Modern Chat Application with AWS Lambda Integration

A full-stack chat application built with React, Django, and AWS Lambda functions. This project demonstrates a responsive web design, real-time chat functionality using WebSockets, and AWS Lambda integration.

Features

Frontend

- Responsive design with automatic scaling based on screen size
- Fixed navigation bar with authentication
- Collapsible left menu for user list
- Real-time chat interface
- Right panel showing active users and recent activity
- Beautiful UI with Tailwind CSS

Backend

- User authentication system
- WebSocket integration for real-time chat
- Message history storage and retrieval
- RESTful API endpoints
- Database integration with Django models

AWS Integration

- Lambda function for basic arithmetic operations
- File upload functionality with S3 integration

Live Demo

- Frontend: Vercel Deployment
- Backend: PythonAnywhere

Technology Stack

- **Frontend**
  - React
  - Tailwind CSS
  - WebSocket client
  - Vite
- **Backend**
  - Django
  - Django Channels

    - Django REST Framework
    - SQLite (Development)
    - PostgreSQL (Production)

- **AWS Services**

    - Lambda
    - S3
    - API Gateway

📋 Prerequisites

- Node.js (v14 or higher)
- Python (3.8 or higher)
- Git
- AWS Account (for Lambda functions)

🔧 Installation

Clone the Repository

```bash git clone https://github.com/shibbu04/python_chat_app.git cd chat-application ```

Frontend Setup

1. Navigate to the frontend directory: ```bash cd frontend ```
2. Install dependencies: ```bash npm install ```
3. Create a .env file: ```bash VITE_API_URL=your_backend_url VITE_WS_URL=your_websocket_url ```
4. Start the development server: ```bash npm run dev ```

Backend Setup

1. Navigate to the backend directory: ```bash cd backend ```
2. Create a virtual environment: ```bash python -m venv venv source venv/bin/activate # On Windows: venv\Scripts\activate ```
3. Install dependencies: ```bash pip install -r requirements.txt ```
4. Set up environment variables: ```bash cp .env.example .env

   # Edit .env with your configuration

   ```
5. Run migrations: ```bash python manage.py migrate ```
6. Start the development server: ```bash python manage.py runserver ```

# Deployment

## Frontend Deployment (Vercel)

1. Fork this repository
2. Connect your fork to Vercel
3. Configure environment variables
4. Deploy

## Backend Deployment (PythonAnywhere)

1. Create a PythonAnywhere account
2. Set up a new web app
3. Clone the repository
4. Configure virtual environment
5. Set up static files
6. Configure WSGI file
7. Update allowed hosts

# Project Structure

``` chat-application/ ├── frontend/ │ ├── src/ │ │ ├── components/ │ │ ├── App.jsx │ │ └── main.jsx │ ├── package.json │ └── vite.config.js ├── backend/ │ ├── chat/ │ ├── chatproject/ │ └── manage.py ├── aws/ │ ├── add-numbers/ │ └── file-upload/ └── README.md ```

# Security

- CORS configuration
- Token-based authentication
- WebSocket authentication
- Secure file uploads
- Environment variable protection

# Contributing

1. Fork the repository
2. Create a feature branch
3. Commit changes
4. Push to the branch
5. Open a pull request

Developed by Shivam ♥ GitHub : shibbu04 Portfolio : shivam04.tech LinekedIn : shivamsingh57680
```