[See Me Please Recording Solution Project Documentation]



Postgraduate Capstone Project 5703

Project Name: See Me Please Recording Solution

Version: v3

Date: 2025-05-20

Team: CS14-2

Document Version: v1

**Index:**

**1.Introduction**

**2.Requirements**

**3.Architecture & Design**

**4.Installation & Deployment**

**5.User Guide**

**6.API Documentation**

**7.Testing**

**8.Operations & Monitoring**

**9.FAQ & Troubleshooting**

**10.Changelog**

**11.License**

**12.Appendix**

**1. Introduction**

**- Background:** *See Me Please* is dedicated to improve the accessibility of digital products. By having real users record their interactions, *See Me Please* evaluates the effectiveness and design quality of products. However, manually collecting and managing large volumes of these recordings is labor-intensive.

**- Project Overview:** To streamline this process and support scalable user testing, we are developing a comprehensive Recording Tool alongside a Video Analysis Tool, so that collected data could be analyzed automatically and provide visualized analysis results to developers. The Recording Tool provides a guided, multi-task workflow—capturing camera input, microphone audio, and screen activity—to automate and simplify the recording phase.

This document focuses on how to install, configure, and use the Recording Tool to efficiently capture user sessions and prepare recordings for analysis.

**2. Requirements**

**- Frontend:** Vue.js 3.2+, Vue Router 4.5+, axios 1.8+, uuid 11.1+, vue-meta 2.4+

**- Backend:** Python 3.9+, Flask 3.1+, Flask-CORS 5.0+, pytest, moto, boto3 1.37+

**- Cloud Storage:** AWS S3 with valid IAM credentials

**- Build Tools:** npm 10.8+

**- Browser Support:** Modern browsers with MediaRecorder API (Chrome, Edge, Firefox, etc)

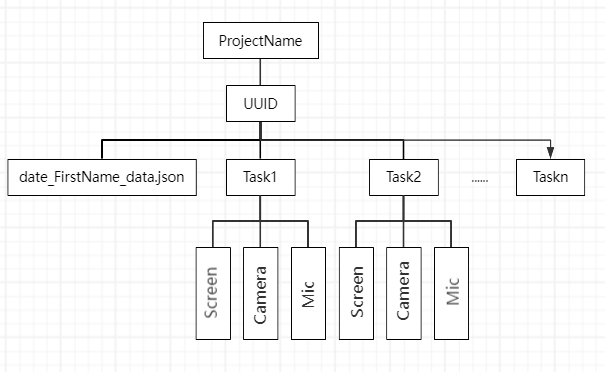
**3. Architecture & Design**

**3.1 Layered Architecture:**

**- Business Logic Layer:** Recording service, CMS service, visualization service, and analysis service(**not implemented yet**).

**- Backend Layer：**Provide recording(**api/recording/\***), CMS(**api/projectTask/\***), visualization(**api/visualization/\***), and analysis(**not implemented yet**) endpoints for frontend request

**- Storage Layer:** AWS S3 integration via backend(bucket name: ***cs14-2-recordingtool***)



*Figure.1. The store structure for recording results*

**3.2 Key Components & Modules**

**Frontend:**

HomePage (parent wizard component with stepper)

PageOne (User Info Form and generate uuid)

PageTwo (Camera & Microphone Permissions)

PageThree (Screen Sharing Permissions)

PageFour (Recording Controller & Popup window)

AboutPage (CMS system management)

OutputPage (Visualization)

**Backend:**

app (http api list)

s3\_client (the certificate for s3)

api\_test (http api test case)

**4. Installation & Deployment**

**Clone Repository:**

*git clone https://github.com/CrashDream-1219/cs14-2.git*

*cd video-recording-tool*

**Install Frontend Dependencies:**

*cd frontend*

*npm install*

*npm install vue@^3.2 vue-router@^4.5 axios@^1.8 uuid@^11.1 vue-meta@^2.4*

**Install Backend Dependencies:**

*cd ../backend*

*pip install -r requirements.txt*

**Run:**

# Backend Locally

*python -m venv venv*

*source venv/bin/activate* (# macOS/Linux)

*.\venv\Scripts\Activate* (# Windows)

*flask run --host=0.0.0.0 --port=5000*  (#port number could be modify)

# Backend deployment on EC2

cd /tmp/pycharm\_project\_114

*source venv/bin/activate*

*flask run --host=0.0.0.0 --port=5000*  (#port number could be modify)

# Frontend

*npm run serve*

**Production Build:**

*npm run build*

**Deploy:**

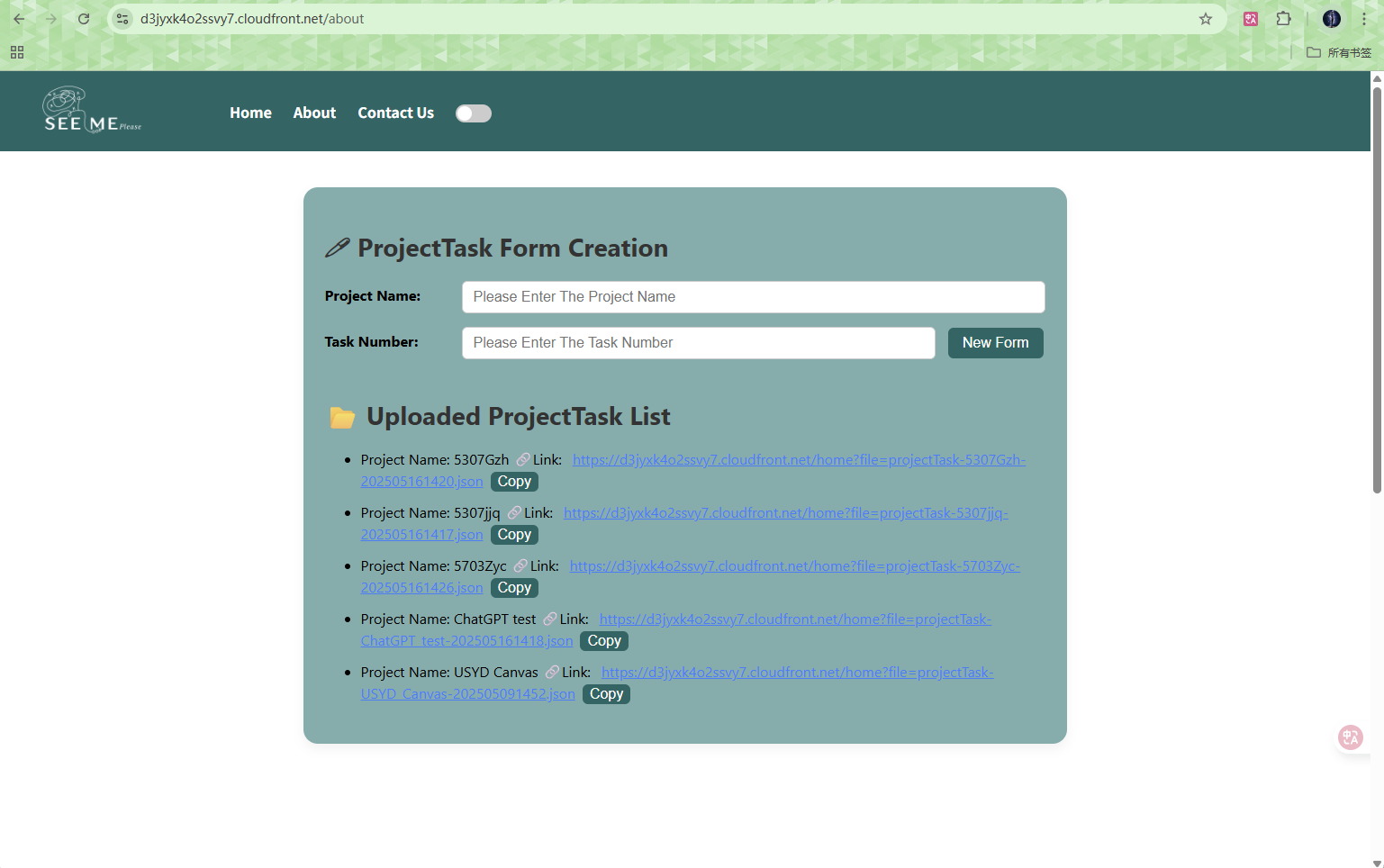
Host frontend on S3 + CloudFront or Netlify

