

## Course Project Part2

• Names \_\_\_\_\_ Xinyu Meng Zijie Li \_\_\_\_\_

• Date: \_\_\_\_\_ May 12th, 2024 \_\_\_\_\_

• Course Section: \_\_\_\_\_ 026 \_\_\_\_\_

Team Members: \_\_\_\_\_ Xinyu Meng, Zijie Li \_\_\_\_\_

Answers to Individual Questions: (100 points total, all questions weighted equally) o

Assumptions provided when required. Total in points (100 points total):

Professor's Comments:

Three cloud platforms used:

- AWS Amplify: Authentication and Cloud deployment
  - Botpress: AI chatbot platform
  - Azure Blob Storage: Data storage and file upload
- 
- AWS: Widely recognized for its comprehensive services, robust hosting capabilities, and strong enterprise support.
  - Azure: Known for its integration with Microsoft products and services, making it ideal for organizations reliant on Windows servers, SQL Server, and other Microsoft technologies.
  - Botpress: Botpress Cloud offers a streamlined and efficient way for developers to deploy and manage chatbots. It provides a one-click deployment method that simplifies the process, making it possible to have a chatbot running in the cloud within minutes.

1. First create our React App

```
npx create-react-app Cloud-computing
```

and install the Amplify package

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

```
src > App.js > App > Chatbot > useEffect() callback
1 import React, { useEffect, useState } from "react";
2 import logo from "./logo.svg";
3 import "./App.css";
4 import { Amplify } from "aws-amplify";
5 import awsconfig from "./aws-exports";
6 // import { AmplifySignOut, withAuthenticator } from '@aws-amplify/ui-react';
7 import { Authenticator, Button } from "@aws-amplify/ui-react";
8 import "@aws-amplify/ui-react/styles.css";
9 import axios from "axios";
10 import { BlobServiceClient, ContainerClient } from "@azure/storage-blob";
11
12 Amplify.configure(awsconfig);
13
14 function App() {
15   const chatbot = ({ showChatbot }) => {
16     useEffect(() => {
17       if (!showChatbot) {
18         return;
19       }
20
21       const script = document.createElement("script");
22       script.src = "https://cdn.botpress.cloud/webchat/v1/inject.js";
23       script.async = true;
24       document.body.appendChild(script);
25
26       script.onload = () => {
27         window.botpressWebChat.init({
28           botId: "cda40821-3a4d-4e01-a8ae-7d5ae88ca7a2",
29           hostUrl: "https://cdn.botpress.cloud/webchat/v1",
30           messagingUrl: "https://messaging.botpress.cloud",
31           clientId: "cda40821-3a4d-4e01-a8ae-7d5ae88ca7a2",
32         });
33       };
34
35       // 清理函数
36       return () => {
37         if (window.botpressWebChat) {
38           // Hide the chat
39           window.botpressWebChat.sendEvent({ type: "hide" });
40         }
41         document.body.removeChild(script);
42       };
43     }, [showChatbot]); // 依赖 showChatbot 状态变化
44
45     if (!showChatbot) {
46       return null;
47     }
48
49     return <div id="webchat" />;
50   };
51 }
```

## 2. Next initialize the amplify platform

*amplify configure*

*amplify init*

*amplify add auth*

*amplify push*

Now, the amplify authentication is successfully linked.

The screenshot shows the VS Code interface with the title bar "Cloud-computing". The Explorer sidebar on the left shows a project structure under "CLOUD-COMPUTING": amplify, node\_modules, public, and src. Inside src, there are files: amplifyconfiguration.json, App.css, App.js, App.test.js, aws-exports.js, Chatbot.js, index.css, index.js, logo.svg, reportWebVitals.js, setupTests.js, .gitignore, package-lock.json, package.json, and README.md. The "amplifyconfiguration.json" file is selected and open in the main editor area. The code content is as follows:

```
1 "aws_project_region": "us-east-1",
2 "aws_cognito_identity_pool_id": "us-east-1:c70b427f-7d06-441f-8684-02824eadb245",
3 "aws_cognito_region": "us-east-1",
4 "aws_user_pools_id": "us-east-1_IKZ1paMI6",
5 "aws_user_pools_web_client_id": "3qjgo317n8ib41frnoa8v2g3it",
6 "oauth": {},
7 "aws_cognito_username_attributes": [
8   "EMAIL"
9 ],
10 "aws_cognito_social_providers": [],
11 "aws_cognito_signup_attributes": [
12   "EMAIL"
13 ],
14 "aws_cognito_mfa_configuration": "OFF",
15 "aws_cognito_mfa_types": [
16   "SMS"
17 ],
18 "aws_cognito_password_protection_settings": {
19   "passwordPolicyMinLength": 8,
20   "passwordPolicyCharacters": []
21 },
22 "aws_cognito_verification_mechanisms": [
23   "EMAIL"
24 ]
25 }
```

The screenshot shows the VS Code interface with the title bar "Cloud-computing". The Explorer sidebar on the left shows a project structure under "CLOUD-COMPUTING": amplify, node\_modules, public, and src. Inside src, there are files: amplifyconfiguration.json, App.css, App.js, App.test.js, aws-exports.js, Chatbot.js, index.css, index.js, logo.svg, reportWebVitals.js, setupTests.js, .gitignore, package-lock.json, package.json, and README.md. The "aws-exports.js" file is selected and open in the main editor area. The code content is as follows:

```
1 /* eslint-disable */
2 // WARNING: DO NOT EDIT. This file is automatically generated by AWS Amplify. It will be overwritten.
3
4 const awsmobile = {
5   "aws_project_region": "us-east-1",
6   "aws_cognito_identity_pool_id": "us-east-1:c70b427f-7d06-441f-8684-02824eadb245",
7   "aws_cognito_region": "us-east-1",
8   "aws_user_pools_id": "us-east-1_IKZ1paMI6",
9   "aws_user_pools_web_client_id": "3qjgo317n8ib41frnoa8v2g3it",
10  "oauth": {},
11  "aws_cognito_username_attributes": [
12    "EMAIL"
13  ],
14  "aws_cognito_signup_attributes": string[],
15  "aws_cognito_signup_attributes": [
16    "EMAIL"
17  ],
18  "aws_cognito_mfa_configuration": "OFF",
19  "aws_cognito_mfa_types": [
20    "SMS"
21  ],
22  "aws_cognito_password_protection_settings": {
23    "passwordPolicyMinLength": 8,
24    "passwordPolicyCharacters": []
25  },
26  "aws_cognito_verification_mechanisms": [
27    "EMAIL"
28  ]
29 };
30
31 export default awsmobile;
32
```

On AWS side, we created the Amplify Administrator:

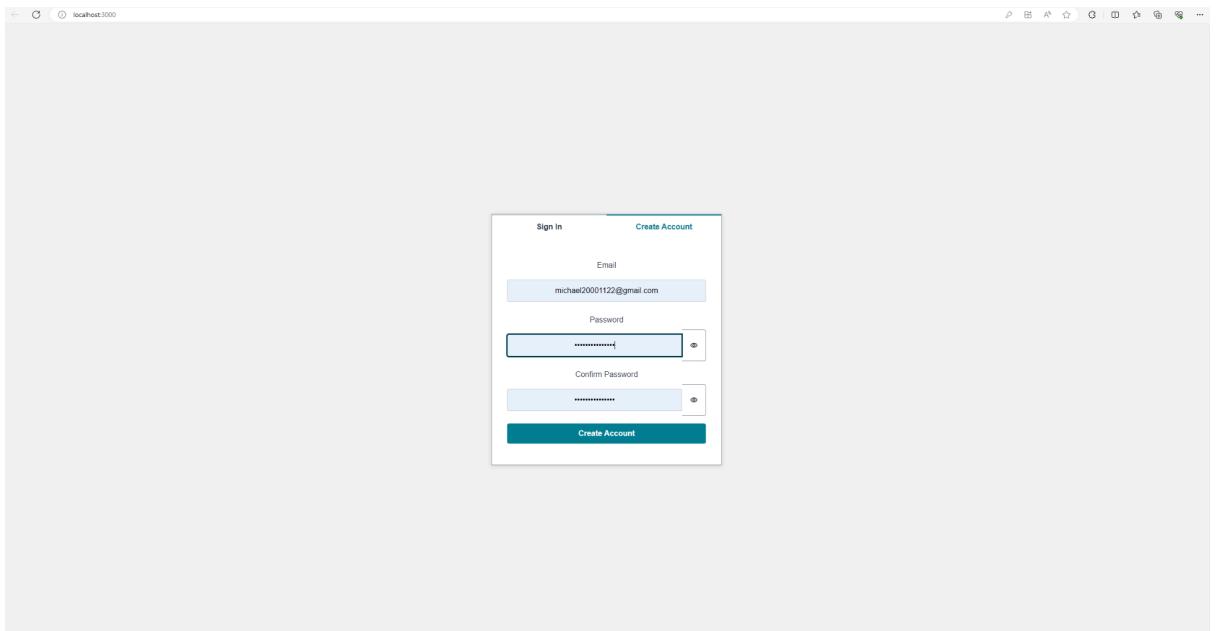
The screenshot shows the AWS Identity and Access Management (IAM) service interface. In the left sidebar, under 'Access management', the 'Users' section is selected. The main content area displays a table titled 'Users (1) info'. The table has one row for the user 'serverless-amplify'. The columns include 'User name' (serverless-amplify), 'Path' (/), 'Last activity' (7 days ago), 'MFA' (none), 'Password age' (none), 'Console last sign-in' (none), 'Access key ID' (arnawsiam:637423511765user/serve...), 'Active key age' (7 days ago), 'Access key last use' (arnawsiam:637423511765user/serve...), 'ARN' (arnawsiam:637423511765user/serve...), and 'Creation date' (8 days ago). There are buttons for 'Create user' and 'Delete' at the top right of the table.

This screenshot shows the detailed information for the 'serverless-amplify' user. The top navigation bar includes 'IAM > Users > serverless-amplify'. The main content is divided into sections: 'Summary', 'Permissions policies (1)', 'Permissions boundary (not set)', and 'Generate policy based on CloudTrail events'. The 'Summary' section shows the ARN (arnawsiam:637423511765user/serve...), console access (Disabled), and last console sign-in (none). It also lists two access keys: 'Access key 1' (AKIAZ2D1QK464K7QUPND - Active, used 8 days ago, 8 days old) and 'Access key 2' (AKIAZ2D1Q5DK2RKLVUJ - Active, used 7 days ago, 7 days old). The 'Permissions policies' section shows a single policy named 'AdministratorAccess' attached directly. The 'Permissions boundary' section is currently empty. The 'Generate policy based on CloudTrail events' section has a 'Generate policy' button and a note about generating a new policy based on recent activity.

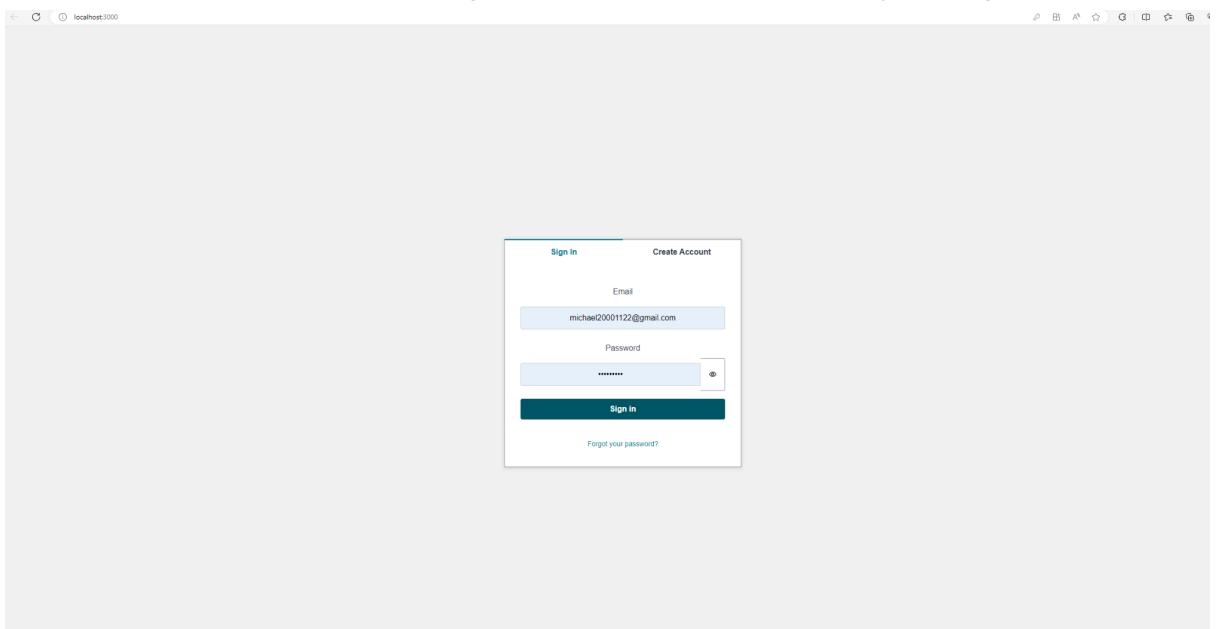
And our React App is successfully linked to our AWS:

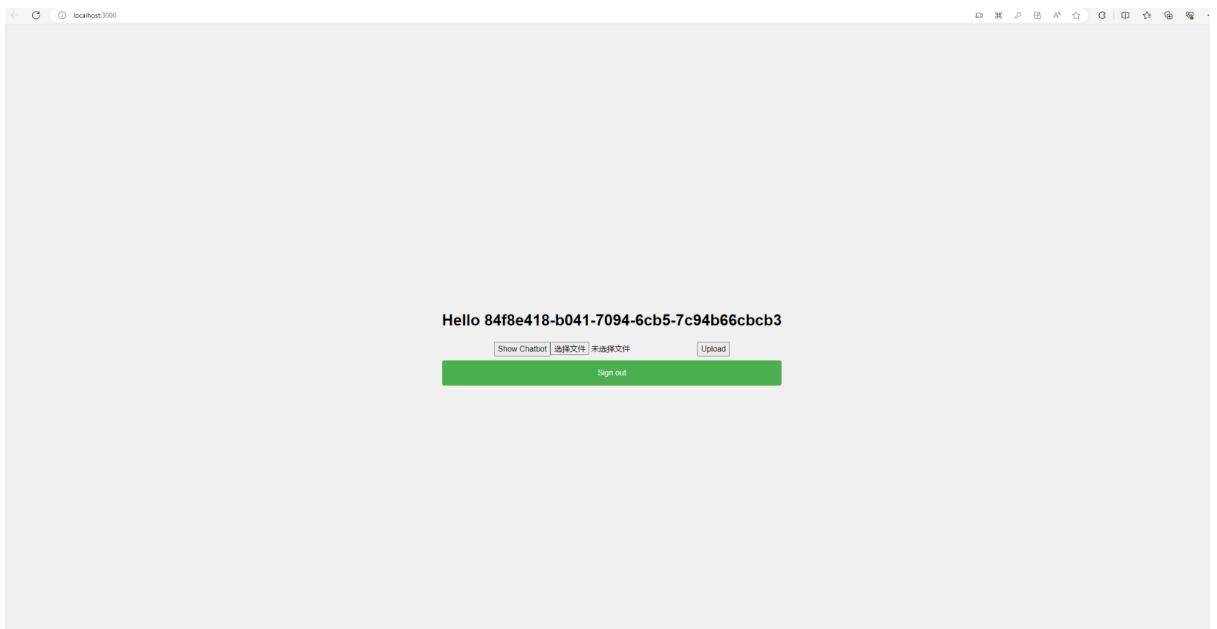
The screenshot shows the AWS Amplify console. The top navigation bar includes 'All apps' and a search bar. Below it, there's a 'Cloudcomputing' app card with a blue icon, showing 'Prod branch' and 'Updated'. To the right of the app card are buttons for 'View app', 'Manage sandboxes', and 'Create new app'. The overall interface is clean and modern, typical of the AWS developer tools.

