will cover vendor offers and transaction management.

Key Insights

- 🔄 **Project Synchronization:** Aligning the backend and mobile app development ensures consistency and functionality across platforms, benefiting developers and users alike.
- 📧 **Viewer Interaction:** Engaging with the audience through emails fosters community and helps tailor content to viewer needs, enhancing satisfaction and retention.
- 🛒 **Cart Necessity:** Implementing a cart system is crucial for user convenience, allowing access to the same items across multiple devices, enhancing the overall user experience.
- 🔧 **Modular Development:** Breaking down the project into manageable modules simplifies the development process, making it easier to troubleshoot and implement features systematically.
- 🛠️ **Error Handling Focus:** Emphasizing error handling in code development improves reliability and user trust, ensuring smoother application performance.
- 📊 **Vendor Features:** Incorporating vendor functionalities like order processing is essential for facilitating business operations, promoting efficiency in order management.
- 📅 **Future Enhancements:** Planning for additional features like offers and transaction tracking indicates a commitment to continuous improvement and user satisfaction.