Frontend Documentation

Authentication

Front-end (Client) establishes a secure HTTPS connection with back-end (Server) to create a token-based session. Client uses public key of server's key pair to generate a key.

Client sends raw user credentials, server hashes and encrypts before storing credentials. Server uses the same algorithms to decipher.

Library Used

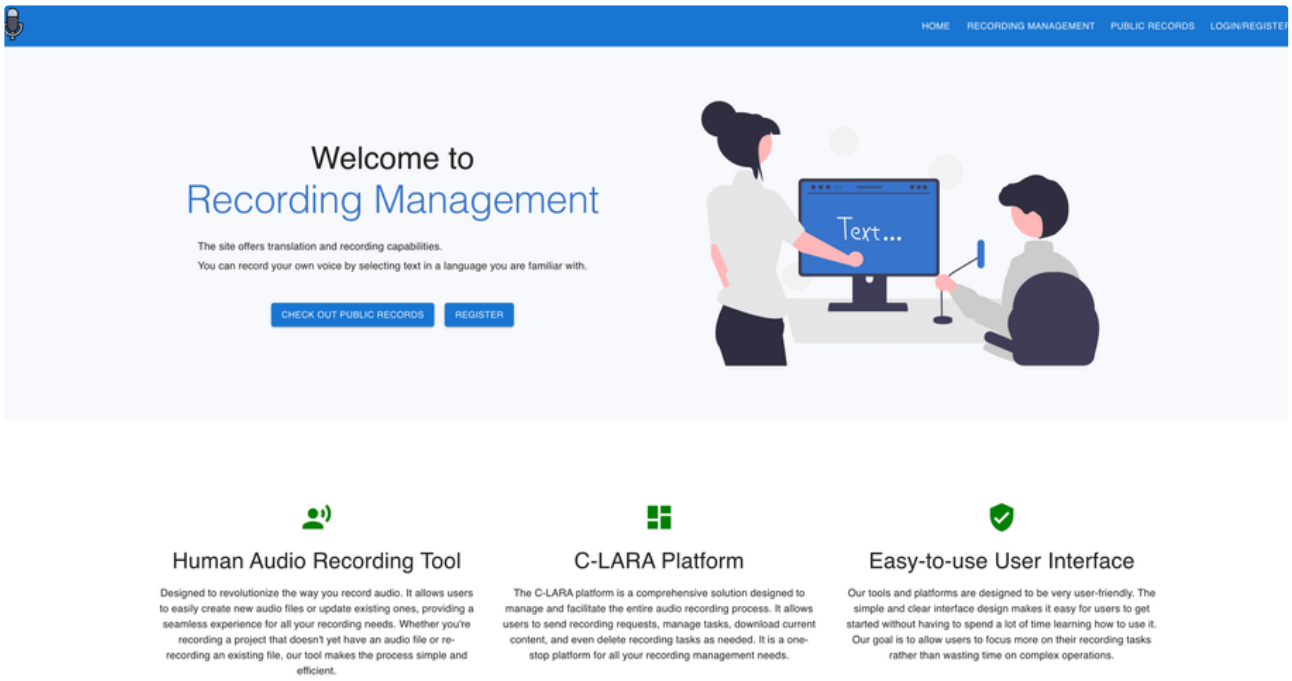
- React: the framework
- React MUI: the component library
- Axios: library to make ajax requests
- react-router: used for client side routing
- react-media-recorder: the functionality to record audio
- zustand: to share state between components without prop drilling
- TypeScript: provide type safety for JavaScript
- Vite: modetn packaging tool for frontend
- Eslint: code linter
- Prettier: code formatter

Folder Structure

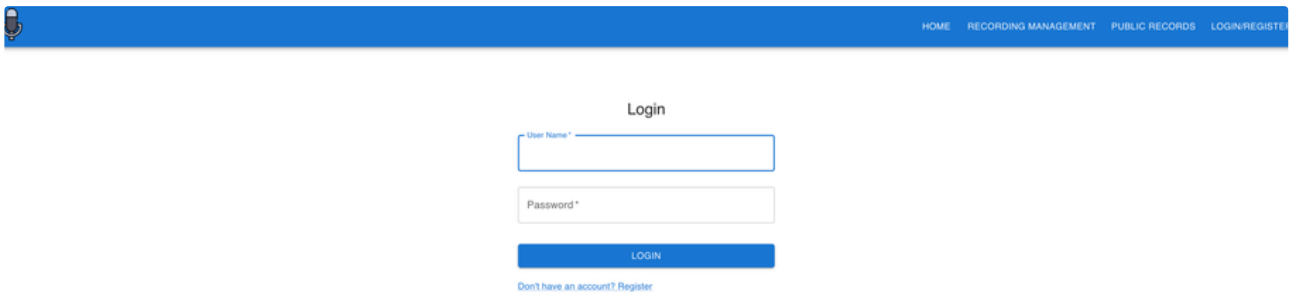
```
Web/                                # frontend project is in the web folder
├── dist                            # the packaged files
├── public                          # the public static files
└── src/                            # the source code
    ├── assets/                     # the asset files
    │   └── react.svg
    ├── components/                 # the React components
    │   ├── Header.tsx
    │   ├── PublicRecordsList.tsx
    │   ├── Recorder.tsx
    │   └── TaskList.tsx
    ├── views/                      # the views for the application
    │   ├── HomeView.tsx
    │   ├── LoginView.tsx
    │   ├── ProfileView.tsx
    │   ├── PublicRecordsView.tsx
    │   ├── RecordingView.tsx
    │   ├── RegisterView.tsx
    │   └── RootView.tsx
    ├── api.ts
    ├── App.tsx
    ├── entity.ts
    ├── env.d.ts
    ├── main.tsx
    ├── store.ts
    ├── T&D.ts
    ├── utils.tsx
    └── vite-env.d.ts
```