Deploy Flask backend on AWS Elastic Beanstalk, ECS, or EC2

**5. User Guide**

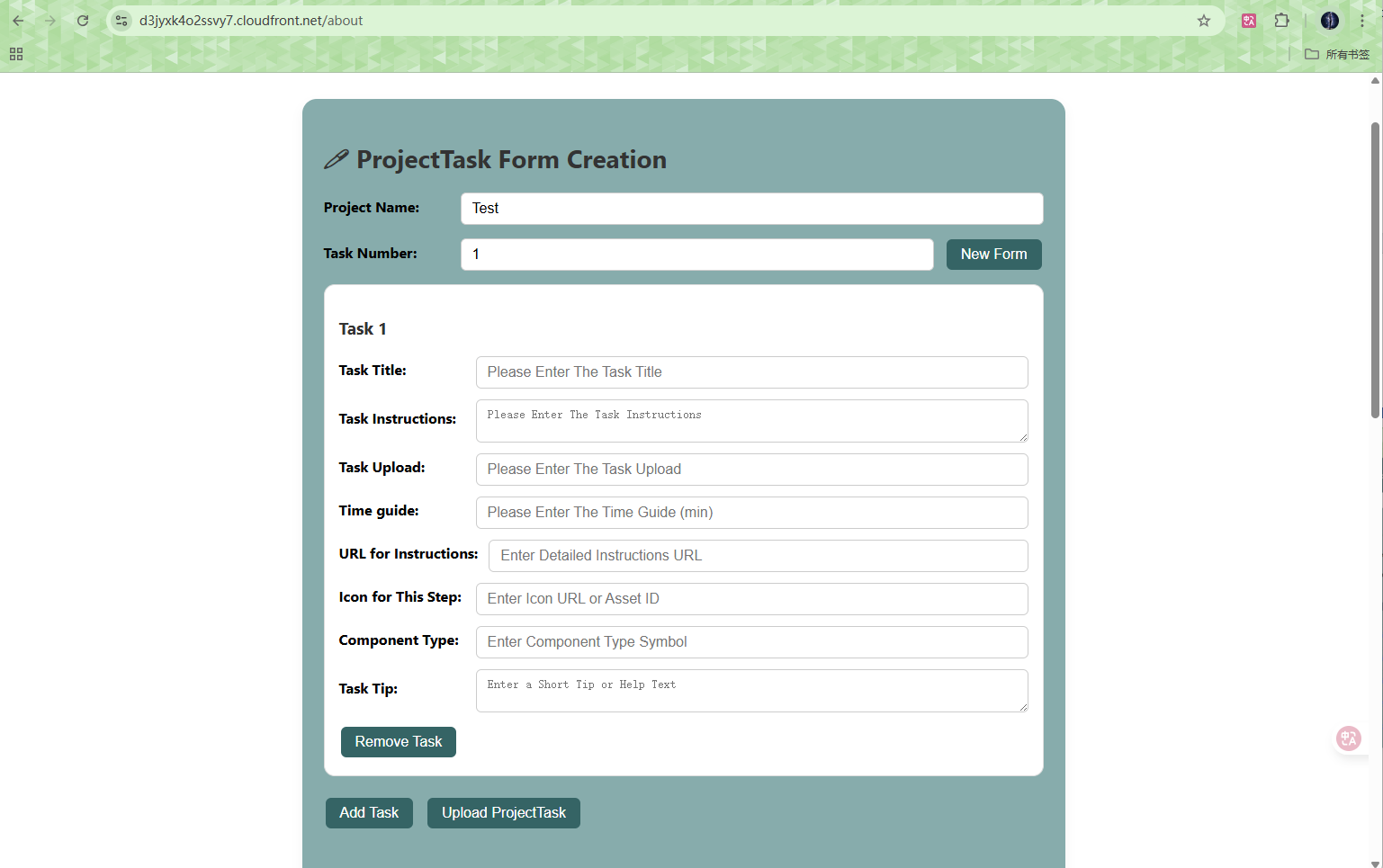
**For CMS system:**

1.Navigate to the app URL(Deployment: *<https://d3jyxk4o2ssvy7.cloudfront.net>* or local: *<http://localhost:8080>*) in a supported browser. Manager could create new projectTask and the existing projectTask will be listed below.



*Figure.2. The CMS management page*

*2.*Input project name and task number, then click ‘New Form’ to create a new projectTask. User could add and remove the subtask, but the task number should be greater than 1. After type all fields, click ‘Upload ProjectTask’ to upload the new projectTask to AWS S3.