3. Now let's have a demo run



This is what it looks like on the front end, runs locally on localhost:3000. After creating the new user, we can try to log in.





This is a successful log in. Let's check the backend authentication.

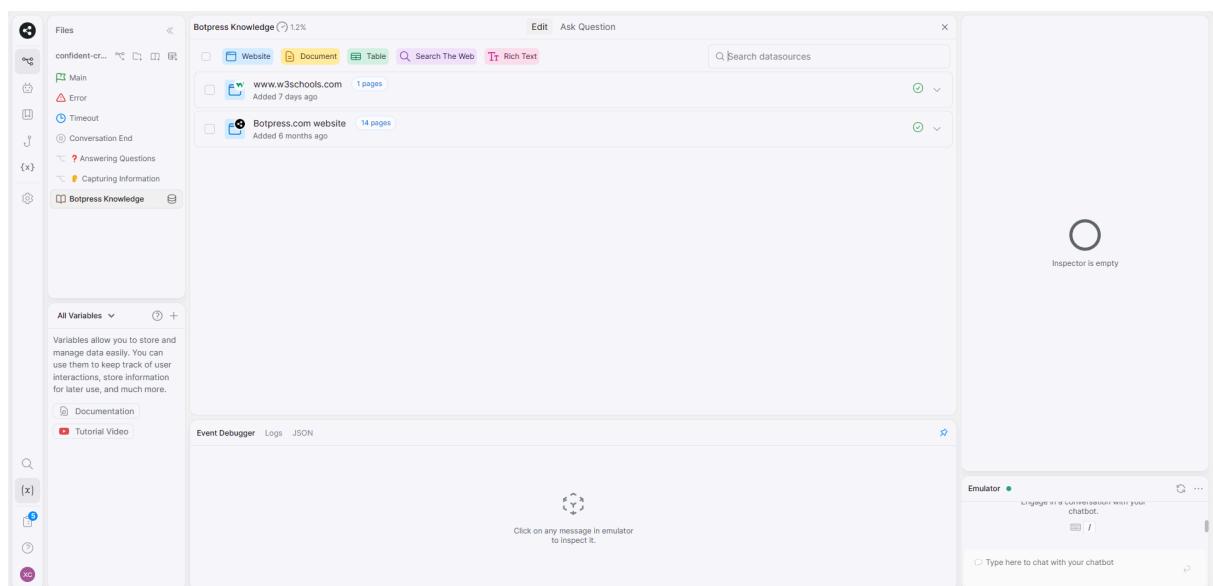
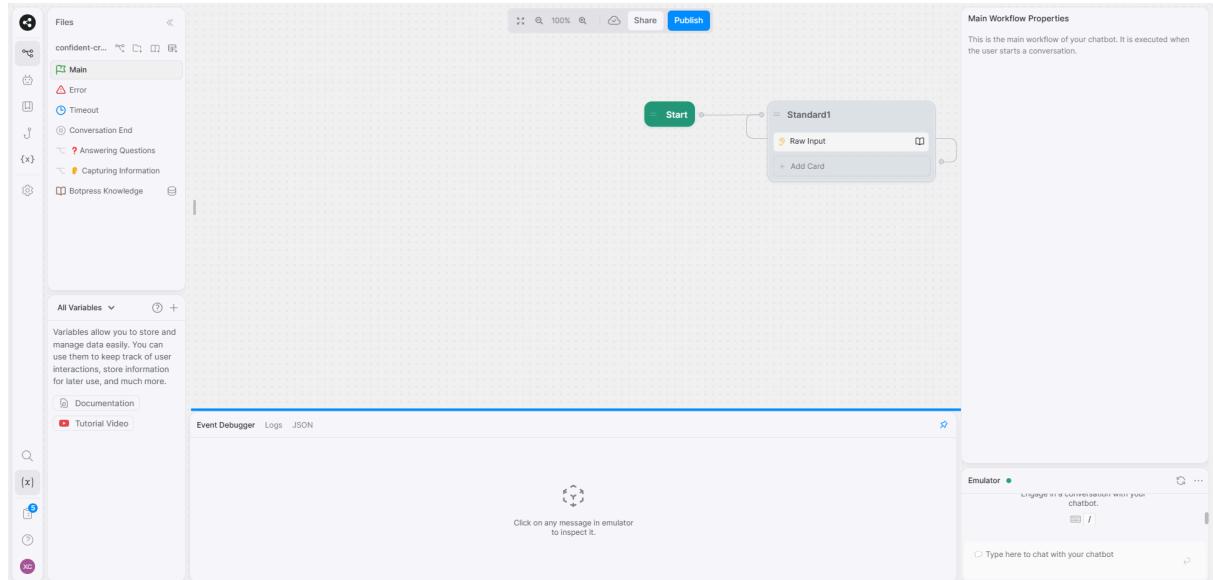
A screenshot of the Amplify Studio interface. On the left, there is a sidebar with various service icons: Home, Manage, Content, User management (which is selected and highlighted in blue), File browser, Design, UI Library, Set up, Data, Authentication, Storage, Functions, GraphQL API, REST API, Analytics, Predictions, Interactions, and Notifications. The main content area is titled "User management" and contains a sub-header "Create, view, and manage users and groups for your application". It shows two tabs: "Users" (selected) and "Groups". Below the tabs is a search bar "Find users by Email". A table lists three users: "Email" (michael20001122@gmail.com, Created Date: May 3, 2024 11:57 PM, Status: CONFIRMED), "xmen975@gmail.com" (Created Date: May 4, 2024 5:35 PM, Status: CONFIRMED), and another row that is partially visible. At the top right of the table are "Actions" and "Create user" buttons.

Here it is on the Amplify Studio. So the process is successfully done.

## 4. Chatbot deployment

Botpress Cloud is a platform that provides hosting and management capabilities for chatbots developed using Botpress, an open-source conversational AI platform. The architecture of Botpress Cloud is designed to support the deployment, scalability, security, and management of chatbot applications.

Our chatbot is developed in the studio of botpress cloud. Currently, the bot only has a handful of websites as its knowledge base. We plan to add a broader knowledge source to the robot in the future and maybe self-learning features.



Our chatbot is hosted on the botpress cloud which runs its own botpress core and database. Integrated with our web app using the prebuild web integration feature.

The image shows two screenshots of a Botpress workspace interface.

**Top Screenshot: Configure webchat**

- Header:** Xinyu Meng's Workspace / confident-crow
- Navigation:** Overview, Integrations (selected), Chat, Logs, Conversations, Analytics, Events, Issues, Configuration variables.
- Section:** Configure webchat
- Bot Information:**
  - Bot name: Amazing Bot
  - Avatar URL: https://example.com/avatar.png
  - Bot description: This is an amazing bot
  - Composer placeholder: Chat with bot
  - Stylesheet URL: https://example.com/style.css
- Switch to the new webchat for a better experience:** A button to switch to the new webchat interface.
- Integration Status:** Enable integration (switch is on), Using classic (switch is off).

