***Real Time Chat Application***

1. Introduction

The project aims to develop a chat application backend using Node.js, PostgreSQL for database management, Redis for session management and message caching, and Kafka for handling real-time message delivery. It provides RESTful API endpoints for user authentication, message sending, user management, and more. The backend is designed to be scalable, secure, and efficient, allowing users to communicate seamlessly in real-time while ensuring data integrity and privacy.

1. Installation

# Clone the repository

git clone https://github.com/d-404/Chat-Application.git

# Navigate to the project directory

cd chat-app

# Install dependencies

npm install

1. Usage

# Run the project

npm start

1. API Documentation

*Authentication Endpoints*

* Sign Up

URL: /signup

Method: POST

Description: Creates a new user account.

Request Body:

{

"email": "example@example.com",

"password": "password123"

}

Response:

Status: 201 Created

Body:

{

"user": {

"id": "1",

"email": "example@example.com"

}

}

* Login

URL: /login

Method: POST

Description: Logs in an existing user.

Request Body:

{

"email": "example@example.com",

"password": "password123"

}

Response:

Status: 200 OK

Body:

{

"message": "Login successful"

}

*User Management Endpoints*

* Get User By ID

URL: /users/:id

Method: GET

Description: Retrieves user information by ID.

Response:

Status: 200 OK

Body:

{

"user": {

"id": "1",

"email": "example@example.com"

}

}

* Update User Details

URL: /users/:id

Method: PUT

Description: Updates user details.

Request Body:

{

"email": "new@example.com",

"password": "newpassword123"

}

Response:

Status: 200 OK

Body:

{

"user": {

"id": "1",

"email": "new@example.com"

}

}

* Delete User Account

URL: /users/:id

Method: DELETE

Description: Deletes a user account.

Response:

Status: 200 OK

Body:

{

"message": "User deleted successfully"

}

*Message Management Endpoints*

* Send Message

URL: /messages

Method: POST

Description: Sends a message from one user to another.

Request Body:

{

"senderId": "1",

"receiverId": "2",

"messageContent": "Hello, how are you?"

}

Response:

Status: 201 Created

Body:

{

"message": "Message sent successfully"

}

1. Configuration

# Set environment variables

export PORT=3000