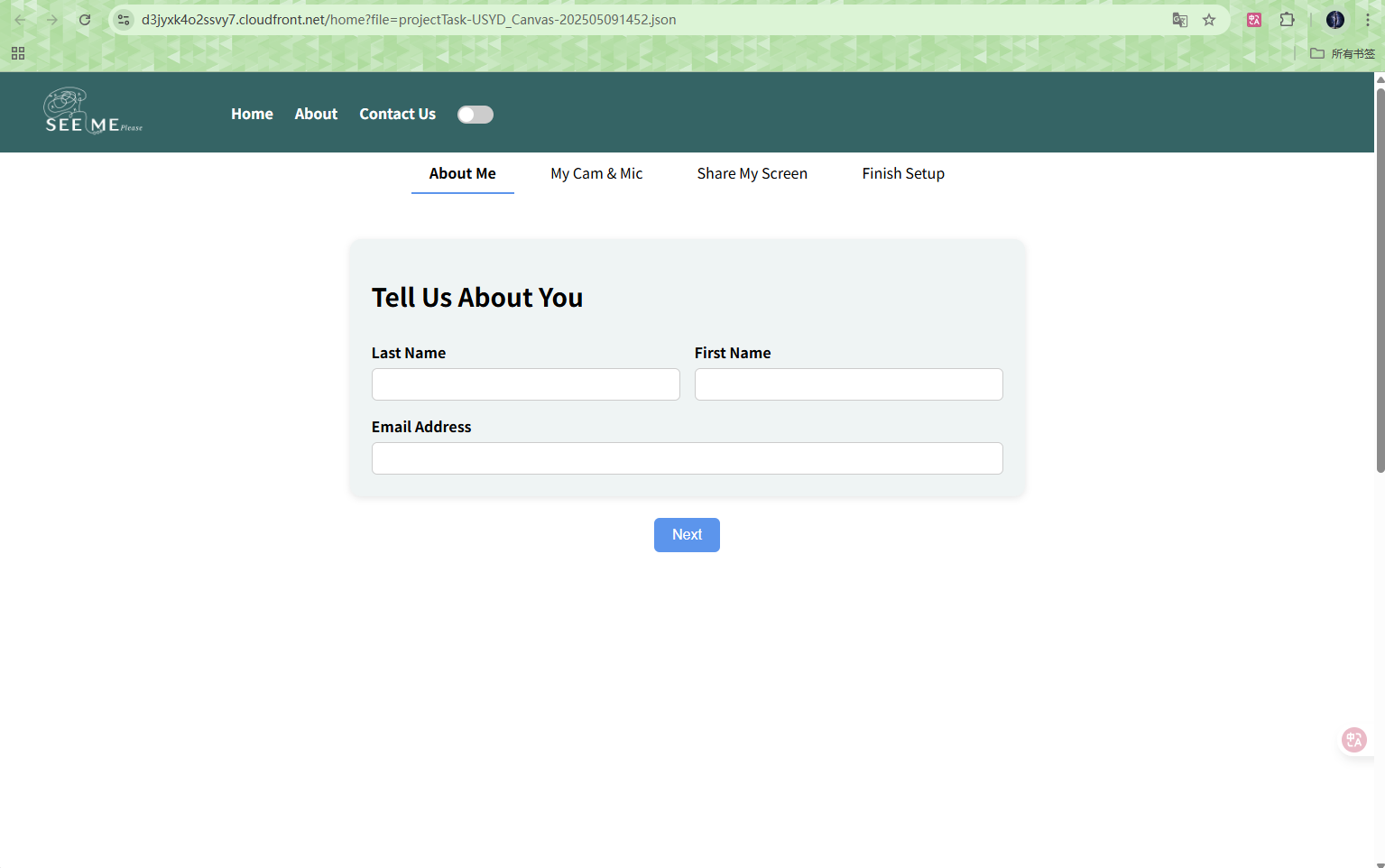


*Figure.3. The projectTask creation form*

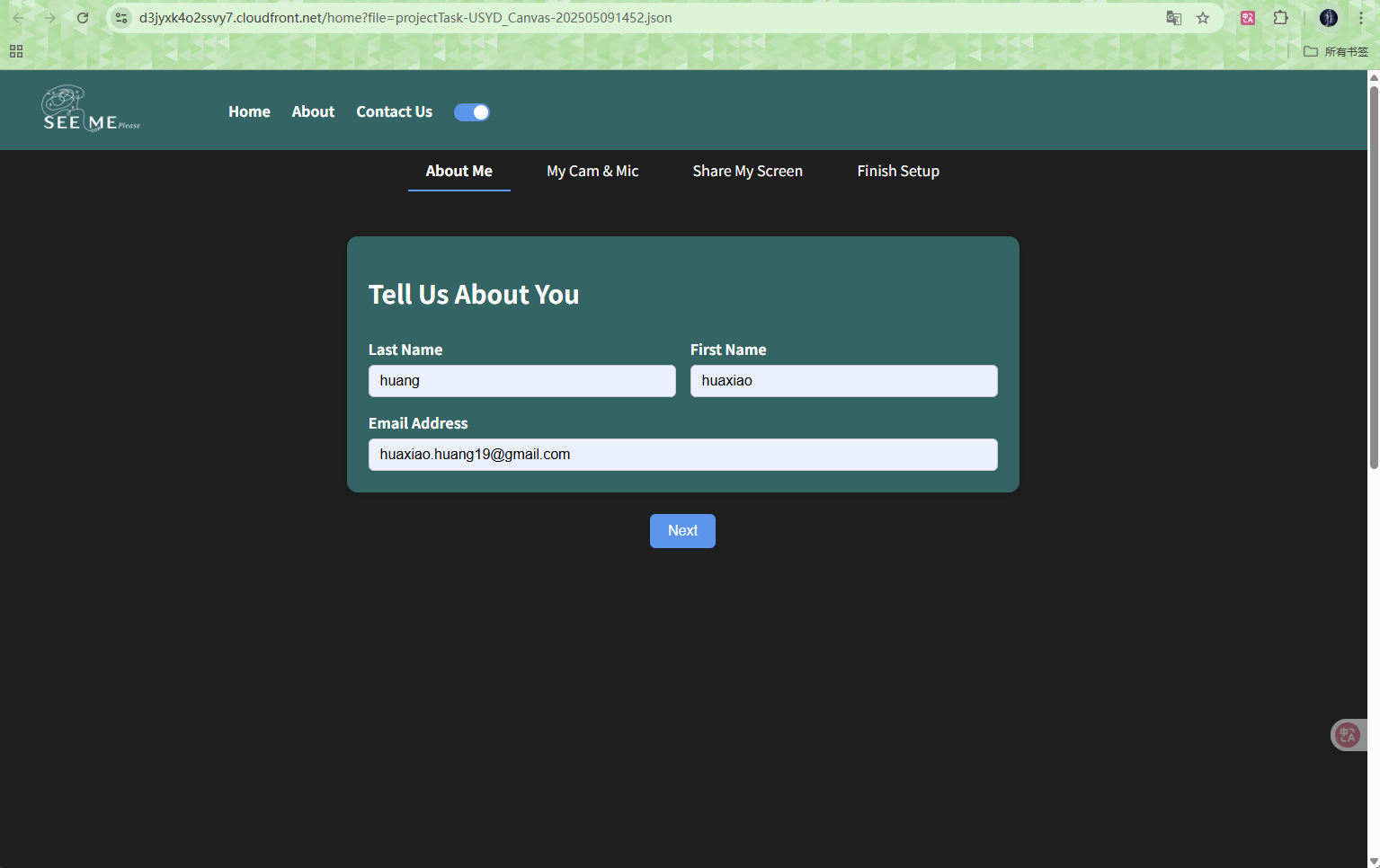
3.Then current page will refreash the projectTask list, and it will generate related link for tester.

**For project tester:**

1.Send the testing app URL(Deployment: *<https://d3jyxk4o2ssvy7.cloudfront.net>/home?file=projectTask-USYD\_Canvas-202505091452.json* or local: *<http://localhost:8080>/home?file=projectTask-USYD\_Canvas-202505091452.json*) and open in a supported browser (The URL is not unique, user could send required URL to tester). The top bar could change the theme to dark mode as shown in Figure.5.

