Invoice Generator API - Project Documentation

Project Overview

The Invoice Generator API is a backend system built with Node.js that enables invoice management with full CRUD functionality. It uses JWT-based authentication and role-based access control (RBAC) to differentiate between Admin and User roles. The Admin role has full access to CRUD operations for managing invoices, while the User role can only view invoices. The system includes essential features such as invoice number generation, payment tracking, tax calculations, and discounts.

Additionally, the project implements an event-driven architecture with event listeners and emitters to handle invoice events like creation, updates, and deletion. This architecture allows for asynchronous processing by using queues of these events, which can be useful for integrations such as sending emails to customer.

Core Features

1. Invoice CRUD Operations:
    - Create: Admin can generate invoices with customer details, items, taxes, total amounts, and payment status.
    - Read: Both Admins and Users can retrieve invoices. Admins can access all invoices, while users can only view invoices they have access to.
    - Update: Admins can update invoice details such as items and payment status.
    - Delete: Admins can delete invoices from the system.
2. Role-Based Access Control (RBAC):

- Admin users have full access to create, update, delete, and view invoices.
- Regular users can only view invoices (read-only).

3. Authentication:
    - Implemented JWT-based authentication to secure API endpoints.
    - Users must log in with credentials, and JWT tokens are issued for authorized access.
4. Invoice Features:
    - Invoice Number Generation: Automatically generates unique invoice numbers.
    - Payment Tracking: Tracks the payment status of each invoice (pending, paid, overdue).
    - Tax Calculations: Calculates taxes (e.g., GST) and applies them to the invoice.
    - Discounts: Allows Admin to apply discounts to the invoice total.
5. Event-Driven Architecture:
    - Emitting events when invoices are created, updated, or deleted.
    - Use of event listeners and emitters to handle events like sending emails.
6. Other Enhancements:
    - PDF Generation: Functionality to generate downloadable PDF versions of invoices.
    - Event Queue: Integration with Redis for handling events asynchronously.

Technologies Used

- Node.js: Backend runtime environment for building the API.
- Express: Web framework for routing and handling HTTP requests.
- MongoDB: Database for storing invoice and user data.
- JWT: For secure authentication and authorization.
- Mongoose: MongoDB ODM (Object Data Modeling) for interacting with the database.
- Redis: For event queue handling
- Puppeteer: For generating PDF invoices.

Folder Structure

/videosdk_task

```
├─── /config      # Database and Mail Service Configuration
├─── /controllers         # Mongoose models for Invoice and User
        ├─── /auth       # Authentication Controllers
        ├─── /invoice   # Invoice Controllers
├─── /routes         # API routes
├─── /middlewares        # Business logic for invoice operations
├─── /utils          # Utility functions (e.g., Invoice number generation, validation)
├─── /events         # Event emitters and listeners
├─── /queues         # To manage events in Queue
└─── app.js          # Main entry point of the application
```

Created By : Dhruv Godhani

API Endpoints

For detailed API documentation, refer to the separate **API Documentation (api-documentation.pdf)**.

Database Models

1. User Model:
    - username: String (unique, required)
    - password: String (hashed)
    - email : String
    - role: String (either "admin" or "user")
2. Invoice Model:
    - invoiceNumber: String (auto-generated or custom)
    - customer: Object (contains name, address, email, phone)
    - invoiceDate: Date
    - dueDate: Date
    - items: Array (contains name, quantity, unitPrice)
    - taxRate: Object (GST)
    - totalAmount: Number (calculated)
    - paymentStatus: String (pending, paid, overdue)
    - createdBy: Admin User who creates invoices

Authentication & Authorization

- JWT Authentication:

    - Users need to login using their username and password.
    - Upon successful authentication, a JWT token is generated and returned.
    - The token is used for accessing protected routes.
- Role-Based Access Control:
    - Admin: Can create, read, update, and delete invoices.
    - User: Can only view invoices they have access to.

Event-Driven Architecture

- Event Listeners:
    - On invoice creation and update events are emitted.
    - Listeners can trigger actions like sending a invoice pdf to customer via mail, updating logs, or notifying users.
- Event Queue:
    - An event queue (e.g., Redis) can be used to handle events asynchronously for scalable operations.

PDF Generation

The system can generate a downloadable PDF version of an invoice, formatted with customer details, item list, taxes, and total amounts. This can be achieved using Puppeteer and HTML library.

Challenges Faced

- Event Handling: Ensuring that the event-driven approach worked seamlessly with the invoice generation and other CRUD operations.

- Manage Events in Queue: Managing Events in Queue and Process them by using BullMQ.
- PDF Generation: Creating a well-formatted PDF that accurately reflects the invoice data and is easy for users to download.

Conclusion

This project demonstrates a comprehensive invoice management system with full CRUD functionality, role-based access control, and event-driven architecture. The system also includes advanced features like tax calculations, invoice number generation, payment tracking, and PDF generation. The use of JWT for authentication ensures secure access, and the event-driven architecture enables scalable and flexible operations.

Created By : Dhruv Godhani