I'll create comprehensive documentation for your Marketing Content Generator project.

Marketing Content Generator

Overview

The Marketing Content Generator is a full-stack web application that leverages OpenAI's GPT-3.5 to generate marketing content based on user-specified parameters.

Technologies Used

  - Spring Boot

    1. Java
    2. MongoDB
    3. OpenAI API

- Frontend:

    1. React
    2. Axios

- Dependencies:

    1. Lombok
    2. OkHttp
    3. Jackson
    4. Spring Data MongoDB

Project Structure

Backend (Spring Boot)

src/main/java/org/generativeai/contentgenerator/
|
├── Controller/
| └── ContentController.java   # REST API endpoint handler
|
├── Model/
| └── ContentRequest.java   # Data model for content generation request
|
├── Repository/
| └── ContentRepository.java   # MongoDB repository interface

```
|
├── Service/
|     └── ContentService.java      # Business logic for content generation
|
└── ContentGeneratorApplication.java  # Main application starter
```

Frontend (React)

```
src/
|
├── components/
|     ├── ContentForm.jsx         # Form for capturing content generation parameters
|     ├── GeneratedContent.jsx     # Display generated content
|     └── Navbar.jsx              # Application navigation bar
|
├── controllers/
|     └── contentController.js      # API call handler
|
└── styles/
      └── components.css           # Styling for components
```

Key Features

- Generate marketing content with customizable parameters
- Select content tone (friendly, professional, funny)
- Choose content length (short, medium, long)
- Pick content format (text, article, email)
- Persist generated content in MongoDB
- Real-time content generation using OpenAI API

Configuration

Backend Configuration

1. Configure OpenAI API Key

  - Add to `application.properties`:

    ```properties

```properties
openai.api.key=your_openai_api_key
```

2. MongoDB Configuration

```properties
spring.data.mongodb.uri=mongodb://localhost:27017/contentgenerator
```

Frontend Configuration

1. API URL Configuration
- In `contentController.js`:
```javascript
const API_URL = 'http://localhost:8080/api/v1/content/generate';
```

Setup and Installation

Prerequisites
- Java 11+
- Node.js and npm
- MongoDB
- OpenAI API Key

Backend Setup
1. Clone the repository
2. Configure `application.properties`
3. Run `mvn clean install`
4. Start the application with `mvn spring-boot:run`

Frontend Setup
1. Navigate to frontend directory
2. Run `npm install`
3. Start development server with `npm start`

API Endpoints

Generate Content

- **URL:** `/api/v1/content/generate`

- **Method:** POST

- **Request Body:**

  ```json
  {
    "tone": "friendly",
    "targetAudience": "marketers",
    "format": "article",
    "length": "medium"
  }
  ```

- **Response:**

  ```json
  {
    "generatedContent": "Your generated marketing content..."
  }
  ```

Error Handling

- Frontend validates required fields

- Backend performs input validation

- Graceful error messages displayed to user

Future Improvements

- Add user authentication

- Implement content saving/history

- Expand tone and format options

- Enhance error handling

- Add more detailed prompt engineering

Troubleshooting

- Ensure OpenAI API key is valid

- Check MongoDB connection

- Verify CORS settings

- Confirm backend and frontend are running on correct ports

Security Considerations

- Never expose OpenAI API key publicly

- Use environment variables for sensitive information

- Implement proper input sanitization

Performance Optimization

- Implement caching for generated content

- Add rate limiting to API

- Optimize OpenAI API calls

Licensing

This project is open-source. Please check the LICENSE file for details.

Contributing

Contributions are welcome! Please read the CONTRIBUTING.md for guidelines.

Contact

For support or inquiries, contact

Rushikesh Satish Jadhav

Email: iam.rushikesh@outlook.com

Phone: +91 8788772381

LinkedIn: linkedin.com/in/the-rsj/