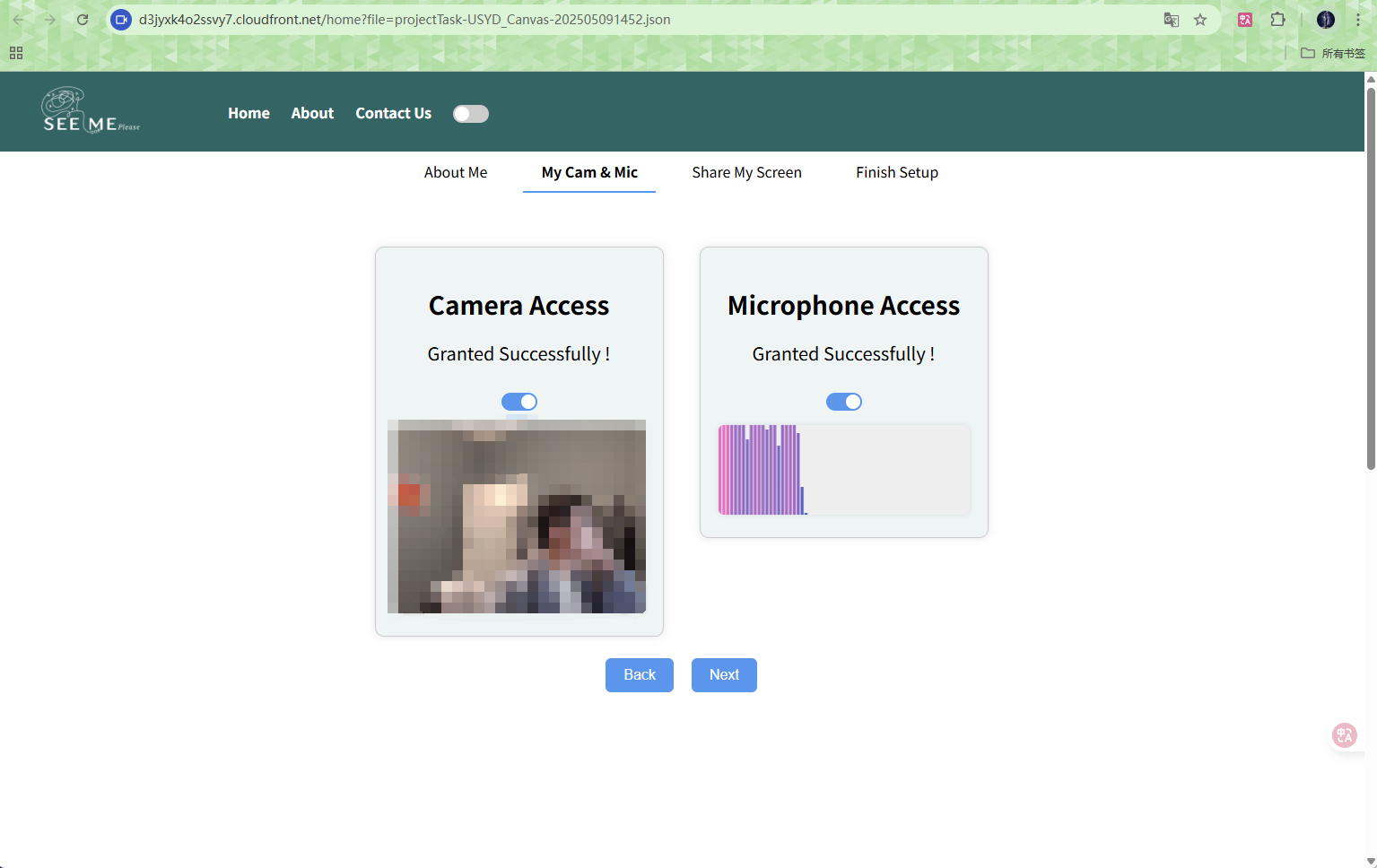


*Figure.4. The recording tool page one*

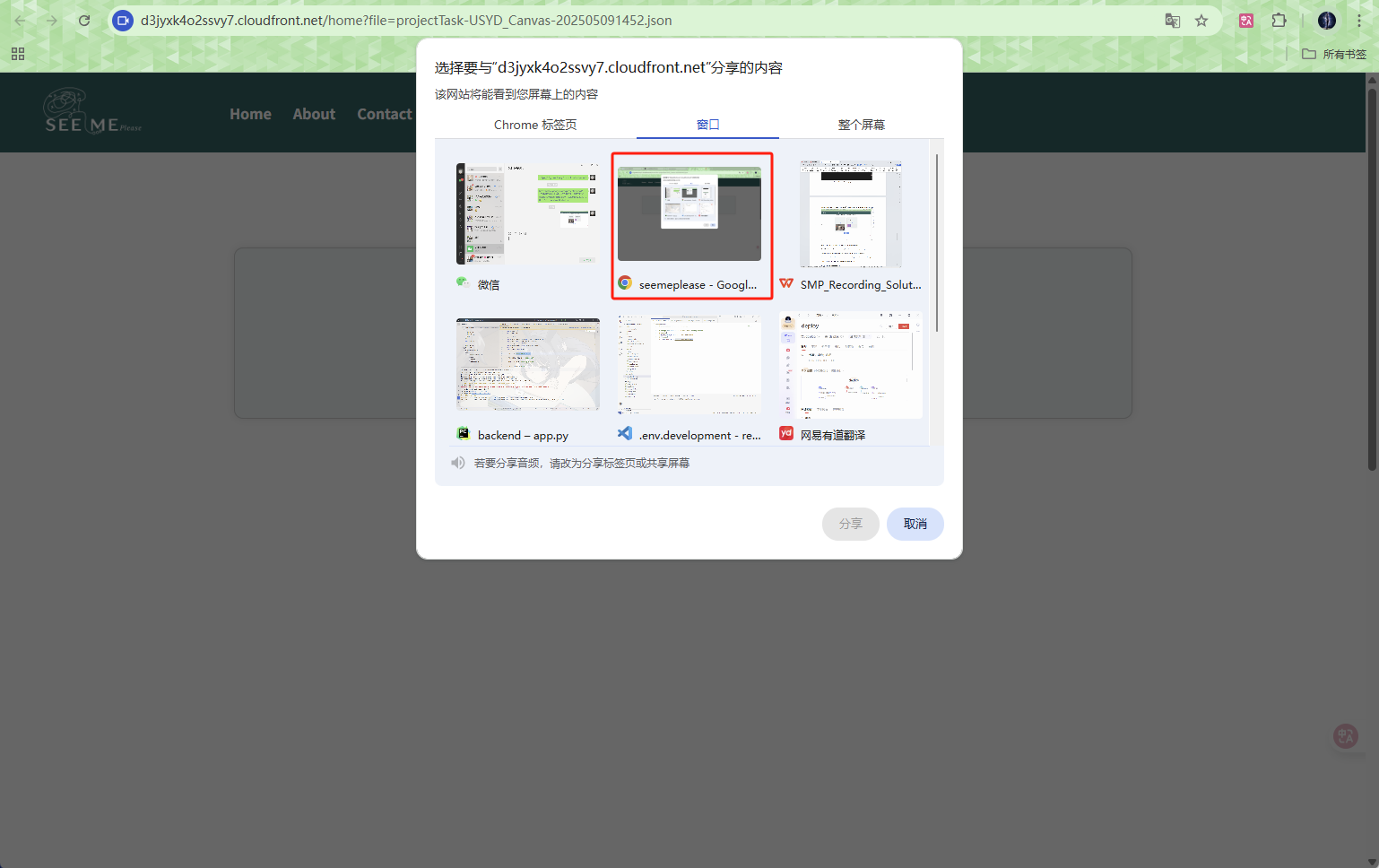


*Figure.5. The recording tool page one with dark mode*

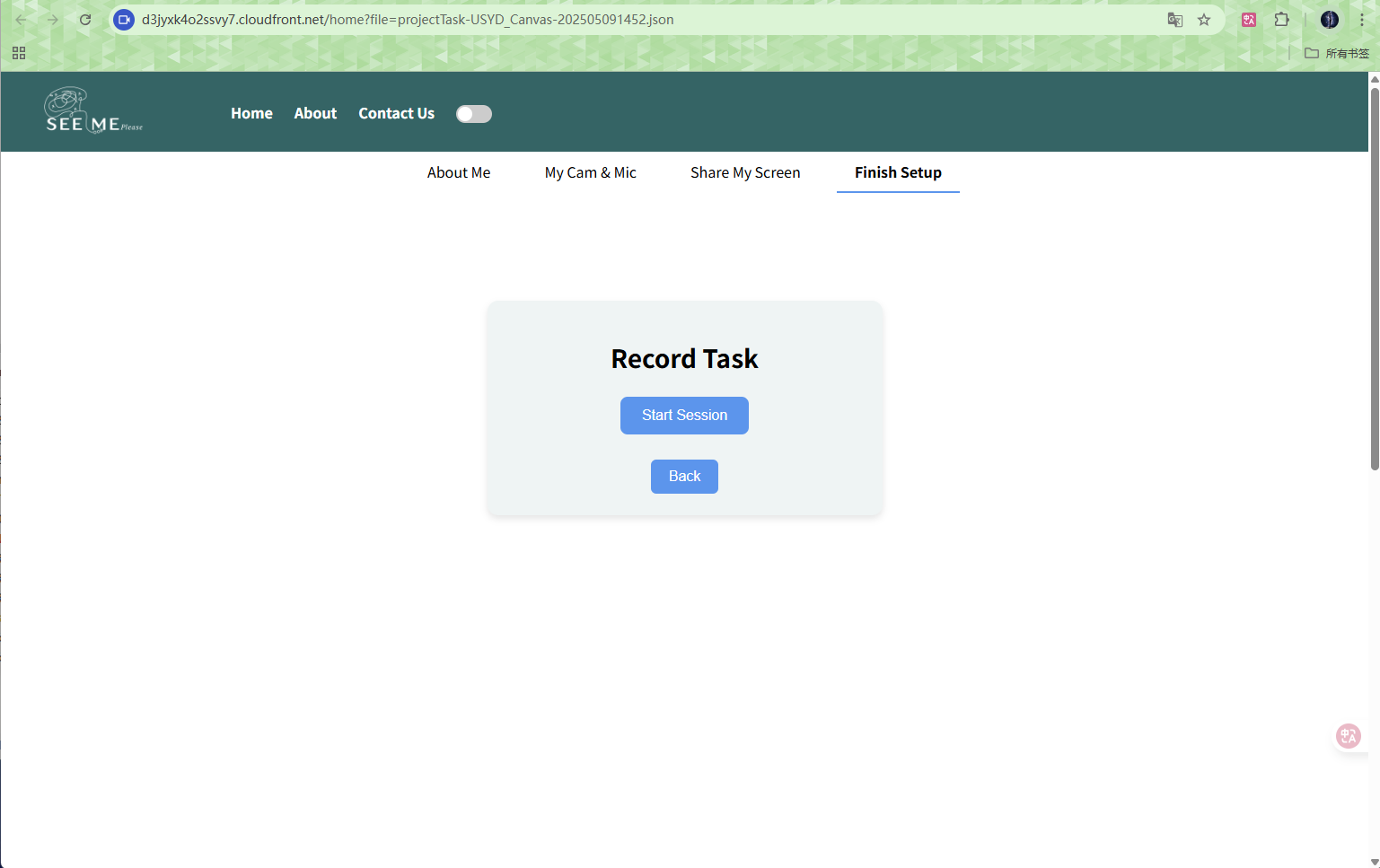
2.Input tester information and allow camera, microphone, then screen sharing as prompted. Click the ‘start session’ to open the popup window and start recording.

