



CHATT

Real-time & Offline-Ready
Messaging App

Built with React, GraphQL, AWS AppSync & AWS Amplify

Chatt

Real-Time Offline Ready Chat App written with GraphQL, AWS AppSync, & AWS Amplify

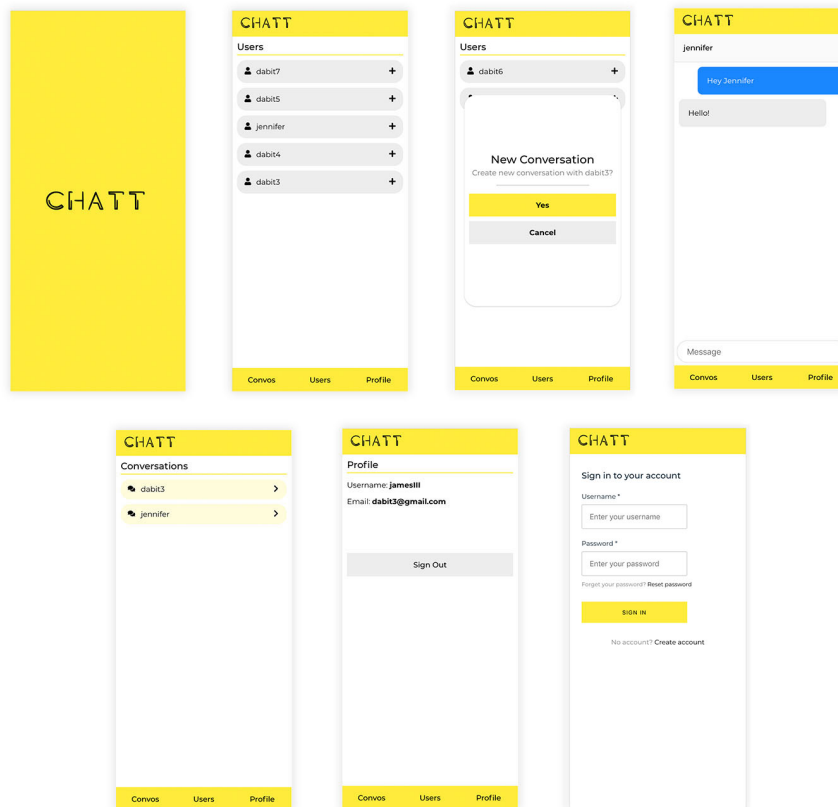
Features

- User management
- Routing (React Router)
- Real-time
- Offline ready (conflict resolution handled for you when user comes back online)

Technologies

- AWS AppSync
- AWS Amplify
- GraphQL
- MobX
- React Router
- React Apollo
- Glamor

Screens



Building the App (automated)

This project contains an Amplify project (`/amplify`) already configured & ready to be deployed. Deploying this project will create the following resources in your account: an authentication configuration using Amazon Cognito, an AWS AppSync GraphQL API, & a DynamoDB table.

1. Make sure you are on the newest version of the AWS Amplify CLI

```
1 npm install -g @aws-amplify/cli
```

You must also have the CLI configured with a user from your AWS account (`amplify configure`). For a walkthrough of how to do this, check out this video.

2. Clone Chatt

```
1 git clone https://github.com/aws-samples/aws-appsync-chat.git
```

3. Install dependencies

```
1 npm install
```

4. Initialize the amplify project

```
1 amplify init
```

- Enter a name for the environment **master**

5. Push the new resources to the cloud

```
1 amplify push
```

6. Run the project

```
1 npm start
```

7. Deleting the project resources

If you'd like to tear down the project & delete all of the resources created by this project, run the `delete` command.

```
1 amplify delete
```

Building the App (manually)

You can also manually set up your resources if you would like. If you would like to manually create & configure the resources for this project, follow these steps:

1. Install & configure the Amplify CLI

```
1 npm install -g @aws-amplify
2
3 amplify configure
```

2. Clone Chatt

```
1 git clone https://github.com/aws-samples/aws-appsync-chat.git
```

3. Install dependencies

```
1 npm install
```

-
4. Delete the amplify folder
 5. Initialize a new Amplify project

```
1 amplify init
```

6. Add authentication

```
1 amplify add auth
```

Here, either choose the default security choice or configure your own.