**Bottom Screenshot: Logs**

- Header:** Xinyu Meng's Workspace / confident-crow
- Navigation:** Overview, Integrations, Chat, Logs (selected), Conversations, Analytics, Events, Issues, Configuration variables.
- Logs:**
  - From: 2024-05-11 00:00:00 To: 2024-05-12 00:00:00
  - Log entries (partial list):
    - May 12, 2024, 12:13:48 AM [a0b63c2750] [agent-hook] [SummaryAgent] [conversation\_turn\_ended]: Last 10 transcript messages available --> ((conversation.SummaryAgent.transcript))
    - May 12, 2024, 12:13:48 AM [a0b63c2750] [agent-hook] [SummaryAgent] [conversation\_turn\_ended]: Summary updated based on 2 new messages --> ((conversation.SummaryAgent.summary))
    - May 12, 2024, 12:13:49 AM [a0b63c2750] [dm] : FINISHED PROCESSING MESSAGE
    - May 12, 2024, 12:13:49 AM [a0b63c2750] [dm] : Billed Tokens: 22.139 | Total Tokens: 22.139 | Cost: \$0.1054 | Cache Savings: 0%
    - May 12, 2024, 12:13:49 AM [a0b63c2750] [dm] : Conversation conv\_01H0NFR86QDRGH1KNOXKHM02A timed out due to inactivity
    - May 12, 2024, 12:43:57 AM [18e9e8dc45e] [dm] : Transition from (Flow:Timeout) [node:OnTimeout] to [node:Handler]
    - May 12, 2024, 12:43:58 AM [18e9e8dc45e] [dm] : Transition from (Flow:Timeout) [node:Handler] to [node:End]
    - May 12, 2024, 12:43:58 AM [18e9e8dc45e] [dm] : End of timeout workflow
    - May 12, 2024, 12:43:58 AM [18e9e8dc45e] [dm] : Execution "Conversation End" workflow
    - May 12, 2024, 12:43:58 AM [18e9e8dc45e] [dm] : Transition from (Flow:Conversation End) [node:On\_Explicit\_Ending] to [node:Handler]
    - May 12, 2024, 12:43:58 AM [18e9e8dc45e] [dm] : Transition from (Flow:Conversation End) [node:Handler] to [node:End]
    - May 12, 2024, 5:45:52 PM [56446fb8e5] [dm] : Event of type 'text' received from user #U01H0NFR86QDRGH1KNOXKHM02A on webhook
    - May 12, 2024, 5:45:52 PM [56446fb8e5] [agent-hook] [KnowledgeAgent] [conversation\_turn\_started]: Message of type 'text' received from user #U01H0NFR86QDRGH1KNOXKHM02A on webhook
    - May 12, 2024, 5:45:53 PM [56446fb8e5] [agent-hook] [KnowledgeAgent] [conversation\_turn\_started]: Extracted 1 questions from user message
    - May 12, 2024, 5:45:53 PM [56446fb8e5] [agent-hook] [KnowledgeAgent] [conversation\_turn\_started]: Generating answer based on 50 results
    - May 12, 2024, 5:45:58 PM [56446fb8e5] [dm] : Sending Message. Type: text. Text: AWS stands for Amazon Web Services, which is one of the major cloud providers. You can do everything in the AWS cloud.
    - May 12, 2024, 5:45:58 PM [56446fb8e5] [dm] : Starting from Main flow
    - May 12, 2024, 5:45:58 PM [56446fb8e5] [dm] : Transition from (Flow:Start) [node:Start] to [node:Standard1]
    - May 12, 2024, 5:45:58 PM [56446fb8e5] [dm] : [Captured] Start information capture
    - May 12, 2024, 5:45:58 PM [56446fb8e5] [dm] : Sending Message. Type: text. Text: What is your question?
    - May 12, 2024, 5:45:58 PM [56446fb8e5] [dm] : Waiting for user input
    - May 12, 2024, 5:45:58 PM [56446fb8e5] [agent-hook] [SummaryAgent] [conversation\_turn\_ended]: Last 10 transcript messages available --> ((conversation.SummaryAgent.transcript))
    - May 12, 2024, 5:45:58 PM [56446fb8e5] [agent-hook] [SummaryAgent] [conversation\_turn\_ended]: Summary updated based on 3 new messages --> ((conversation.SummaryAgent.summary))
    - May 12, 2024, 5:45:59 PM [56446fb8e5] [dm] : Stack trace for event
    - May 12, 2024, 5:46:00 PM [56446fb8e5] [dm] : Billed Tokens: 21.978 | Total Tokens: 21.978 | Cost: \$0.1061 | Cache Savings: 0%
    - May 12, 2024, 5:46:00 PM [56446fb8e5] [dm] : FINISHED PROCESSING MESSAGE
    - May 12, 2024, 6:16:09 PM [4c088105cb] [dm] : Conversation conv\_01H0X0C99R8K422ME15SF9BEG3 timed out due to inactivity
    - May 12, 2024, 6:16:09 PM [4c088105cb] [dm] : Transition from (Flow:Timeout) [node:OnTimeout] to [node:Handler]
    - May 12, 2024, 6:16:09 PM [4c088105cb] [dm] : Transition from (Flow:Timeout) [node:Handler] to [node:End]
    - May 12, 2024, 6:16:09 PM [4c088105cb] [dm] : Executing "Conversation End" workflow
    - May 12, 2024, 6:16:09 PM [4c088105cb] [dm] : End of timeout workflow
    - May 12, 2024, 6:16:09 PM [4c088105cb] [dm] : Transition from (Flow:Conversation End) [node:On\_Explicit\_Ending] to [node:Handler]
    - May 12, 2024, 6:16:09 PM [4c088105cb] [dm] : Transition from (Flow:Conversation End) [node:Handler] to [node:End]

## 5. Azure Blob Storage initialize

```
npm install @azure/storage-blob
```

And here is our storage built:

and generate SAS key from here:

Also manage CORS to allow all the methods:

Finally by adding these two methods in React:

```
async function uploadToBlobStorage() {
  let accountName = "zjservice";
  let sas =
    `sv=2022-11-02&ss=bfqt&srt=sco&sp=rwdlacuiytfx&se=2024-05-12T11:14:49Z&st=2024-05-12T03:14:49Z&spr=https&sig=rnbrkhz%2BAroAQD3M2CynDsH35Z9mS%2FA`;
  const blobServiceClient = new BlobServiceClient(
    `https://${accountName}.blob.core.windows.net?${sas}`);
  const containerClient = blobServiceClient.getContainerClient("files");
  await containerClient.createIfNotExists({
    access: "container",
  });

  const blobClient = containerClient.getBlockBlobClient(file.name);
  const options = { blobHTTPHeaders: { blobContentType: file.type } };

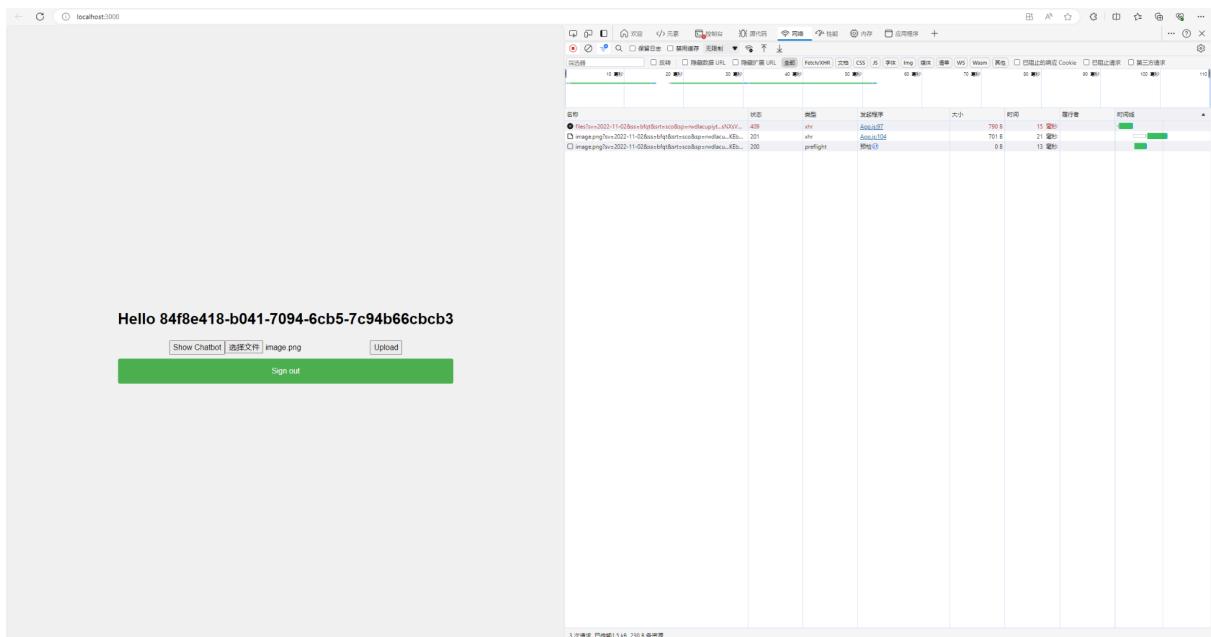
  await blobClient.uploadBrowserData(file, options);
}

function handleFileChange(event) {
  setFile(event.target.files[0]);
}
```

We finished the upload function.

