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

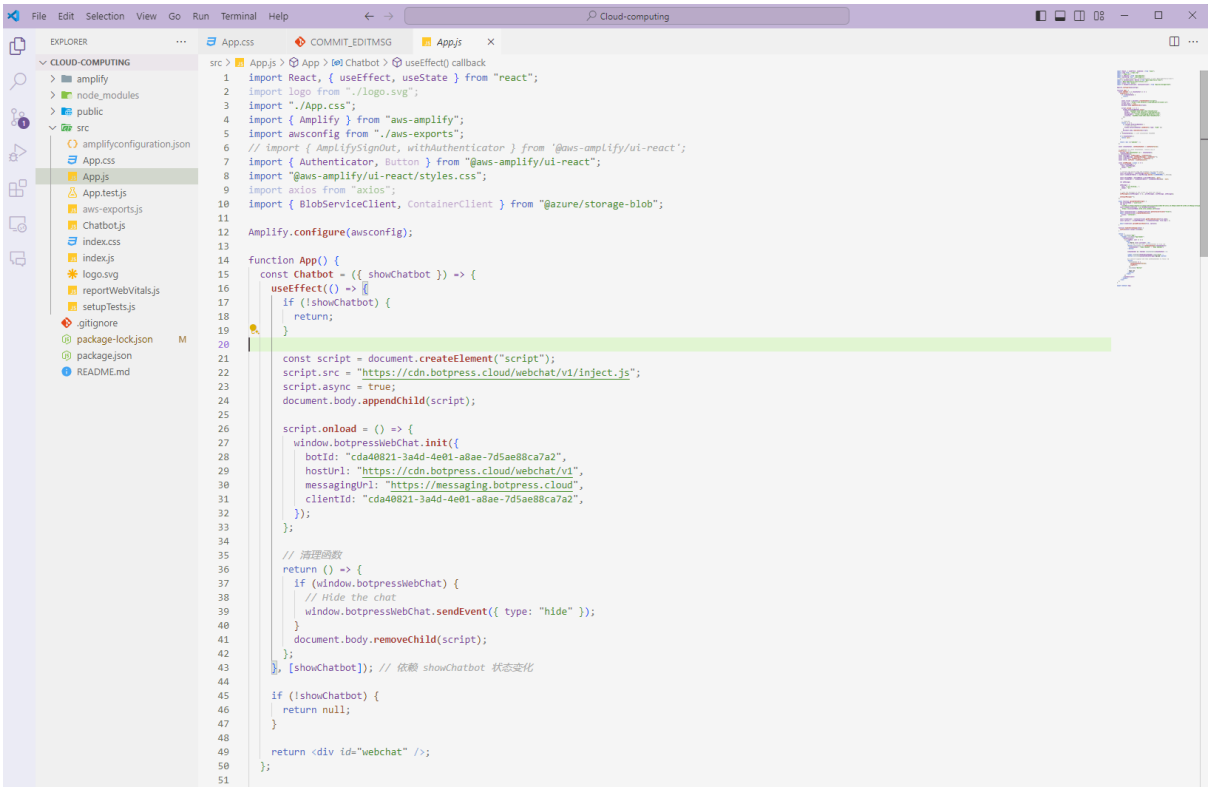
- AWS: Widely recognized for its comprehensive services, robust hosting capabilities, and strong enterprise support.
- Azure: Known for its integration with Microsoft products and services, making it ideal for organizations reliant on Windows servers, SQL Server, and other Microsoft technologies.
- Botpress: 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