7. Add the api

```
1 amplify add api
```

Choose Cognito User Pools as the authentication type. When prompted for the GraphQL schema, use the following schema:

```
1 type User
2   @model
3   @auth(rules: [{ allow: owner, ownerField: "id", queries: null }]) {
4     id: ID!
5     username: String!
6     conversations: [ConvoLink] @connection(name: "UserLinks")
7     messages: [Message] @connection(name: "UserMessages")
8       createdAt: String
9       updatedAt: String
10  }
11
12 type Conversation
13   @model(
14     mutations: { create: "createConvo" }
15     queries: { get: "getConvo" }
16     subscriptions: null
17   )
18   @auth(rules: [{ allow: owner, ownerField: "members" }]) {
19     id: ID!
20     messages: [Message] @connection(name: "ConvoMsgs", sortField: "
21       createdAt")
22     associated: [ConvoLink] @connection(name: "AssociatedLinks")
23     name: String!
24     members: [String!]!
25       createdAt: String
26       updatedAt: String
27   }
28
29 type Message
30   @model(subscriptions: null, queries: null)
```

```

30  @auth(rules: [{ allow: owner, ownerField: "authorId" }]) {
31    id: ID!
32    author: User @connection(name: "UserMessages", keyField: "authorId")
33    authorId: String
34    content: String!
35    conversation: Conversation! @connection(name: "ConvoMsgs")
36    messageConversationId: ID!
37      createdAt: String
38      updatedAt: String
39  }
40
41  type ConvoLink
42    @model(
43      mutations: { create: "createConvoLink", update: "updateConvoLink" }
44      queries: null
45      subscriptions: null
46    ) {
47    id: ID!
48    user: User! @connection(name: "UserLinks")
49    convoLinkUserId: ID
50    conversation: Conversation! @connection(name: "AssociatedLinks")
51    convoLinkConversationId: ID!
52      createdAt: String
53      updatedAt: String
54  }
55
56  type Subscription {
57    onCreateConvoLink(convoLinkUserId: ID!): ConvoLink
58      @aws_subscribe(mutations: ["createConvoLink"])
59    onCreateMessage(messageConversationId: ID!): Message
60      @aws_subscribe(mutations: ["createMessage"])
61  }

```

8. Run the `push` command to create the resources in your account:

```
1 amplify push
```

9. Run the project

```
1 npm start
```

10. Deleting the project resources

If you'd like to tear down the project & delete all of the resources created by this project, run the `delete` command.

```
1 amplify delete
```

Hosting with the AWS Amplify Console

The AWS Amplify Console provides continuous deployment and hosting for modern web apps (single page apps and static site generators) with serverless backends. Continuous deployment allows developers to deploy updates to either the frontend or backend (Lambda functions, GraphQL resolvers) on every code commit to the Git repository.

1. Push your code to a Git repository of your choice.
2. Login to the AWS Amplify Console and choose **Connect app**
3. Connect your repository and branch.
4. Accept the default build settings.
5. Give the Amplify Console permission to deploy backend resources with your frontend. This will allow the Console to detect changes to your backend on every code commit. If you do not have a service role, follow the prompts to create one.
6. Review your changes and then choose **Save and deploy**. Your app will now be available at <https://master.unique-id.amplifyapp.com>.

About

Schema

This application utilizes 4 database tables:

- User table (stores user's identity information)
- Conversation table (stores conversations)
- Messages table (stores messages)
- ConvoLinkTable (stores relations between conversations & users)

License

This library is licensed under the Apache 2.0 License.