Let's have a test:

Randomly select “image.png”



And we can see on the right-hand side, the Network console showed the status changes. We can see “201” as created and “200” as “OK” which stands for upload successfully. Let's check the Azure container.



It's successfully uploaded on Azure.

The screenshot shows the Microsoft Azure Storage Explorer interface. It displays a list of blobs in a container named 'files'. The blobs are:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
会议 2024年4月1日.pdf	5/1/2024, 11:51:10 PM	Hot (inferred)		Block blob	957.89 KB	Available
final chatheet.pdf	5/1/2024, 11:55:56 PM	Hot (inferred)		Block blob	2.57 MB	Available
Image.png	5/12/2024, 5:49:21 PM	Hot (inferred)		Block blob	9.67 KB	Available
IMG_8489.jpg	5/11/2024, 11:59:40 PM	Hot (inferred)		Block blob	136 KB	Available
training.history.xlsx	5/12/2024, 5:09:54 PM	Hot (inferred)		Block blob	6.19 KB	Available

## 6. Adding P2P

Used the chatbox sample from: [Building a P2P Chat Room Web App with React and LioWebRTC | by Lazorfuzz | Medium](#)

But we don't use the LioWebRTC, cause it's a very old version and won't be compatible with our node version.

We decide to upload the chatLogs to Azure Blob Storage and fetch it down to show the chatLog history.

Before adding the P2P functionalities, here is the raw chatbox built.

cloud-computing

EXPLORER    OPEN EDITORS    CLOUD-COMPUTING

```

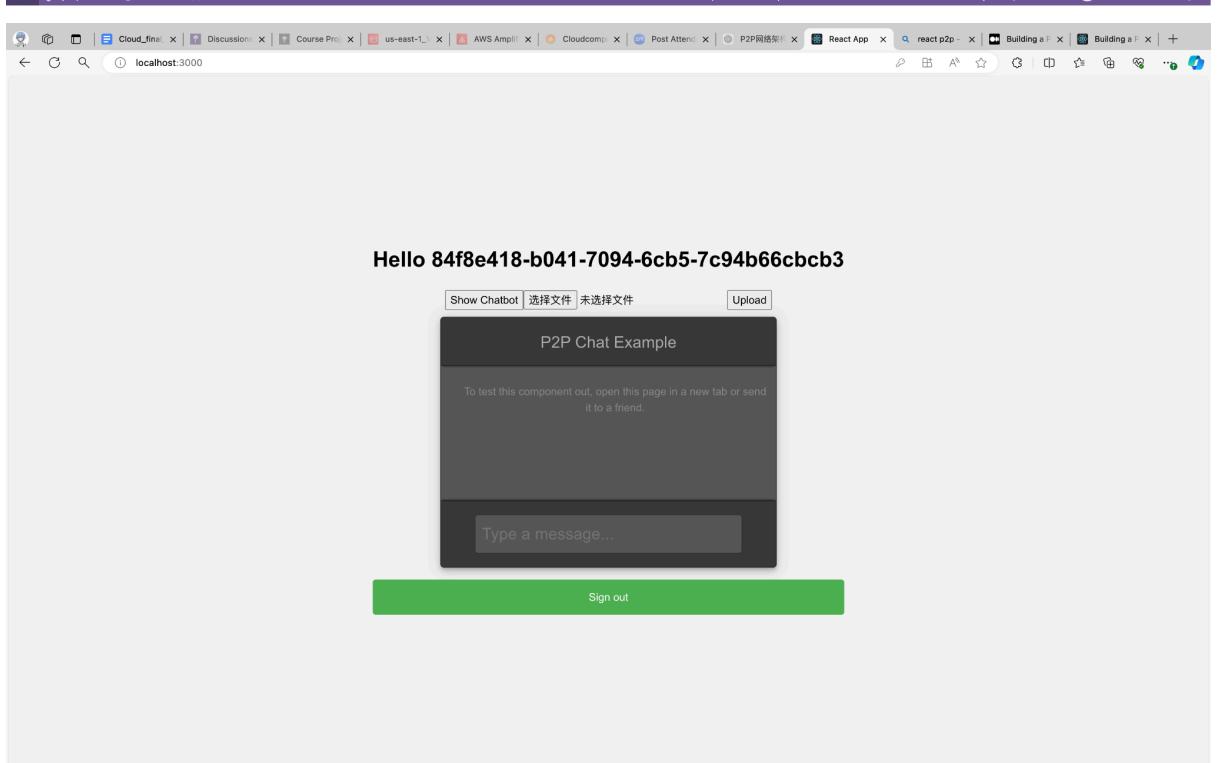
src > Chatbox.js > ChatBox > handleSend
1 import React, { Component } from "react";
2 import "./Chatbox.css";
3
4 class ChatBox extends Component {
5   constructor(props) {
6     super(props);
7     this.state = { inputMsg: "" };
8   }
9   generateChats = () => {
10     if (this.chatBox) {
11       setTimeout(() => {
12         this.chatBox.scrollTop = this.chatBox.scrollHeight;
13       }, 2);
14     }
15     return this.props.chatLog.map((item) => (
16       <div className="chat" key={`chat-${item.name}-${item.timestamp}`}>
17         <b className="name" style={{ color: item.alert ? "#888" : "#333" }}>
18           {item.name}
19         </b>{" "}
20         <span className="msg">{item.message}</span>{" "}
21       </div>
22     ));
23   };
24   handleSend = (chatMsg) => {
25     this.props.onSend(chatMsg);
26   };
27   handleKeyUp = (evt) => {
28     if (evt.keyCode === 13) {
29       this.handleSend(this.state.inputMsg);
30       this.setState({ inputMsg: "" });
31     }
32   };
33 }
34
35 
```

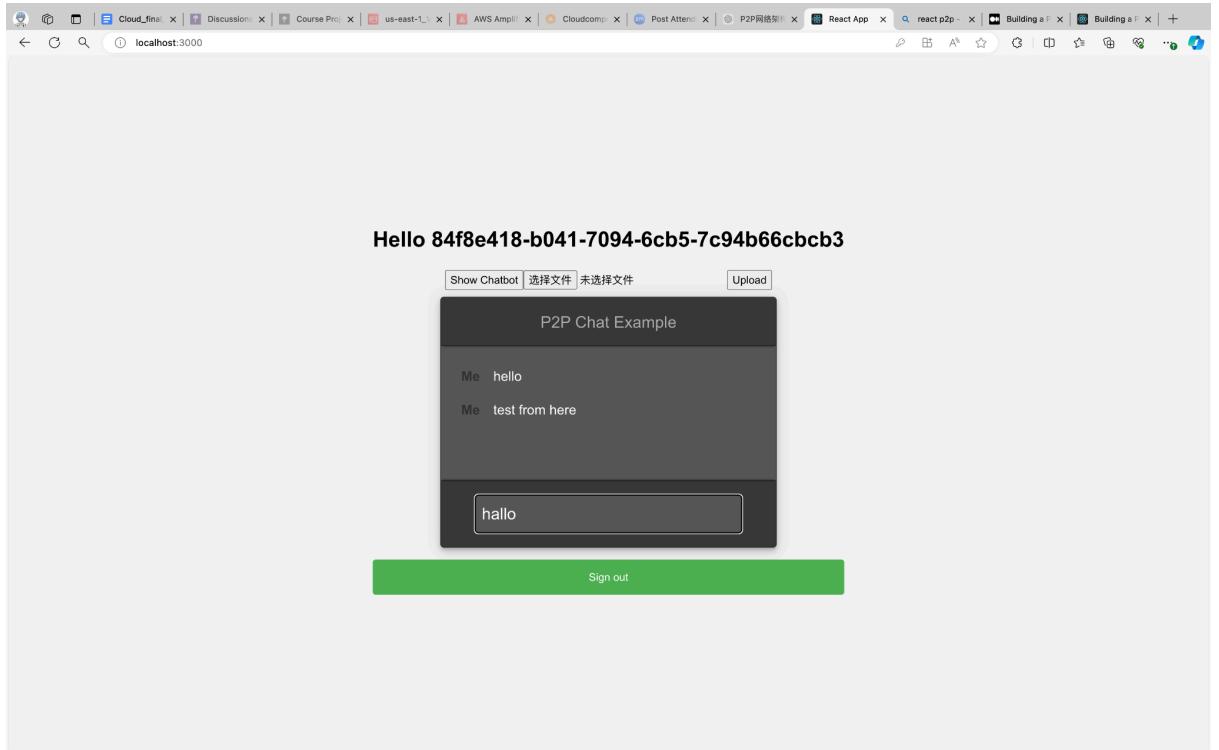
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