2. Next initialize the amplify platform

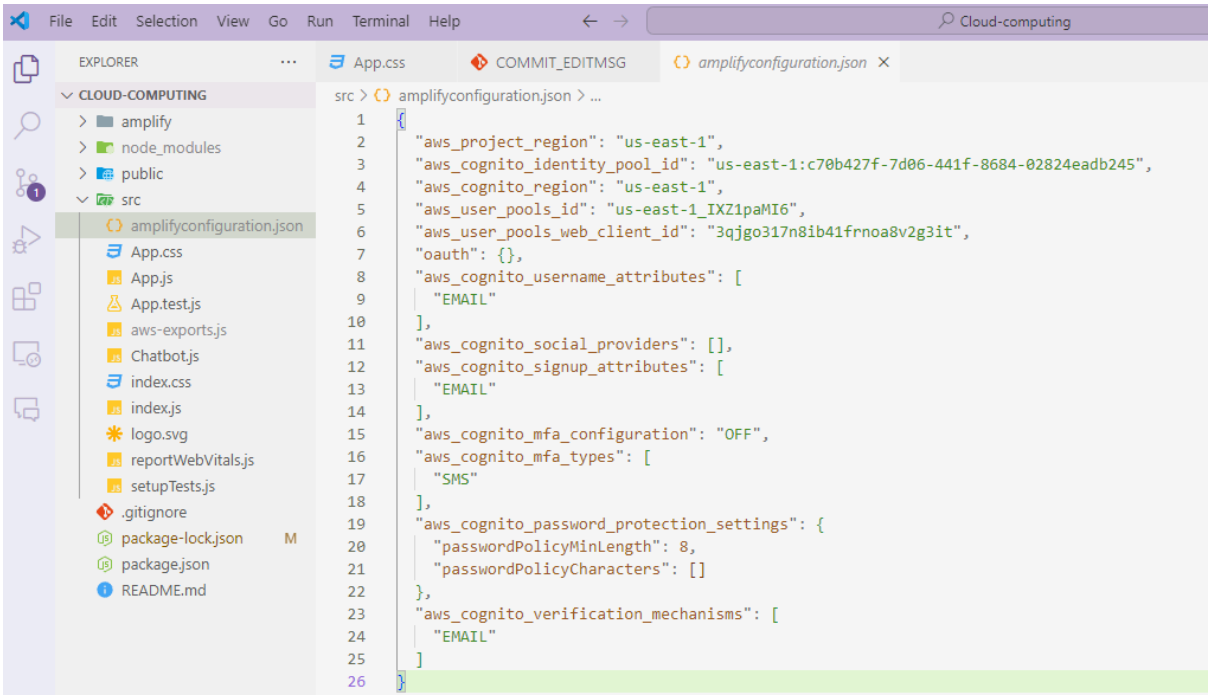
   amplify configure

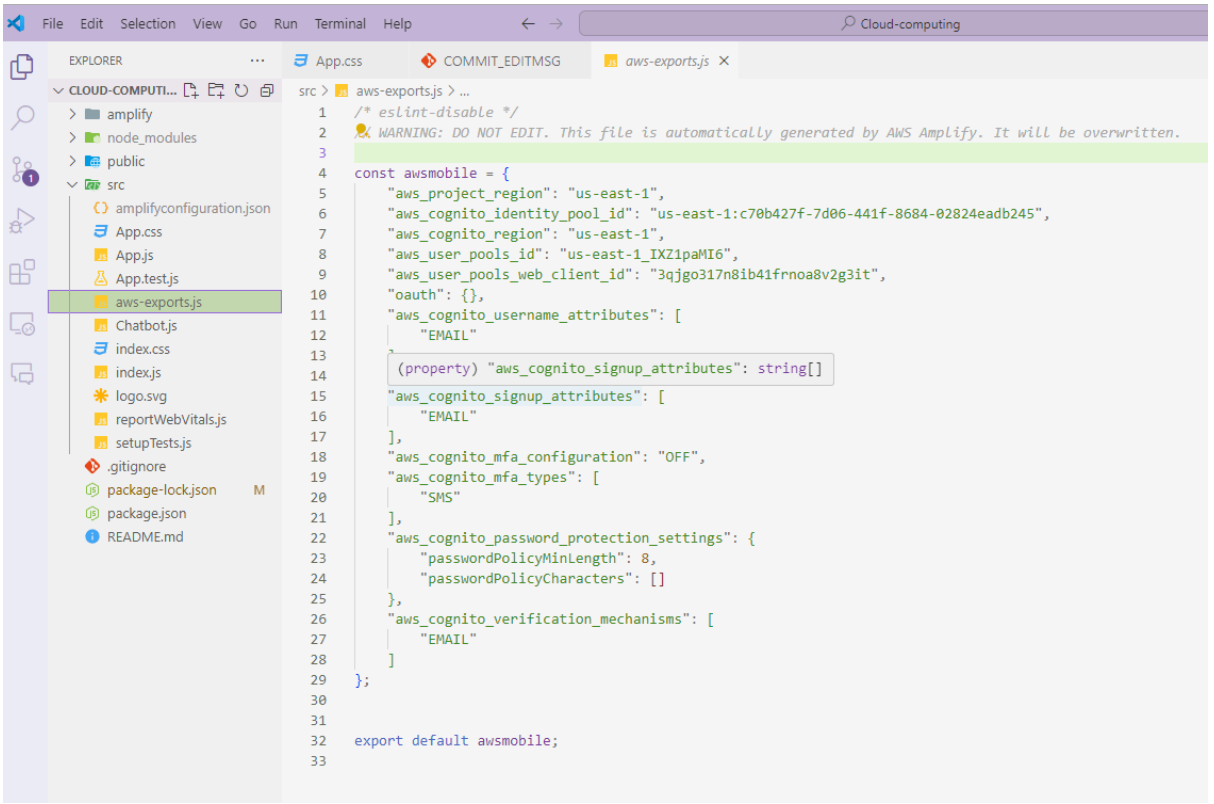
   amplify init

   amplify add auth

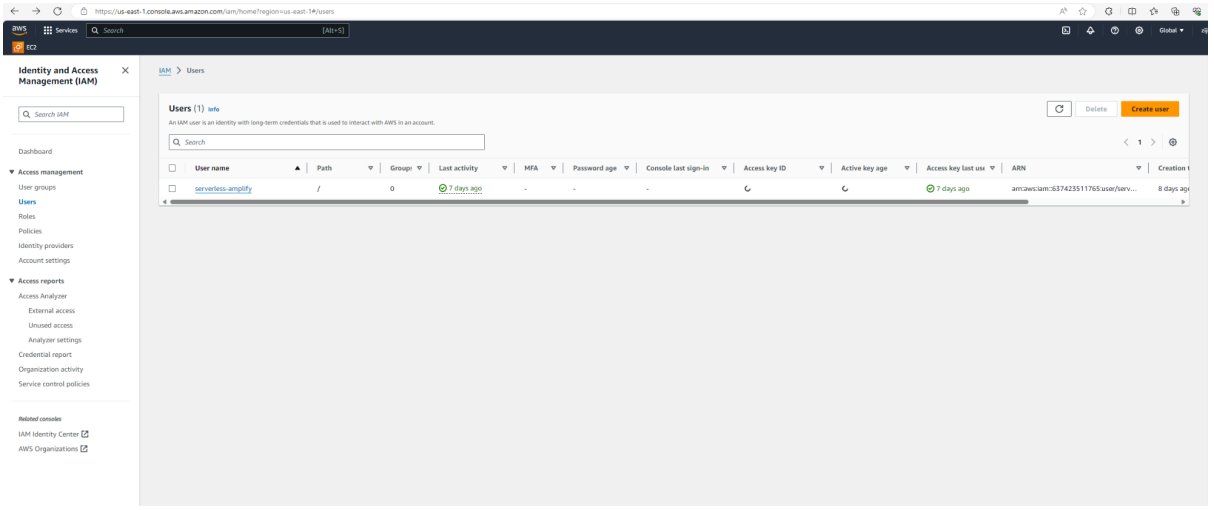
   amplify push

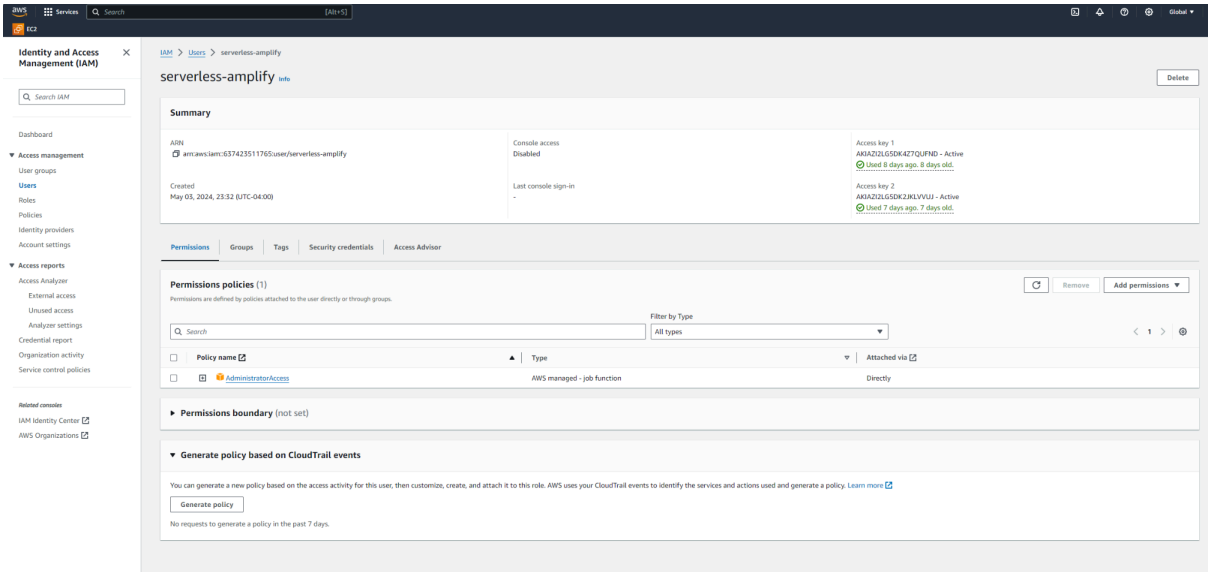
   Now, the amplify authentication is successfully linked.



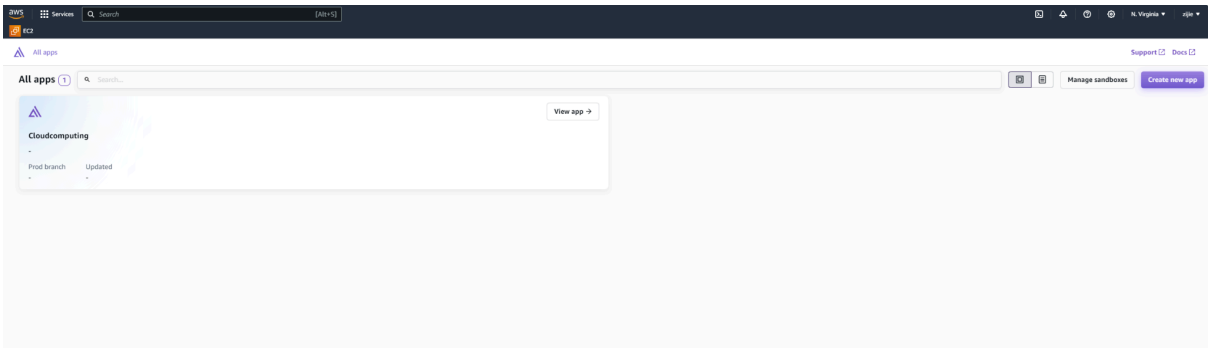


On AWS side, we created the Amplify Administrator:

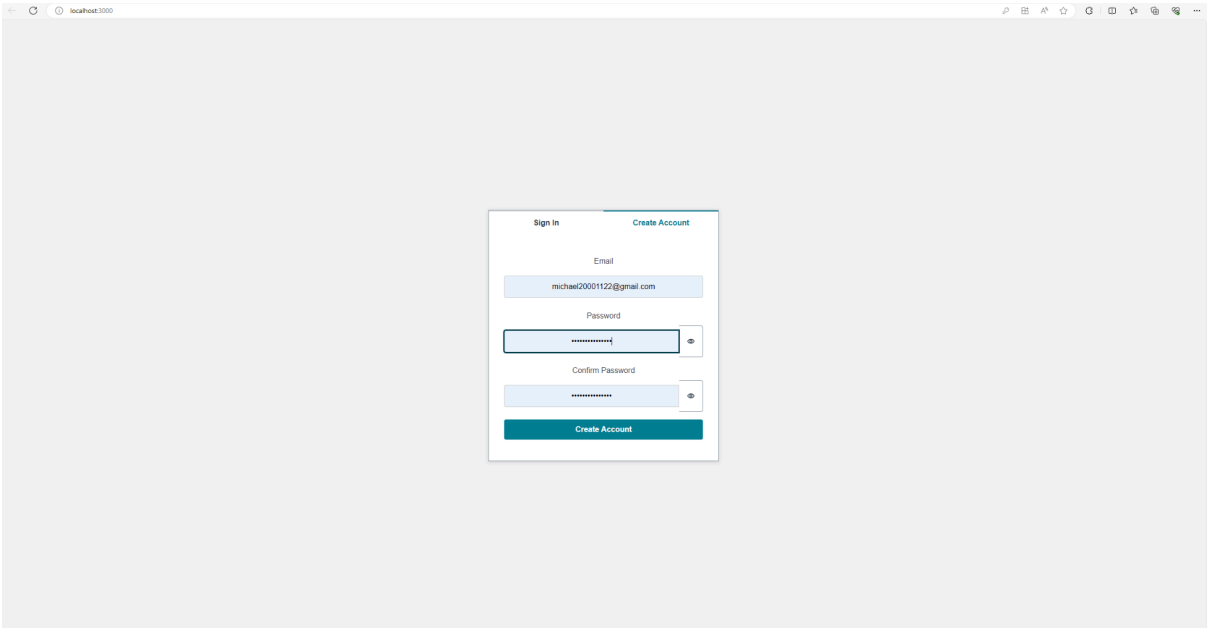




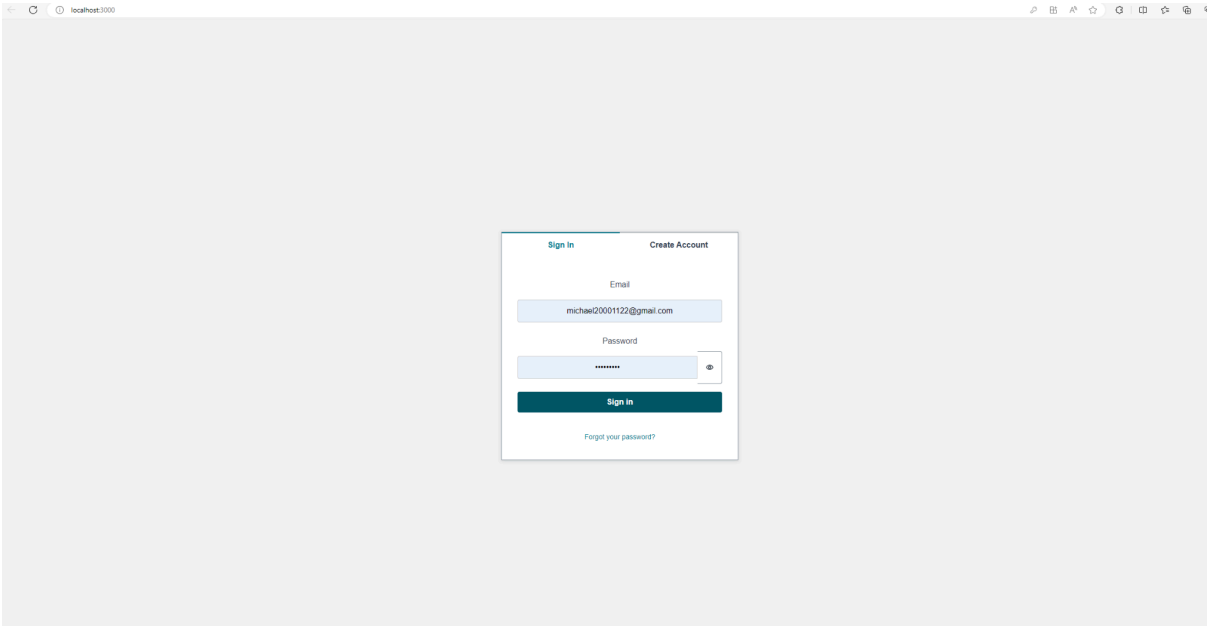
And our React App is successfully linked to our AWS:

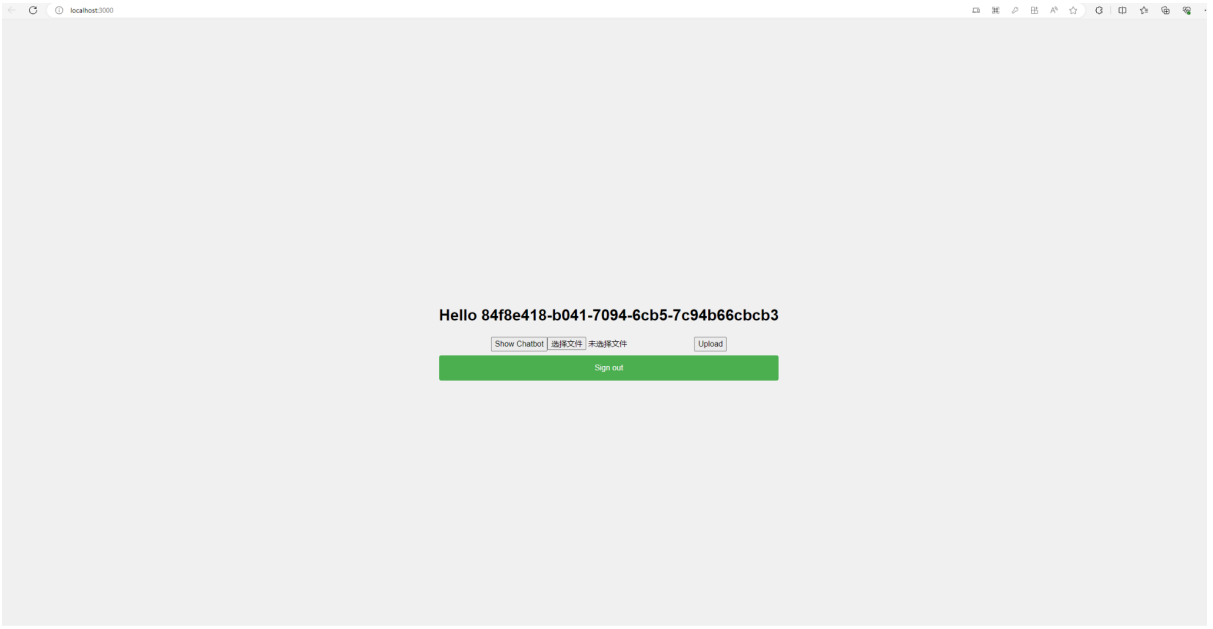


3. Now let’s have a demo run

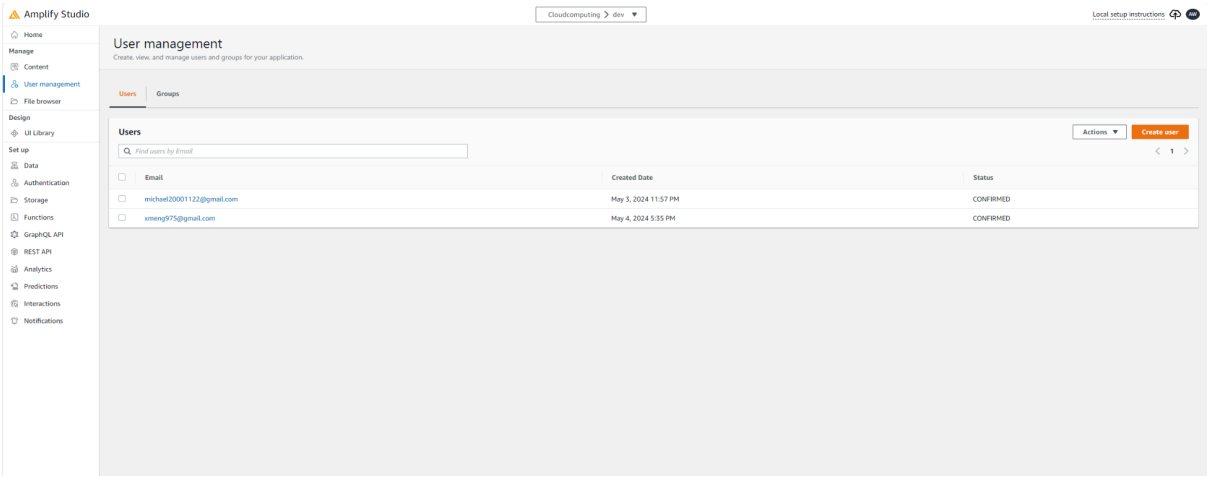


This is what it looks like on the front end, runs locally on localhost:3000. After creating the new user, we can try to log in.





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

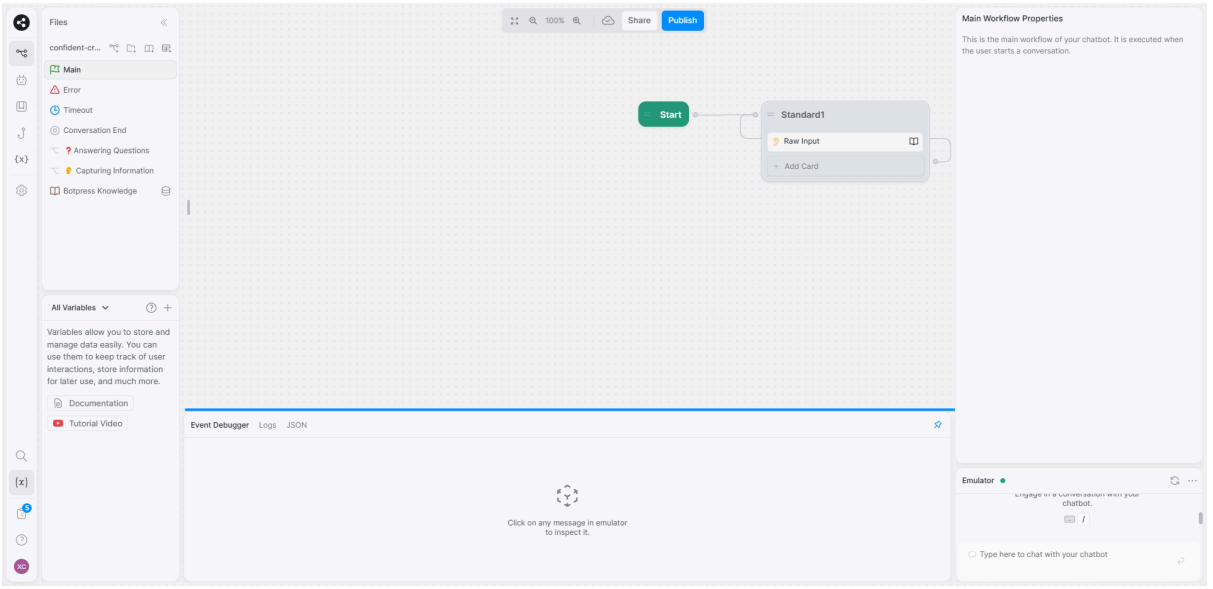


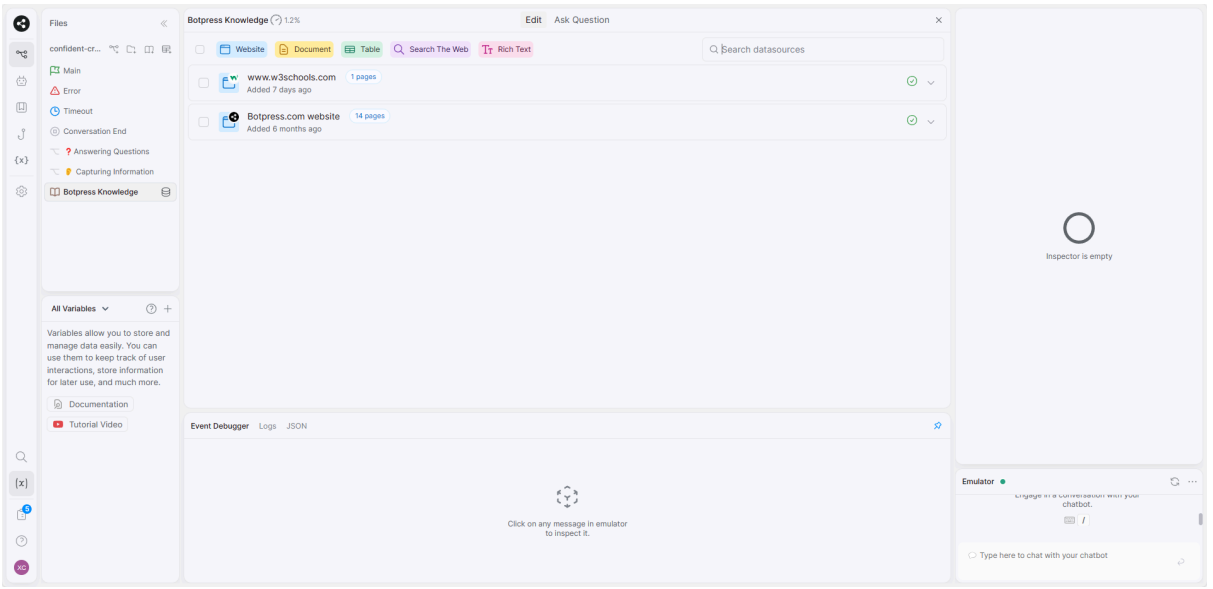
Here it is on the Amplify Studio. So the process is successfully done.