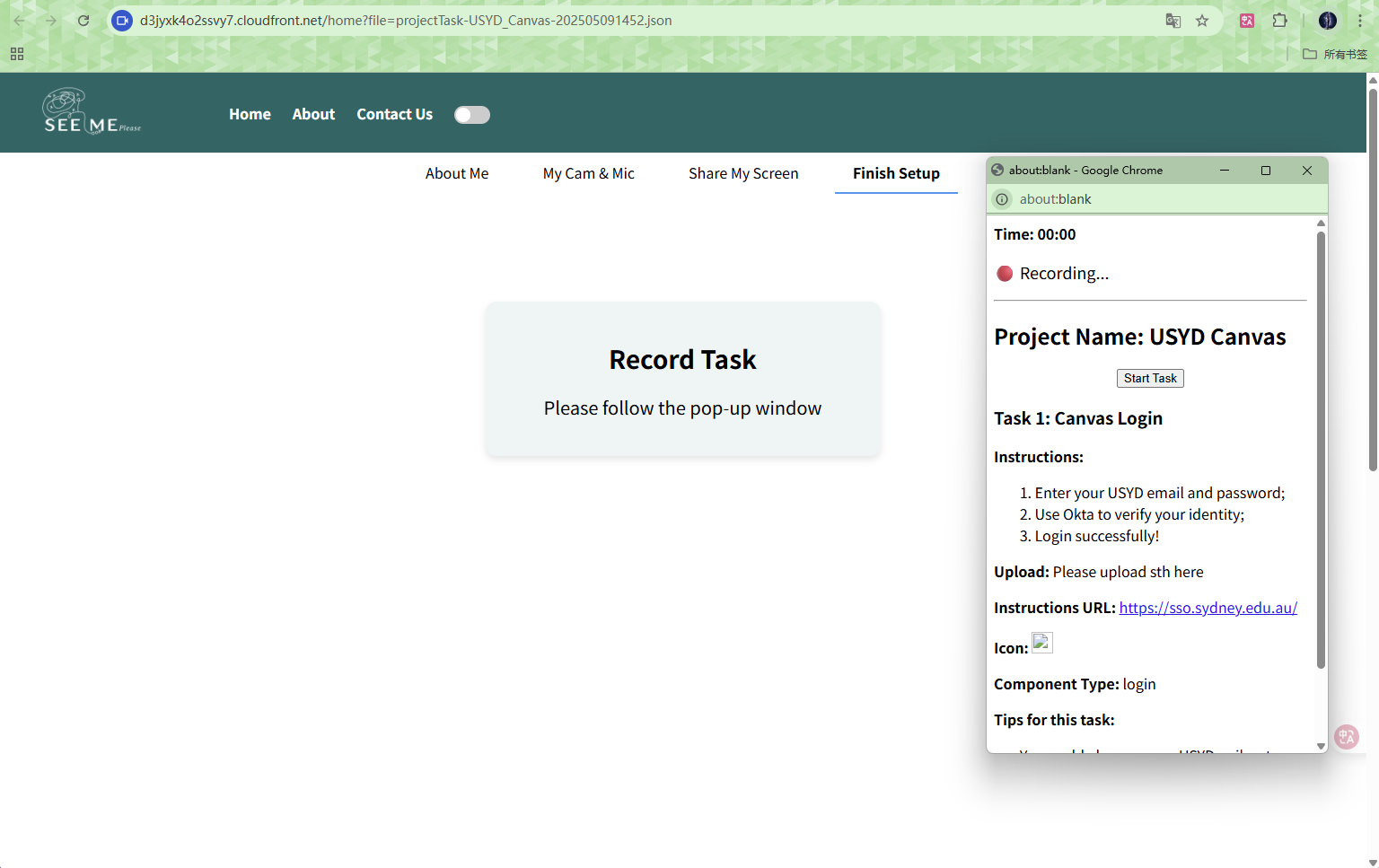


*Figure.6. The recording tool page two to allow camera and microphone*



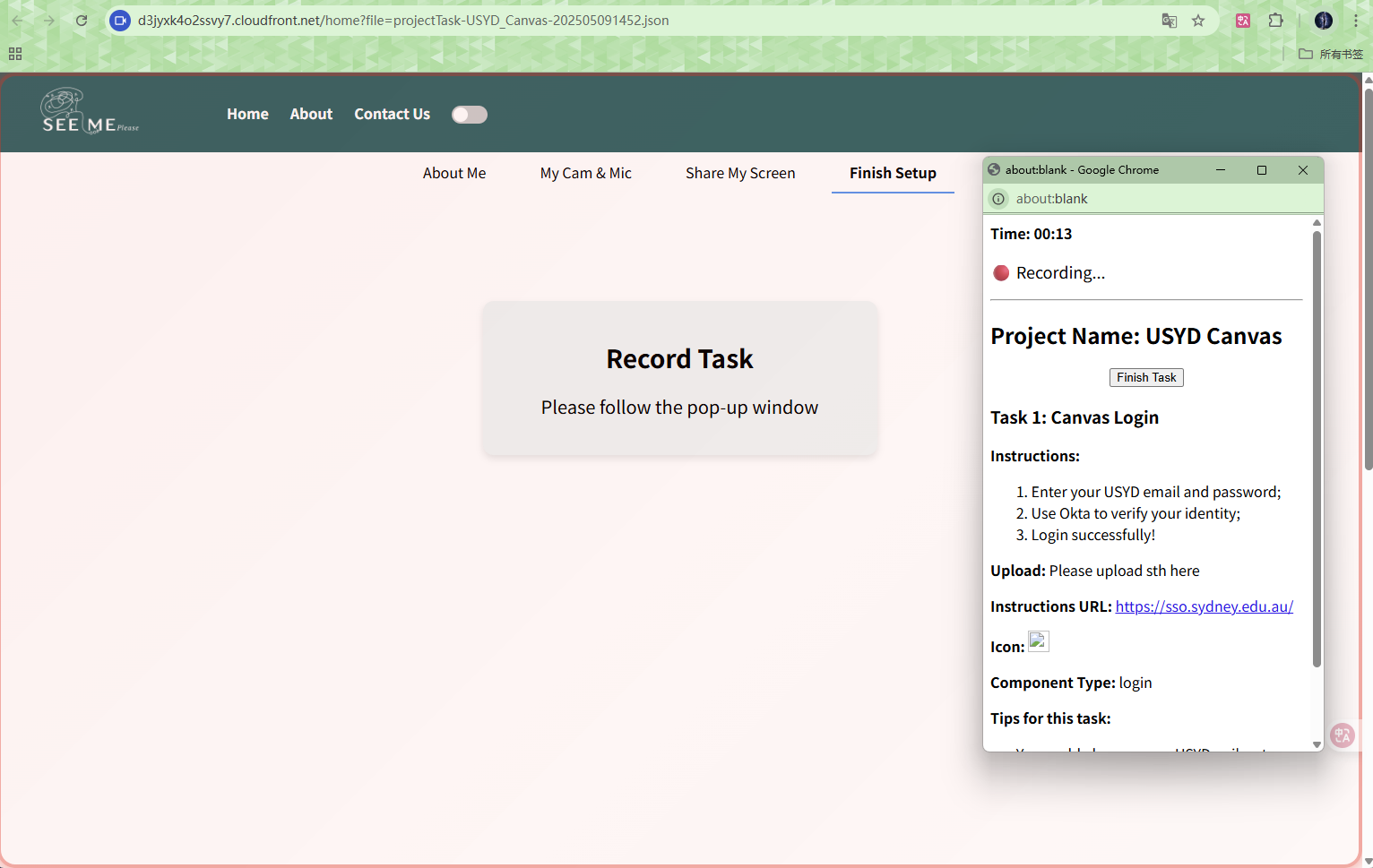
*Figure.7. The recording tool page three to allow screen shaing(choose the second one ‘window’ and select the current browser)*



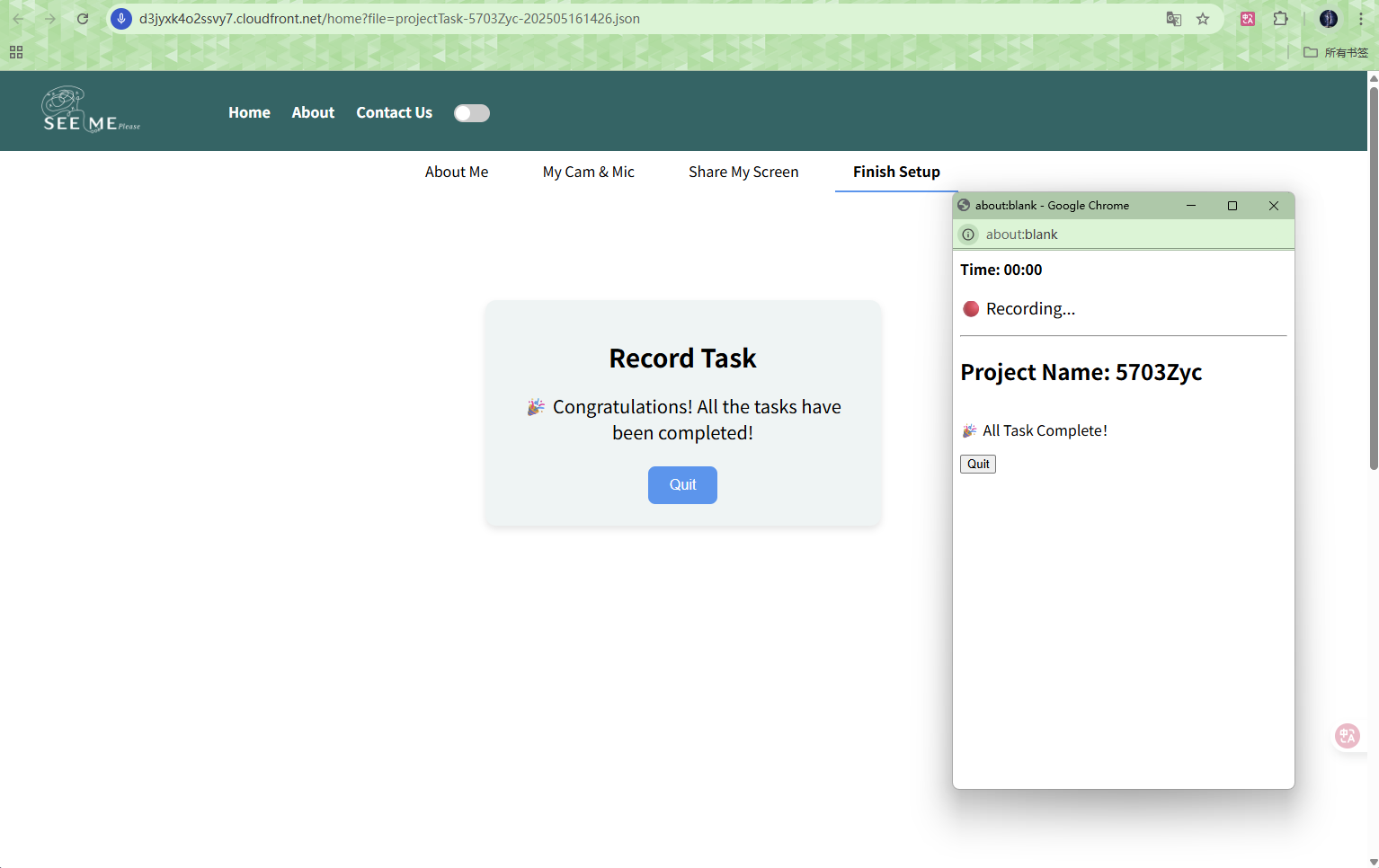


*Figure.8. The recording tool page four and the popup window*

3.Click the ‘start task’ to start the recording, the red frame and timer could remind tester that the recording is starting. Click ‘finish task’ to complete the recording of the current task. After all tasks complete, tester could click ‘quit’ to close the recording tool.