node    zsh

SonarLint focus: overall code

Ln 26, Col 32    Spaces: 4    UTF-8    LF    JavaScript    Go Live    Prettier



A screenshot of the Microsoft Azure Storage Container named "chatlogs". The container overview shows a list of uploaded files. The table has the following columns:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
chatlog-2024-05-15T00:19:14.706Z.json	5/14/2024, 8:19:14 PM	Hot (Inferred)		Block blob	2 B	Available
chatlog-2024-05-15T00:21:26.494Z.json	5/14/2024, 8:21:26 PM	Hot (Inferred)		Block blob	75 B	Available
chatlog-2024-05-15T00:23:15.870Z.json	5/14/2024, 8:23:15 PM	Hot (Inferred)		Block blob	149 B	Available
chatlog-2024-05-15T00:25:23.355Z.json	5/14/2024, 8:25:23 PM	Hot (Inferred)		Block blob	2 B	Available
chatlog-2024-05-15T00:27:59.222Z.json	5/14/2024, 8:27:59 PM	Hot (Inferred)		Block blob	63 B	Available
chatlog-2024-05-15T00:29:10.196Z.json	5/14/2024, 8:29:10 PM	Hot (Inferred)		Block blob	2 B	Available
chatlog-2024-05-15T00:29:24.089Z.json	5/14/2024, 8:29:24 PM	Hot (Inferred)		Block blob	2 B	Available
chatlog-2024-05-15T00:29:30.100Z.json	5/14/2024, 8:29:30 PM	Hot (Inferred)		Block blob	75 B	Available
chatlog-2024-05-15T00:35:59.599Z.json	5/14/2024, 8:35:59 PM	Hot (Inferred)		Block blob	75 B	Available
chatlog-2024-05-15T00:42:05.082Z.json	5/14/2024, 8:42:05 PM	Hot (Inferred)		Block blob	75 B	Available
chatlog-2024-05-15T00:42:50.591Z.json	5/14/2024, 8:42:50 PM	Hot (Inferred)		Block blob	147 B	Available
chatlog-2024-05-15T00:43:26.388Z.json	5/14/2024, 8:43:26 PM	Hot (Inferred)		Block blob	220 B	Available
chatlog-2024-05-15T00:43:51.811Z.json	5/14/2024, 8:43:51 PM	Hot (Inferred)		Block blob	292 B	Available
chatlog-2024-05-15T00:44:47.640Z.json	5/14/2024, 8:44:47 PM	Hot (Inferred)		Block blob	365 B	Available
chatlog-2024-05-15T00:44:53.543Z.json	5/14/2024, 8:44:53 PM	Hot (Inferred)		Block blob	437 B	Available
chatlog-2024-05-15T00:47:51.199Z.json	5/14/2024, 8:47:51 PM	Hot (Inferred)		Block blob	75 B	Available
chatlog-2024-05-15T00:47:58.079Z.json	5/14/2024, 8:47:58 PM	Hot (Inferred)		Block blob	151 B	Available
chatlog-2024-05-15T00:48:38.652Z.json	5/14/2024, 8:48:38 PM	Hot (Inferred)		Block blob	223 B	Available
chatlog-2024-05-15T00:49:24.748Z.json	5/14/2024, 8:49:48 PM	Hot (Inferred)		Block blob	76 B	Available

Used these two functions to Upload and Download the chatLogs

```
async function uploadChatLog(chatLog) {
  const accountName = "zjservice";
  const sas =
    `sv=2022-11-02&ss=bfqt&srt=sco&sp=rwdlacupiytfx&se=2024-05-15T08:16:18Z&st=2024-05-15T00:16:18Z&spr=https&sig=sWXv
  const blobServiceClient = new BlobServiceClient(
    `https://${accountName}.blob.core.windows.net?${sas}`
  );
  const containerClient = blobServiceClient.getContainerClient("chatlogs");
  await containerClient.createIfNotExists({
    access: "container",
  });

  const blobName = `chatlog-${new Date().toISOString()}.json` // 创建一个基于时间的唯一文件名
  setFileName((fileName) => [...fileName, blobName]);
  // console.log(fileName.length);
  const blobClient = containerClient.getBlockBlobClient(blobName);
  const options = [
    blobHTTPHeaders: { blobContentType: "application/json" },
  ];

  const chatLogData = JSON.stringify(chatLog);
  console.log("ChatLogData: ", chatLogData);
  await blobClient.uploadData(chatLogData, options);
}
```