4. Chatbot deployment

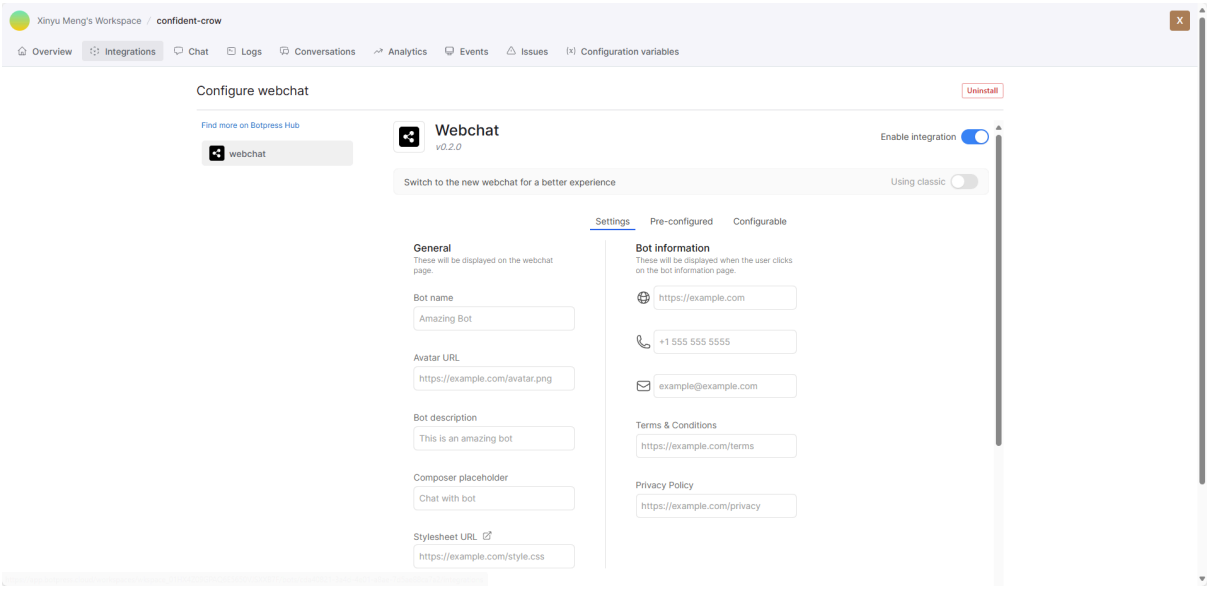
Botpress Cloud is a platform that provides hosting and management capabilities for chatbots developed using Botpress, an open-source conversational AI platform. The architecture of Botpress Cloud is designed to support the deployment, scalability, security, and management of chatbot applications.

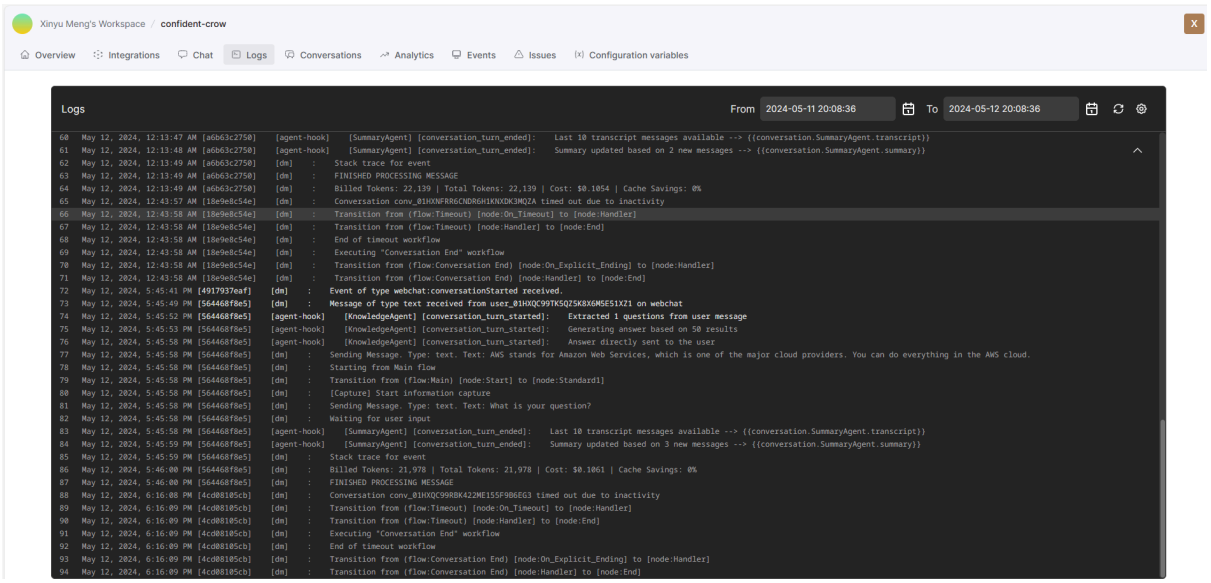
Our chatbot is developed in the studio of botpress cloud. Currently, the bot only has a handful of websites as its knowledge base. We plan to add a broader knowledge source to the robot in the future and maybe self-learning features.





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.

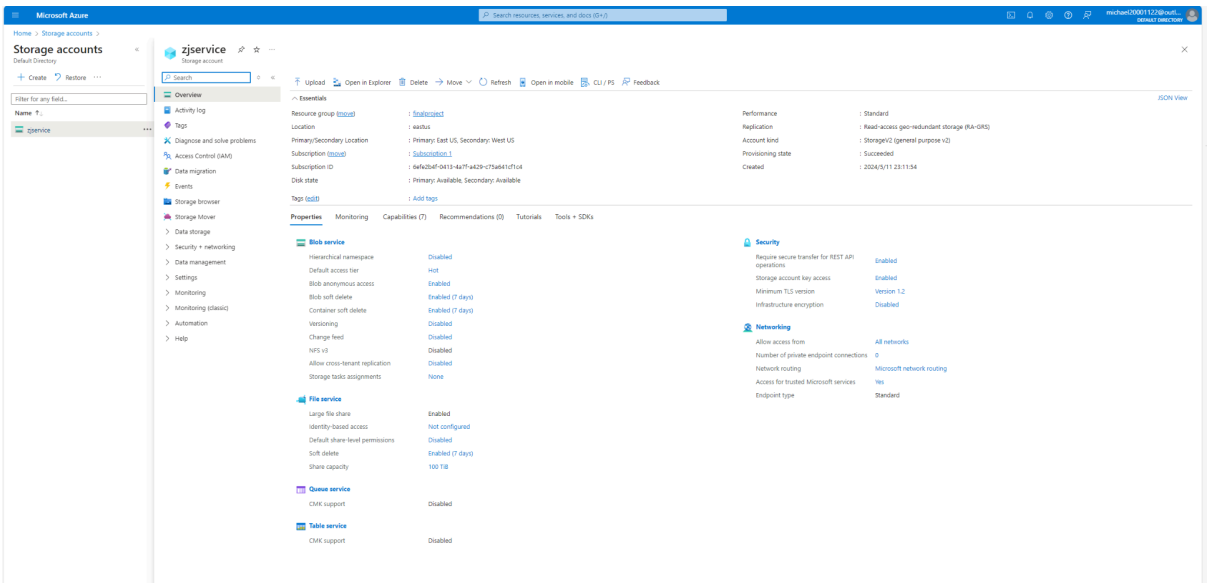




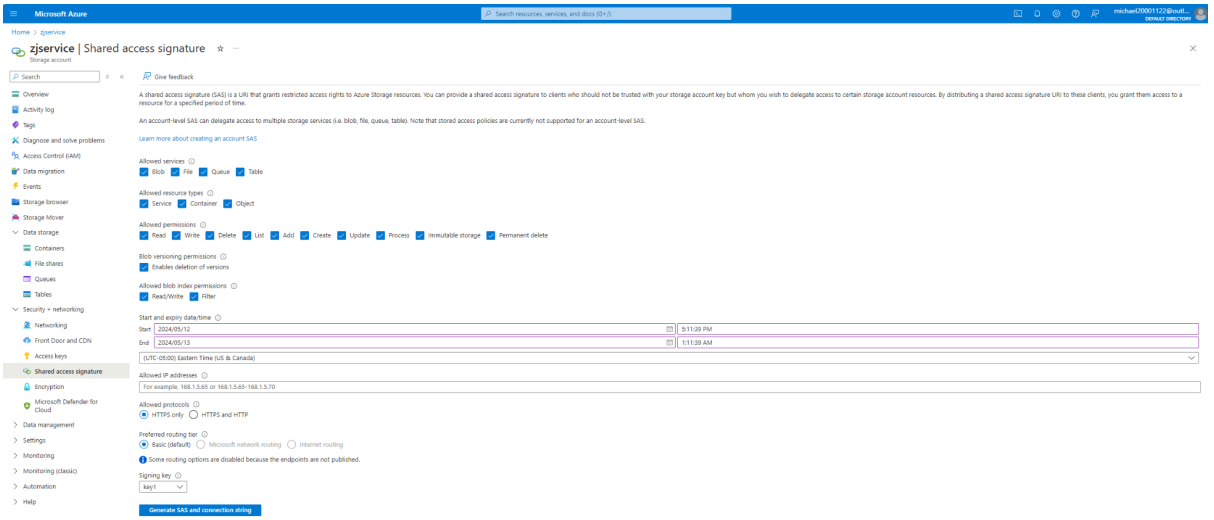
5. Azure Blob Storage initialize

   npm install @azure/storage-blob

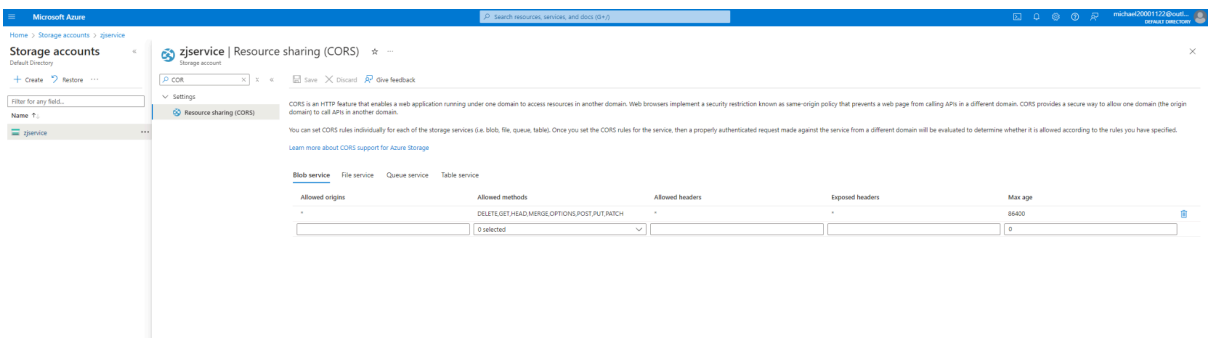
   And here is our storage built:



and generate SAS key from here:



Also manage CORS to allow all the methods:



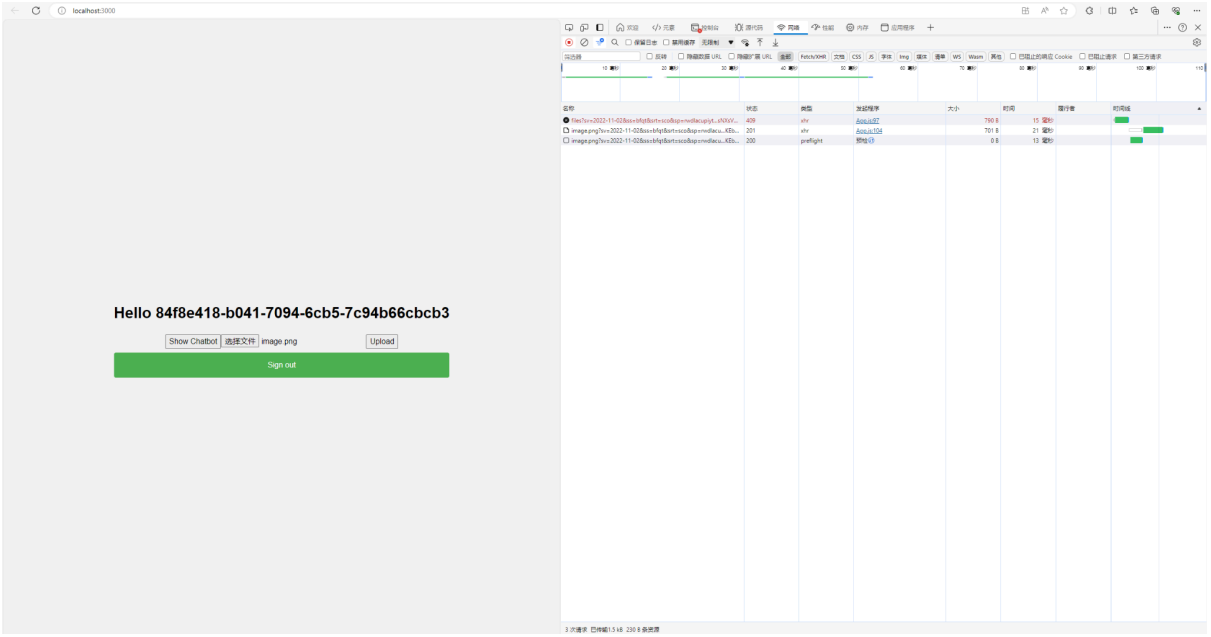
Finally by adding these two methods in React:

```
async function uploadToBlobStorage() {
  let accountName = "zjservice";
  let sas =
    "sv=2022-11-02&ss=bfqt&srt=sco&sp=rwdlacupiytfx&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.

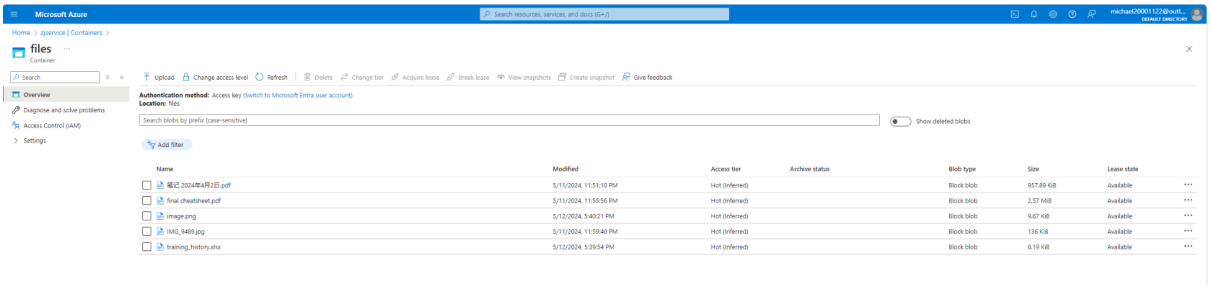
Let’s have a test:

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.



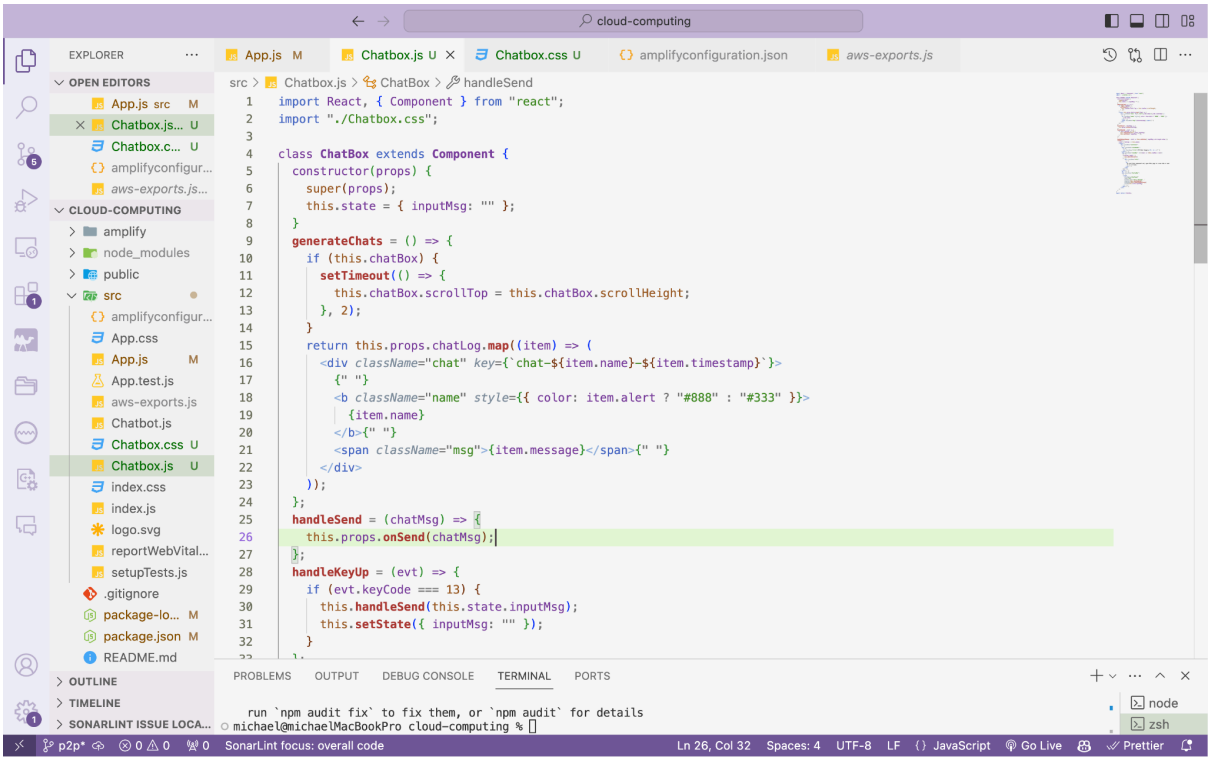
6. Adding P2P

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

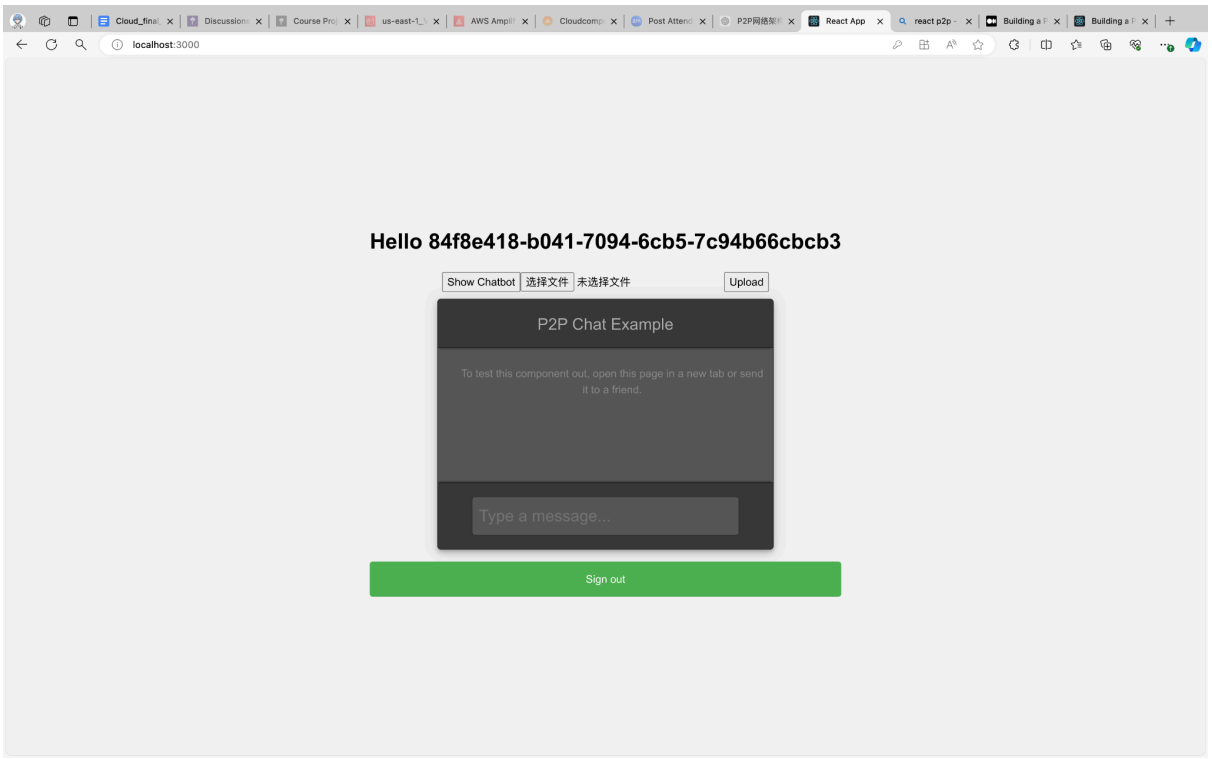
But we don’t use the LioWebRTC, cause it’s a very old version and won’t be compatible with our node version.

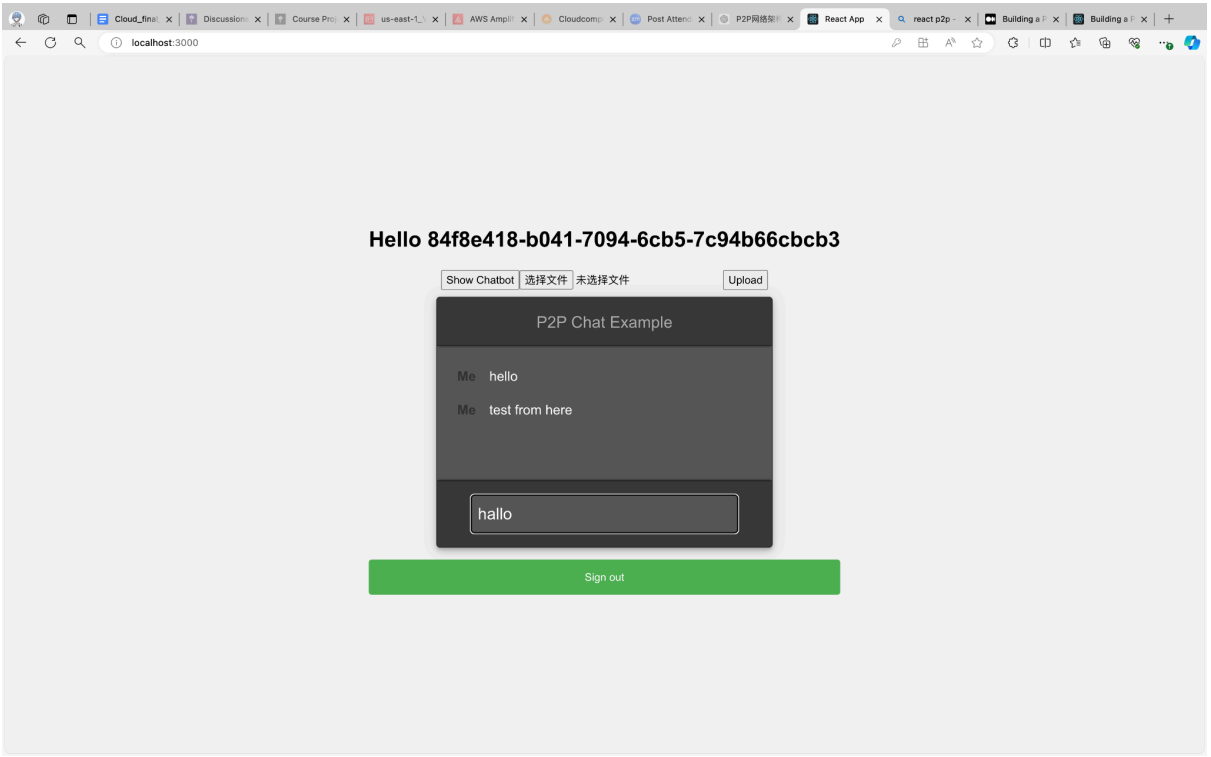
We decide to upload the chatLogs to Azure Blob Storage and fetch it down to show the chatLog history.

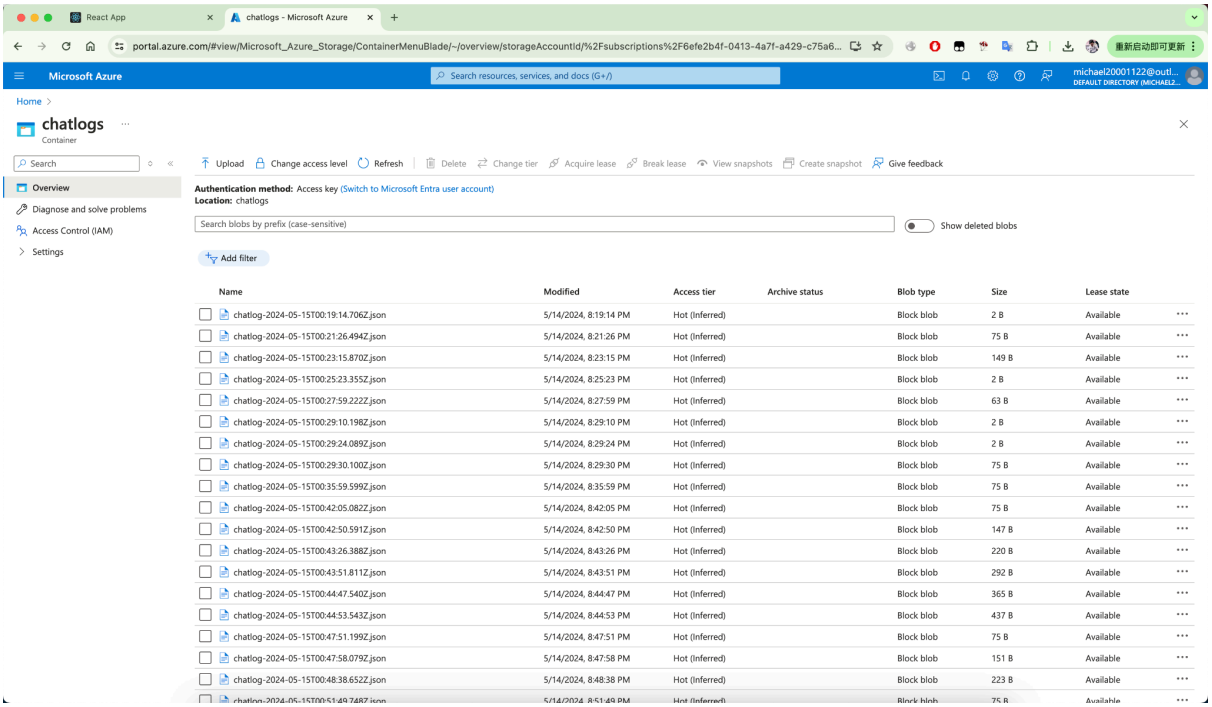
Before adding the P2P functionalities, here is the raw chatbox built.


cloud-computing
EXPLORER
App.js M
Chatbox.js U