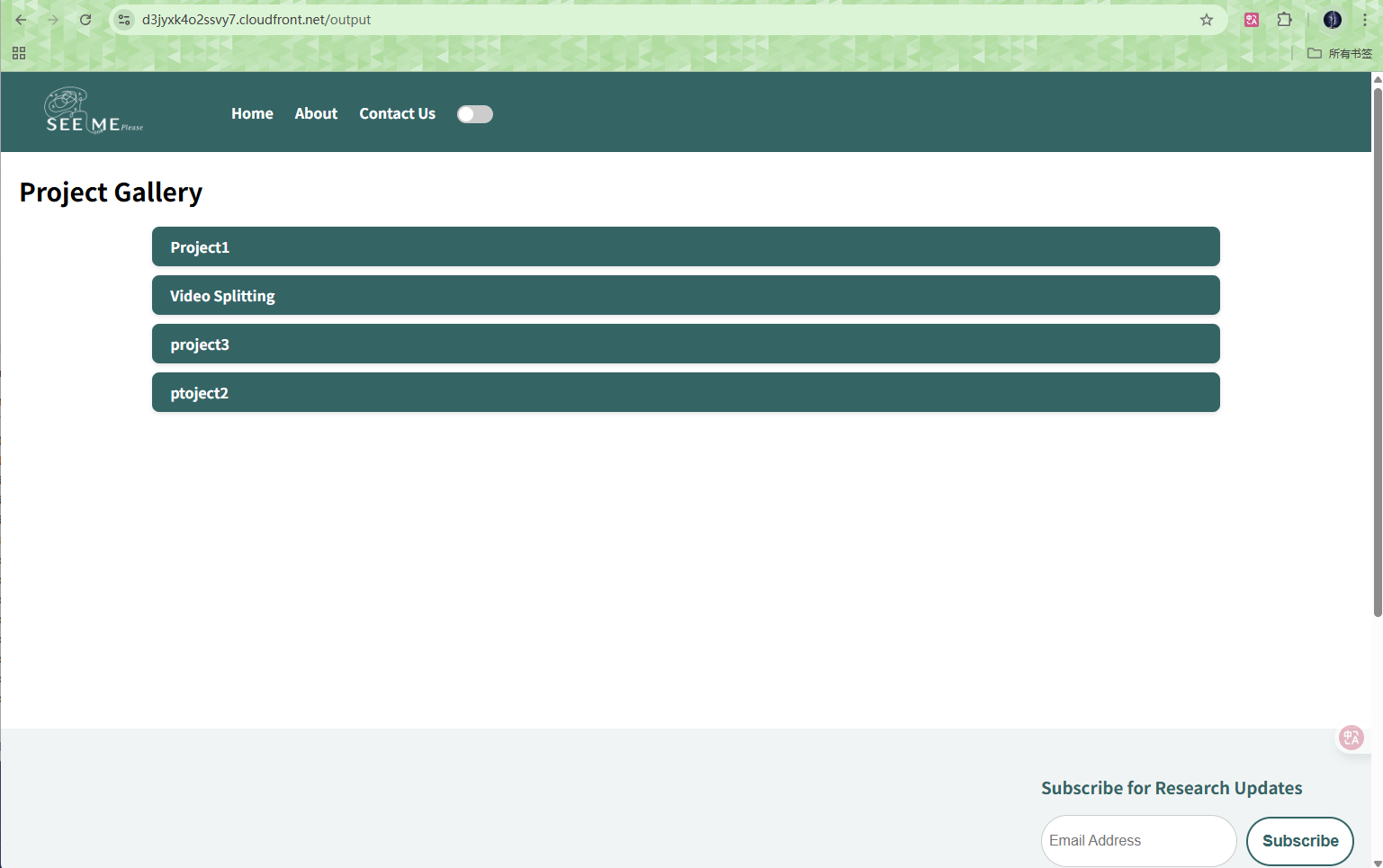
*Figure.9. The recording tool page four and the popup window with recording status*



*Figure.9. The recording tool page four and the popup window with finish status*

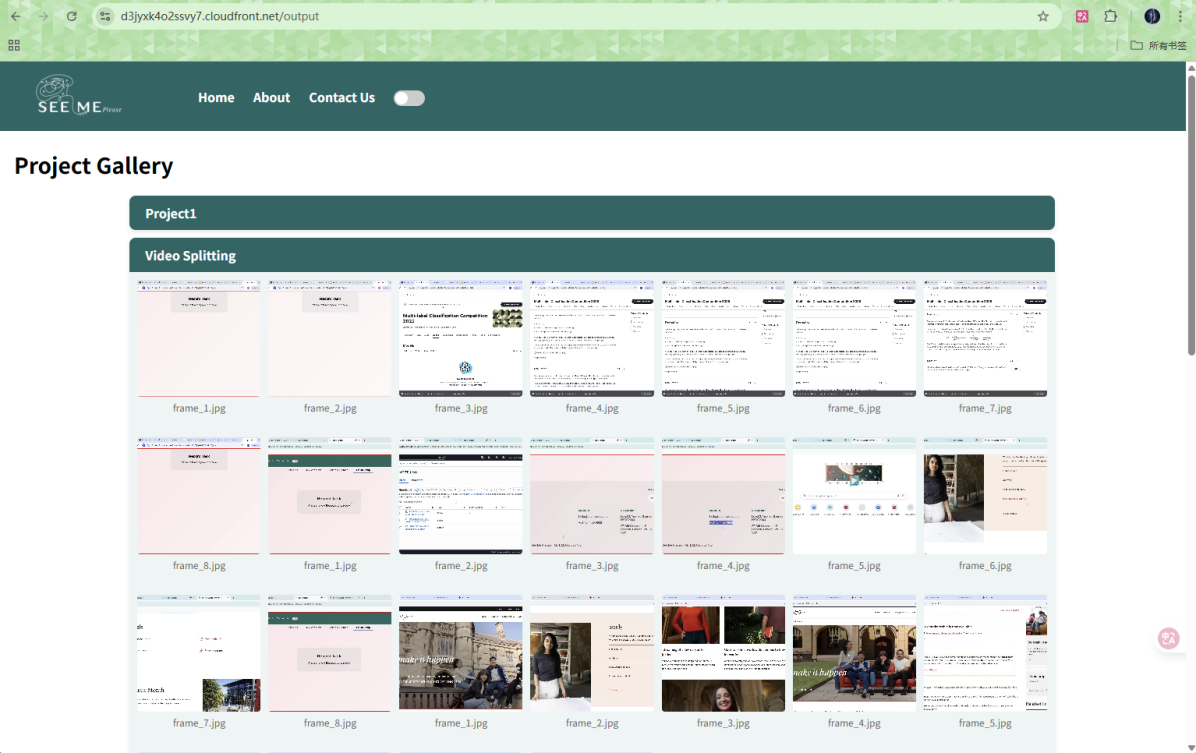
**For manager visualization:** (Due to the time limited, it is a only simple sample)