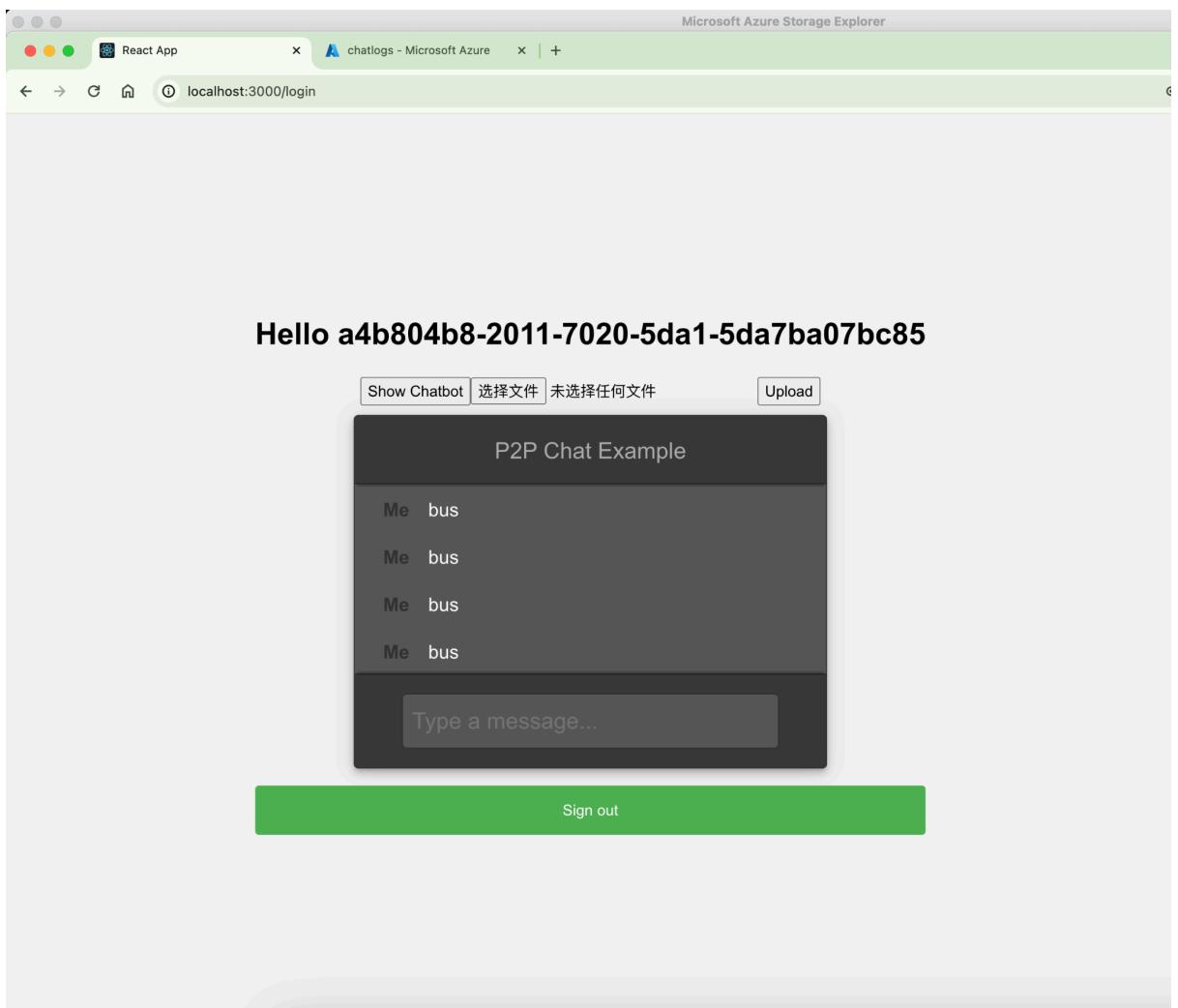
```
async function downloadChatLog() {
  console.log("FileName Array", fileName);

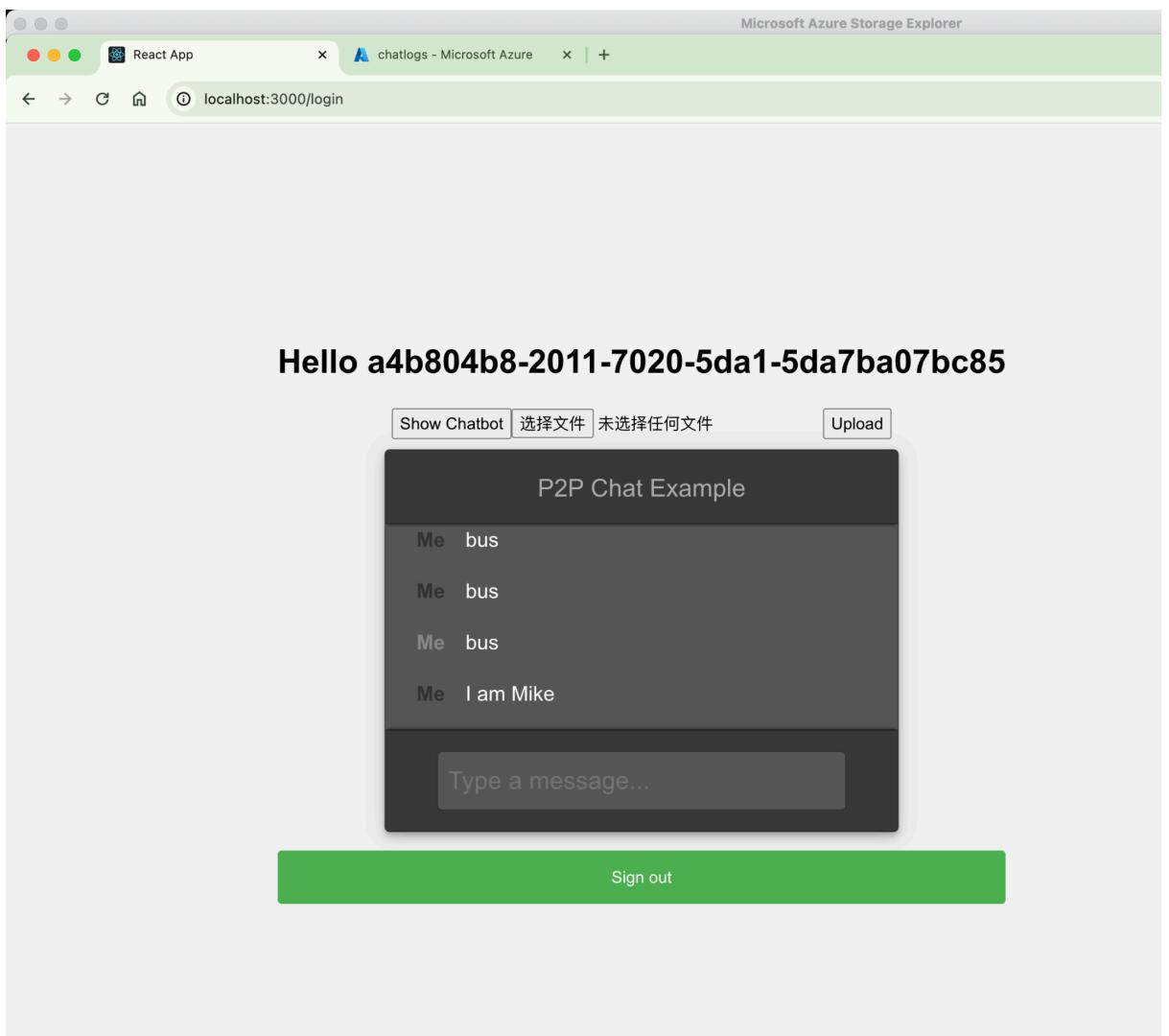
  if (fileName.length === 0) {
    return;
  }
  const accountName = "zjservice";
  const sas =
    `sv=2022-11-02&ss=bfqt&srt=sco&sp=rwdlacupiytfx&se=2024-05-15T08:16:18Z&st=2024-05-15T00:16:18Z&spr=https&sig=sWXv
  const blobServiceClient = new BlobServiceClient(
    `https://${accountName}.blob.core.windows.net?${sas}`
  );
  const containerClient = blobServiceClient.getContainerClient("chatlogs");

  // 假设你知道文件名或者通过某种方式获取最新的文件名

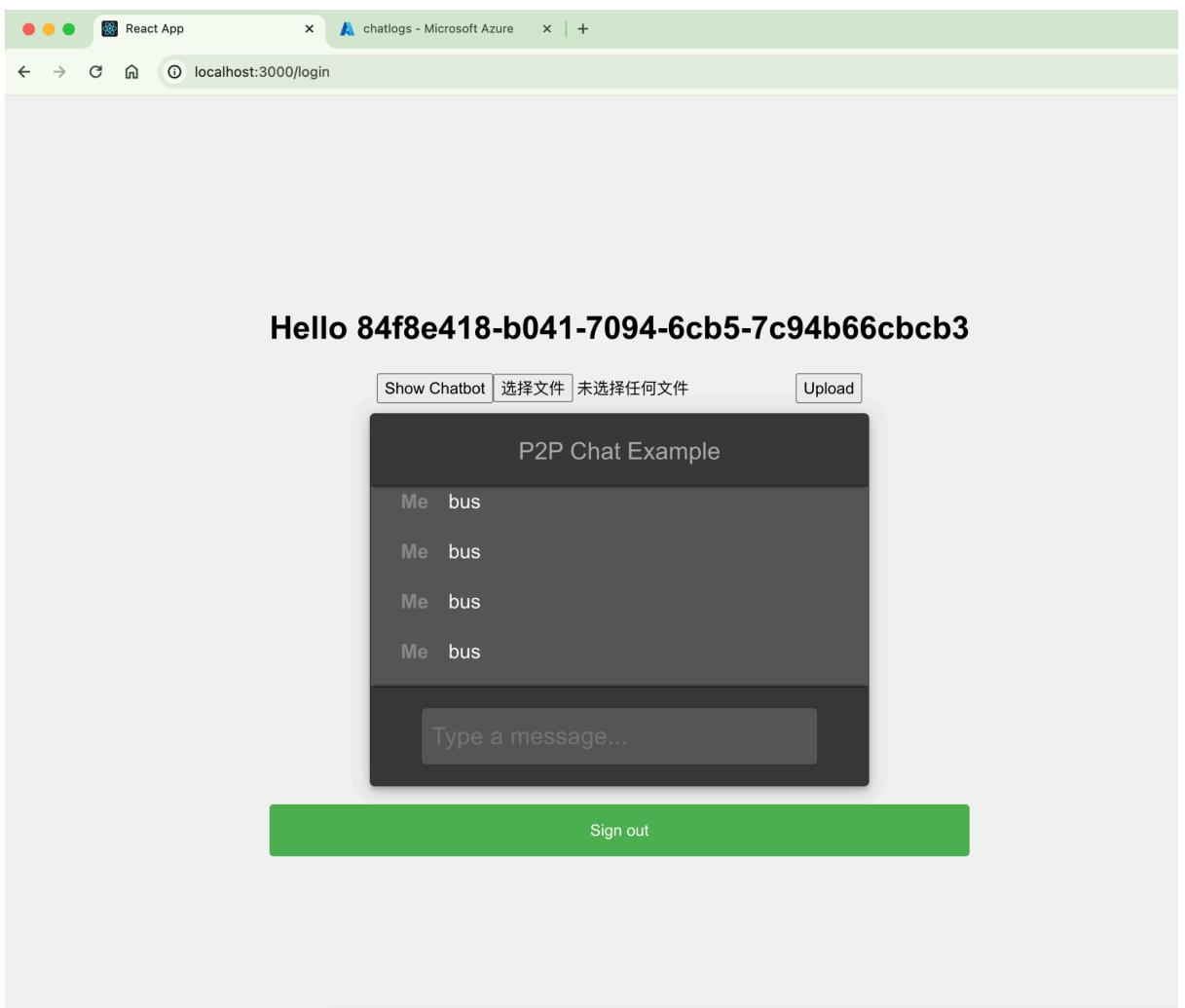
  const blobName = fileName[fileName.length - 1];
  // const blobName = "chatlog-2024-05-15T00:44:53.543Z.json"
  const blobClient = containerClient.getBlobClient(blobName);

  const downloadBlockBlobResponse = await blobClient.download(0);
  const downloaded = await blobToString(
    await downloadBlockBlobResponse.blobBody
  );
  console.log("Downloaded Data: ", JSON.parse(downloaded));
  return JSON.parse(downloaded);
}
```