Chatbox.css U
amplifyconfiguration.json
aws-exports.js
OPEN EDITORS
CLOUD-COMPUTING
amplify
node_modules
public
src
App.css
App.test.js
Chatbot.js
index.css
index.js
logo.svg
reportWebVital...
setupTests.js
.gitignore
package.json
README.md
OUTLINE
TIMELINE
SONARLINT ISSUE LOCA...
import React, { Component } from "react";
import "./Chatbox.css";
class ChatBox extends Component {
constructor(props) {
super(props);
this.state = { inputMsg: "" };
}
generateChats = () => {
if (this.chatBox) {
setTimeout(() => {
this.chatBox.scrollTop = this.chatBox.scrollHeight;
}, 2);
}
return this.props.chatLog.map((item) => (
<div className="chat" key={`chat-${item.name}-${item.timestamp}`}>
{" "}
<b className="name" style={{ color: item.alert ? "#888" : "#333" }}>
{item.name}
{" "}
{item.message}{" "}

));
};
handleSend = (chatMsg) => {
this.props.onSend(chatMsg);
};
handleKeyUp = (evt) => {
if (evt.keyCode === 13) {
this.handleSend(this.state.inputMsg);
this.setState({ inputMsg: "" });
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
run `npm audit fix` to fix them, or `npm audit` for details
michael@michaelMacBookPro cloud-computing %
node
zsh
p2p*
SonarLint focus: overall code
Ln 26, Col 32
Spaces: 4
UTF-8
LF
JavaScript
Go Live
Prettier


localhost:3000
Hello 84f8e418-b041-7094-6cb5-7c94b66cbcb3
Show Chatbot
选择文件
未选择文件
Upload
P2P Chat Example
To test this component out, open this page in a new tab or send it to a friend.
Type a message...
Sign out

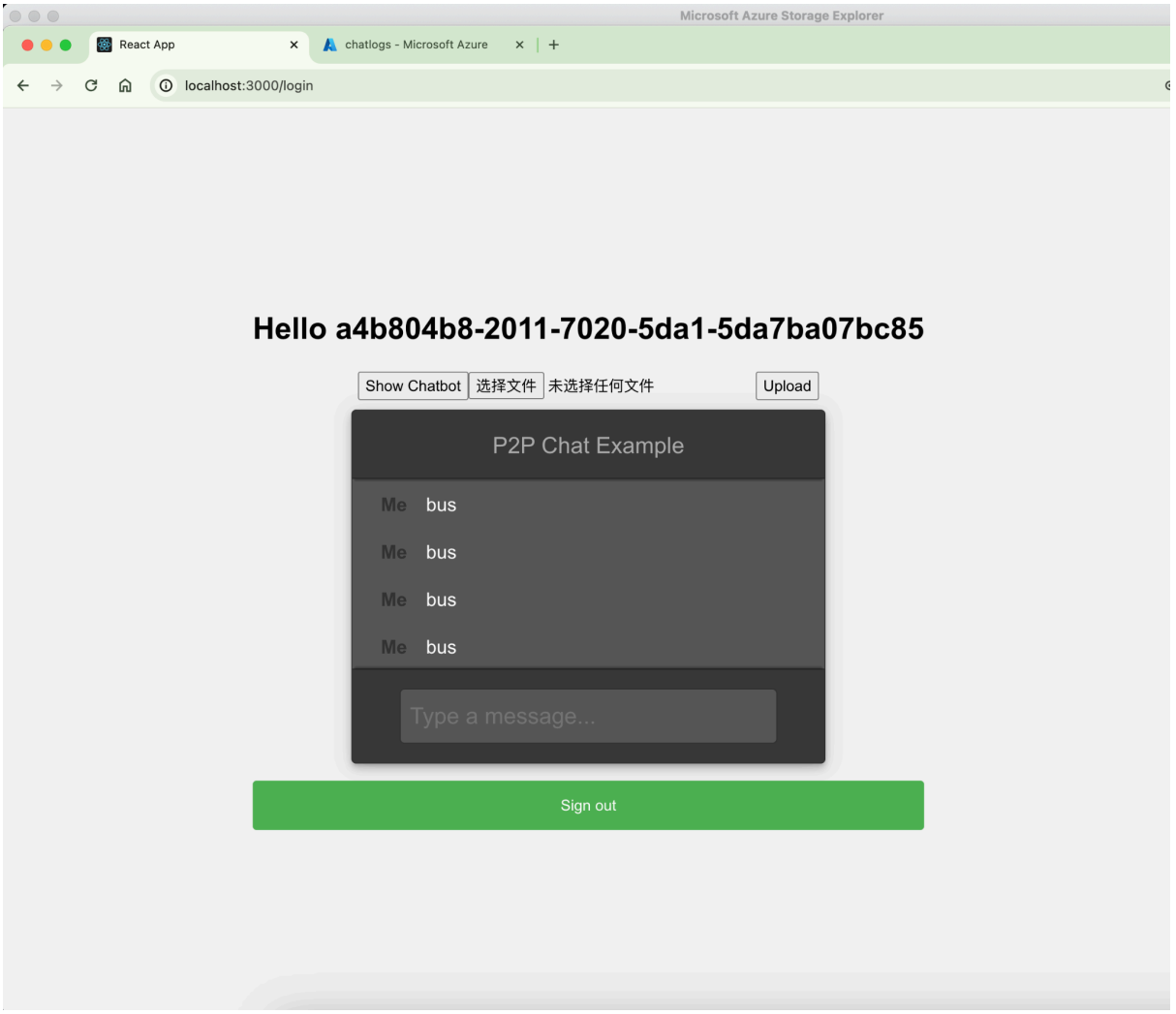


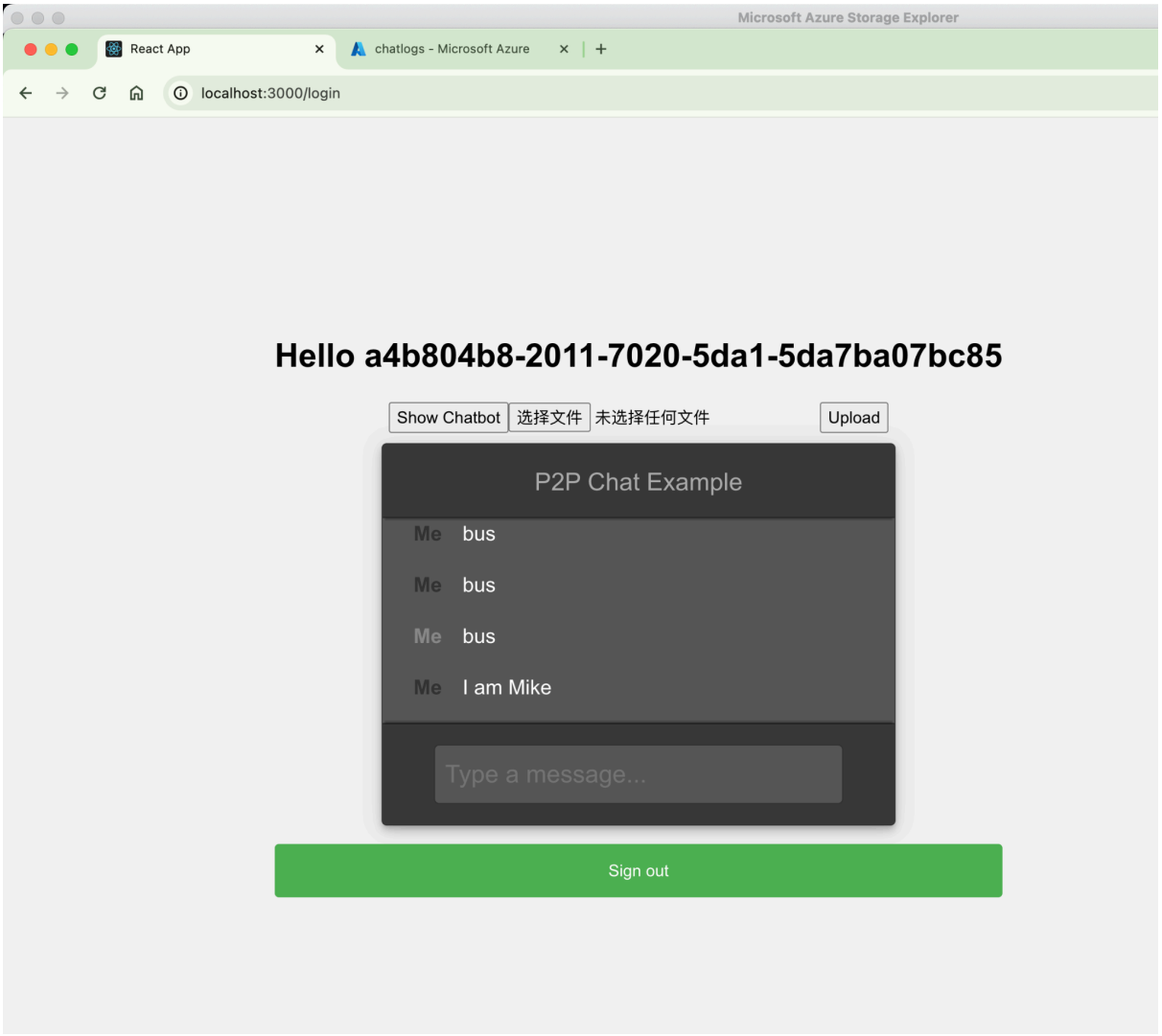


Used these two functions to Upload and Download the chatLogs

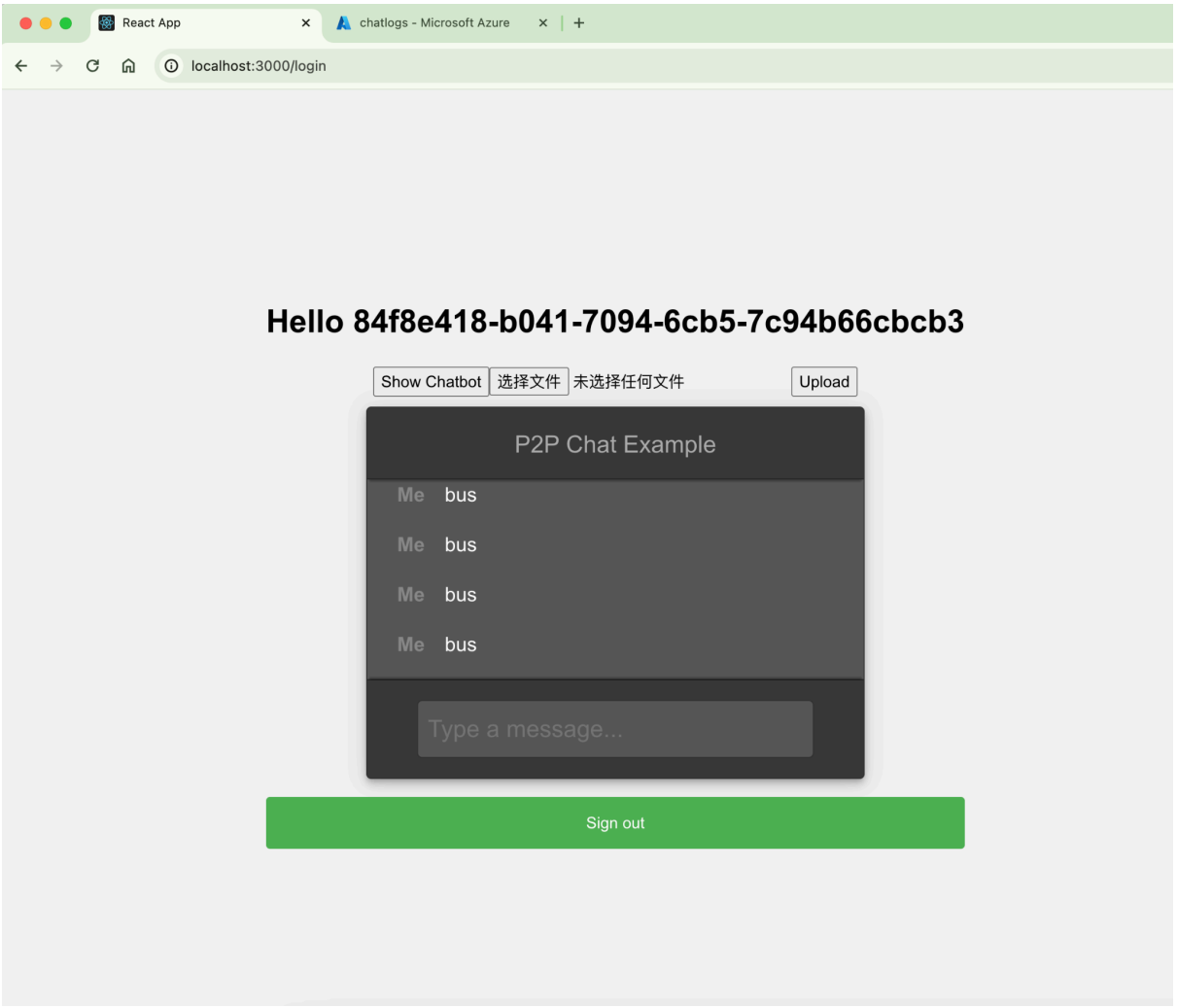
```
async function uploadChatLog(chatLog) {
  const accountName = "zjservice";
  const sas =
    "sv=2022-11-02&ss=bfqt&srt=sco&sp=rwdlacupiytfx&se=2024-05-15T08:16:18Z&st=2024-05-15T00:16:18Z&spr=https&sig=swX
  const blobServiceClient = new BlobServiceClient(
    `https://${accountName}.blob.core.windows.net?${sas}`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