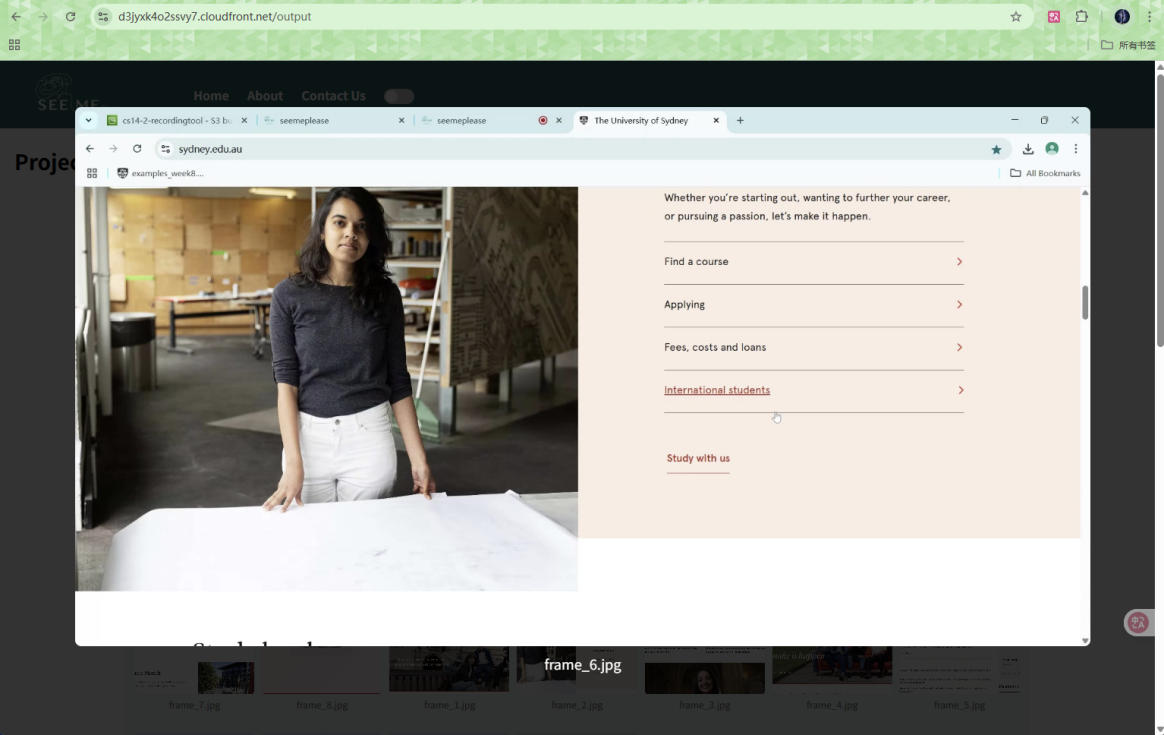
1.Navigate to the app URL(Deployment: *<https://d3jyxk4o2ssvy7.cloudfront.net>* or local: *<http://localhost:8080>*) in a supported browser. Click the ‘Contact us’ on the top bar to swith to the visualization page. All the project will be listed below.



*Figure.10. The visualization page*

2.Select the required project and click, all the analysis results(heat map, sentiment analysis, etc) will be shown here. And user could select specific image to enlarge the picture.





*Figure.11. The visualization page with details*

**6. API Documentation**

**POST** /api/projectTask/upload

Description: Uploads new projectTask json file to S3.

Request Header: Content-Type: multipart/form-data

Request Body: file stream: .json file.

Response: status code 200, Body ‘Upload projectTask.json Successfully!’

**GET** /api/projectTask/get\_projectTask\_list

Description: Retrieves a list of available projectTasks.

Response: status code 200, Body JSON array of filenames.

status code 500, Body ‘List Acquisition Failed: <error message>’

**GET**  /api/projectTask/get\_projectTask?file=<filename>

Description: Retrieves the specified .json file based on the file name.

Response: status code 200, Body JSON file.

status code 400, Body ‘Missing File Name’

status code 500, Body ‘Read Failed: <error message>’

**POST** /api/recording/upload

Description: Uploads recording segments to S3.

Request Header: Content-Type: multipart/form-data

Request Body: FormData with fields projectName, uuid, firstName, lastName, email, taskIndex, recordedScreen, recordedCamera, and recordedAudio.

Response: status code 200, Body JSON {

"status": "success",

"keys": {

"recordedScreen": "recording\_results/.../task\_1/screen.webm",

"metadata": "recording\_results/.../20250520123045\_user\_data.json"

}

}

status code 500, Body JSON {

"status": "error",

"message": "<error message>"

}

**GET** /api/visualization/get\_project\_list

Description: Retrieves a list of available project list.

Response: status code 200, JSON array of project name.

status code 500, Body "error": "<error message>".

**GET**  /api/visualization/get\_heatmap\_list/<project\_name>

Description: Retrieves a list of available heatmap based on provided project name.

Response: status code 200, JSON array of url for image.

status code 500, Body "error": "<error message>".

**7. Testing**

# Backend tests

pytest -q -s --disable-warnings --maxfail=1

**8. Operations & Monitoring**

Error Tracking: Sentry integration for both frontend and backend.

Health Checks: /health endpoint returns service status.

**9. FAQ & Troubleshooting**

Mixed Content: for the deployed case, the cloudfront applied the HTTPS to frontend, and the backend is applied to HTTP. Then please set the behavior for cloudfront to accept both HTTP and HTTPS request.

Permission for Screen sharing: In order to prevent the affect from irrelevant content during the analysis, then please select the current browser when give the permission to screen sharing

**10. Changelog**

V0.0.1(2025-03-23): Initial release with full recording flow, tester could start screen, camera, and audio recording with one step, then tester could download the recording results.

v1.0.0 (2025-03-30): Header and footer bar, camera recording could be open and close at anytime. Red frame and voice to remind recording status.

V2.0.0 (2025-04-06): Reallocate the UI components. Fixe bugs and address issues encountered during the recording process. Deploy the frontend to S3.

V2.1.0 (2025-04-13): Implement the uploading of recording results to AWS S3. Deploy the backend to EC2.

V3.0.0 (2025-04-20): Reallocate the UI components, separate the authorization process to four steps.

V3.1.0 (2025-04-27): Implement the popup window and CMS system. Store recording results with a clear structure.

V3.2.0 (2025-05-11): Implement the task based recording. Fixe bugs and address issues encountered during the recording process.

V3.3.0 (2025-05-18): Implement the visualization page. Update the API to RESTful and redeploy the frontend and backend to AWS.

**11. License**

This project is licensed under the MIT License.

**12. Appendix**