Therefore, two different accounts can chat in this chatbox. As whenever there's a change in the chatLogs, it will store to the Azure Blob and then fetch it down. Different color of "Me" means two different users.



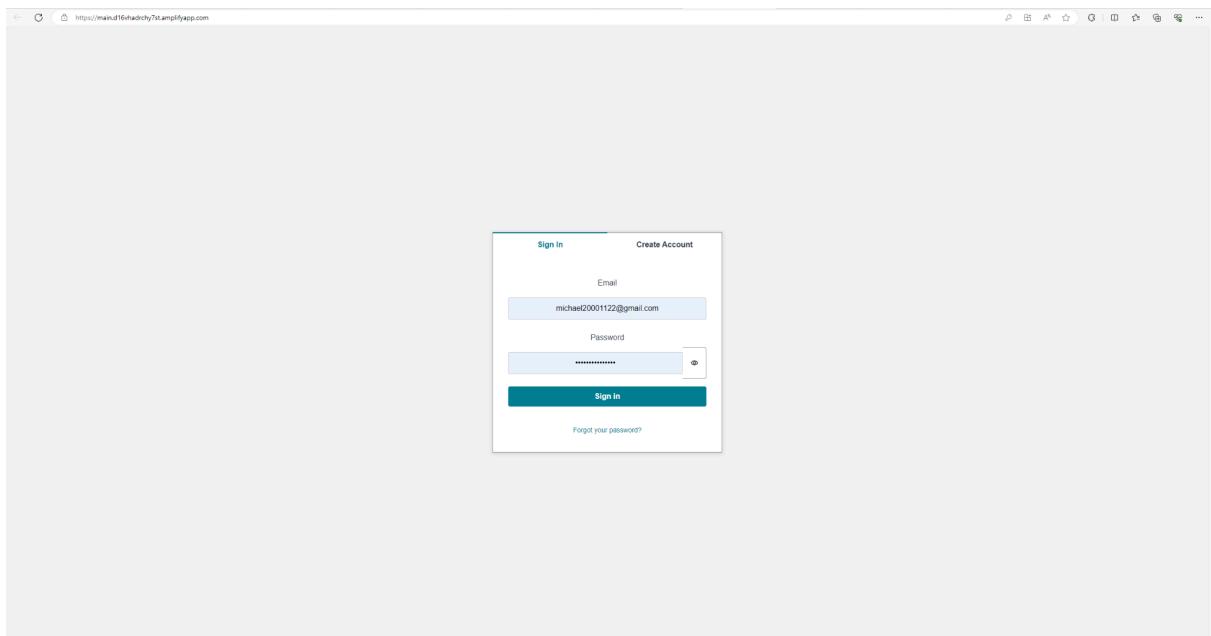
## 7. App deployment

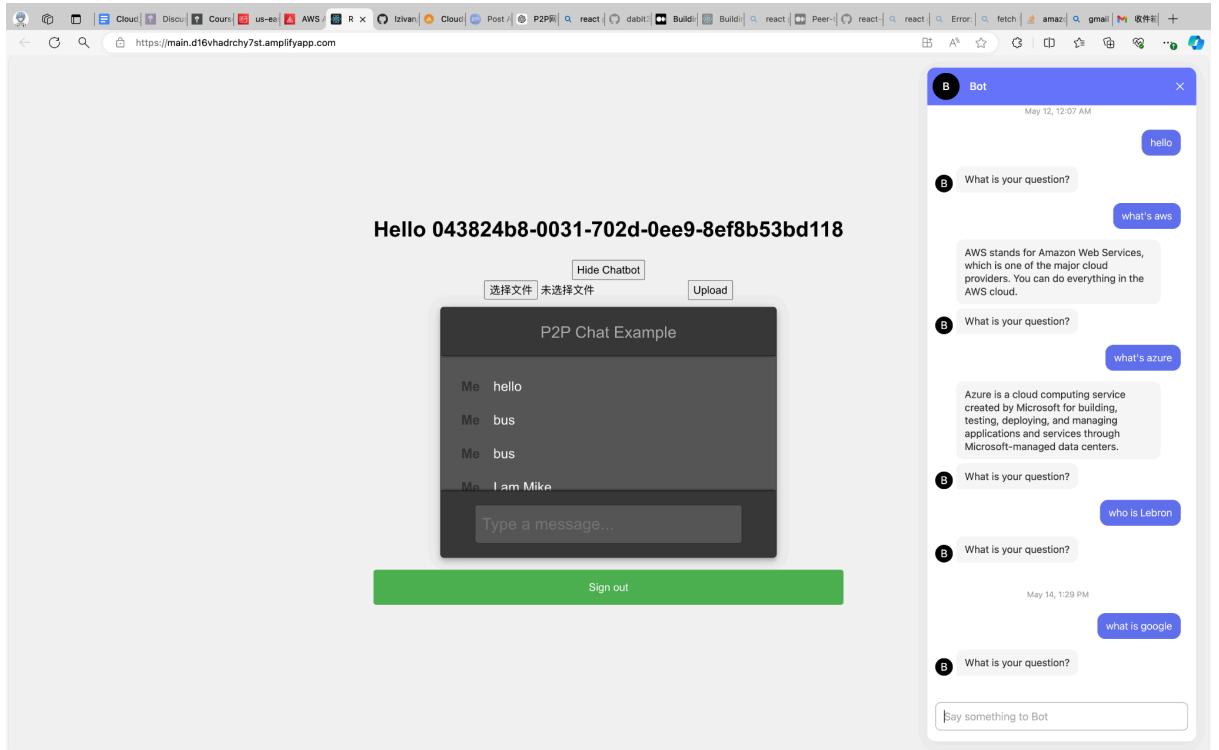
We decided to deploy our React app to AWS amplify. So, we cloned our Github Repository to Amplify, and used the main branch for deployment.

The screenshot shows the AWS Amplify console interface. On the left, a sidebar lists various project components: main, Deployments, Authentication, Data, Functions, and UI Library. The main content area is titled "Deployments" and shows a deployment for "Deployment 13". The deployment status is "Deployed" with a timestamp of "2 minutes 56 seconds ago". It includes a log section with 220 entries, a "Repository" section showing "Cloudcomputing/main", and a "Domain" section with "https://main.d16vhadrhy7st.amplifyapp.com". A "Redeploy" button is visible. Below the deployment details, there's a "Deployment history" table with three rows: Deployment 12 (Status: Deployed), Deployment 11 (Status: Failed), and Deployment 10 (Status: Failed). The bottom of the page includes links for "CloseShell", "Feedback", and copyright information.

Whenever the main branch is updated, the service will auto compile and auto deploy.

And then it runs on: <https://main.d16vhadrhy7st.amplifyapp.com/>





All the functions are restored. Deployed successfully.