  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);
}
```


Microsoft Azure Storage Explorer
React App
chatlogs - Microsoft Azure
localhost:3000/login
Hello a4b804b8-2011-7020-5da1-5da7ba07bc85
Show Chatbot
选择文件
未选择任何文件
Upload
P2P Chat Example
Me bus
Me bus
Me bus
Me bus
Type a message...
Sign out

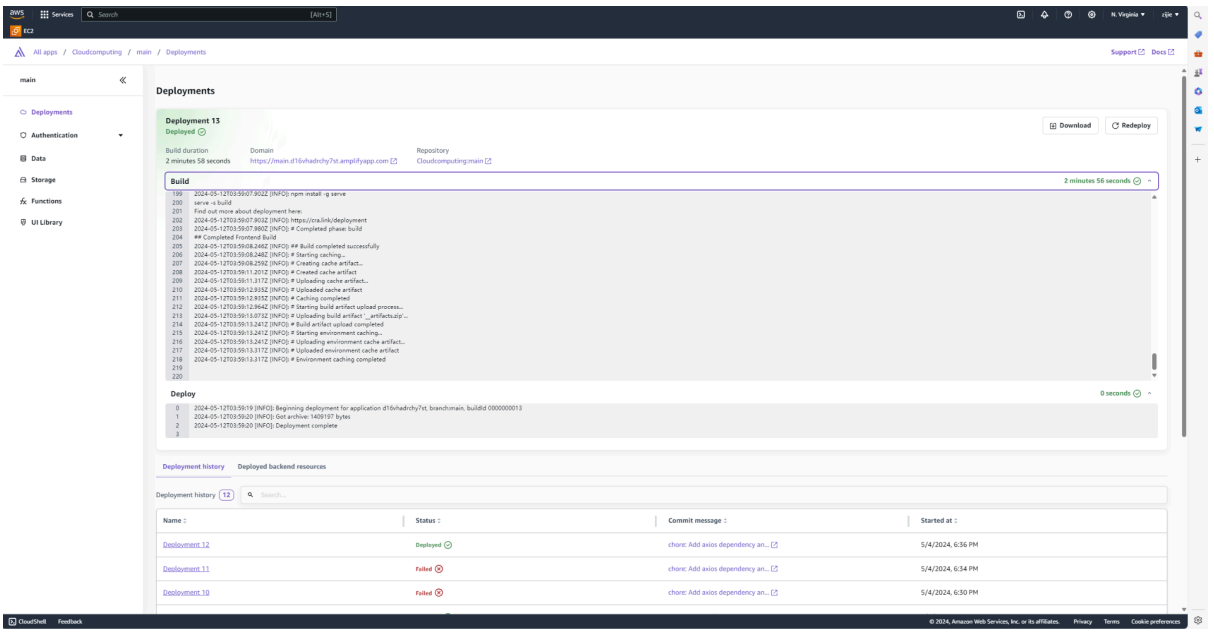


Therefore, two different accounts can chat in this chatbox. As whenever there's a change in the chatLogs, it will store to the Azure Blob and then fetch it down. Different color of "Me" means two different users.



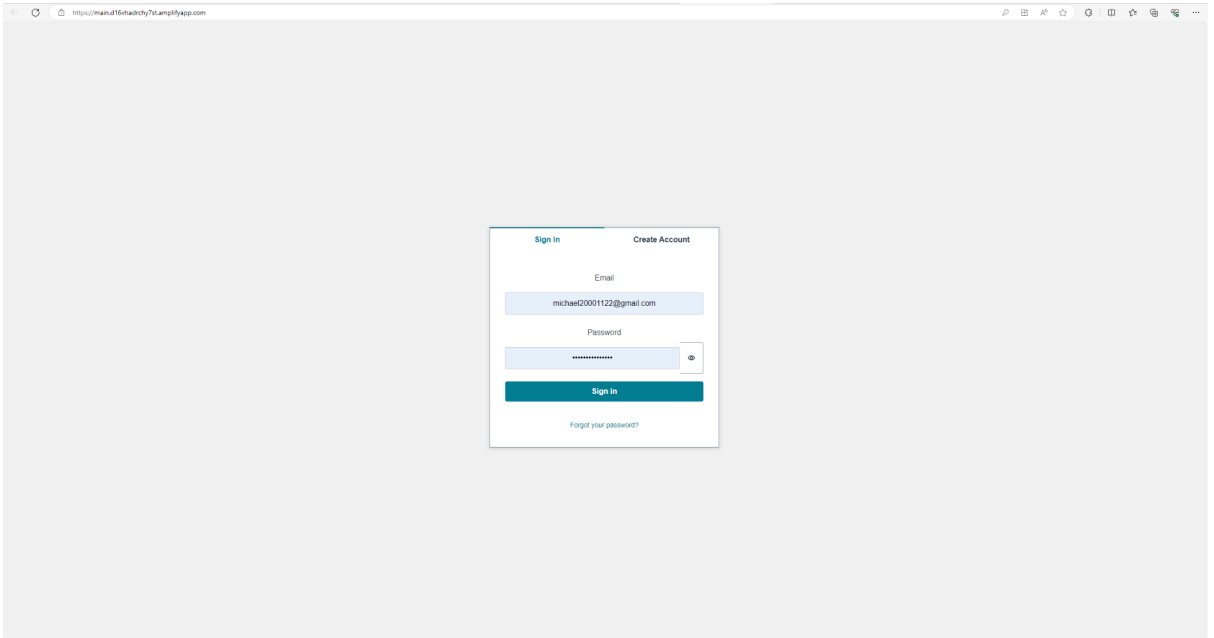
7. App deployment

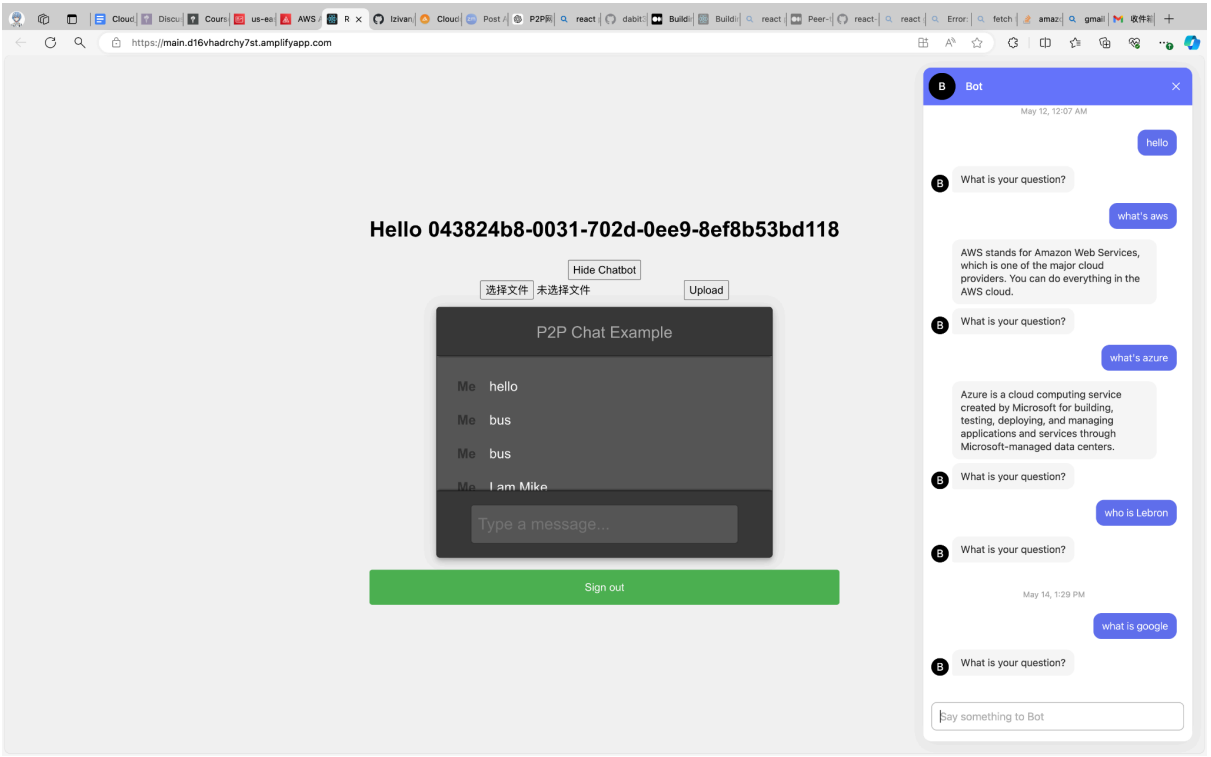
   We decided to deploy our React app to AWS amplify. So, we cloned our Github Repository to Amplify, and used the main branch for deployment.



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

And then it runs on: https://main.d16vhadrchy7st.amplifyapp.com/





All the functions are restored. Deployed successfully.