Frontend Pages

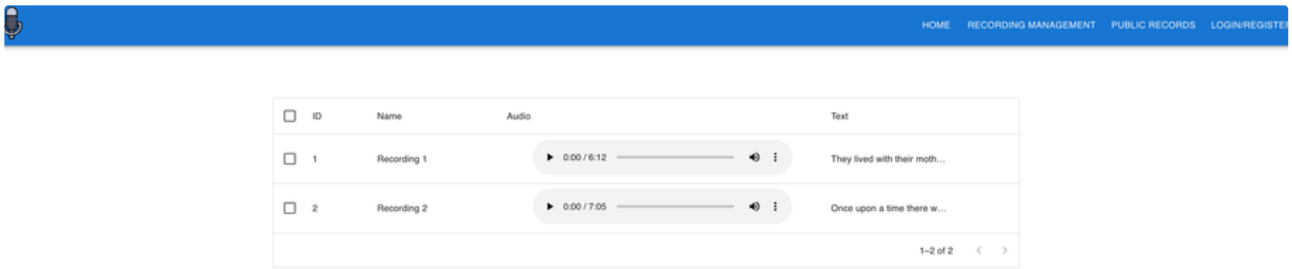
1. Home: welcome texts



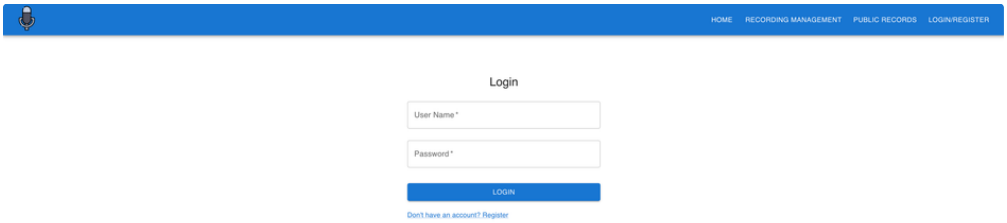
2. Recording Management: the page where user can record an audio and upload to backend, and also manages his/her own audios.

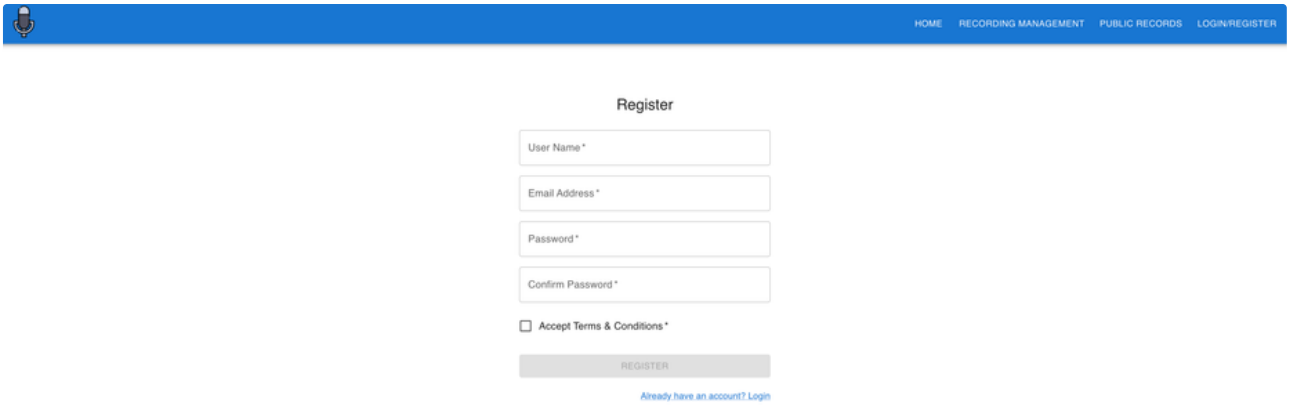


3. Public Recordings: the page where the user can get access to the publicly available recordings.

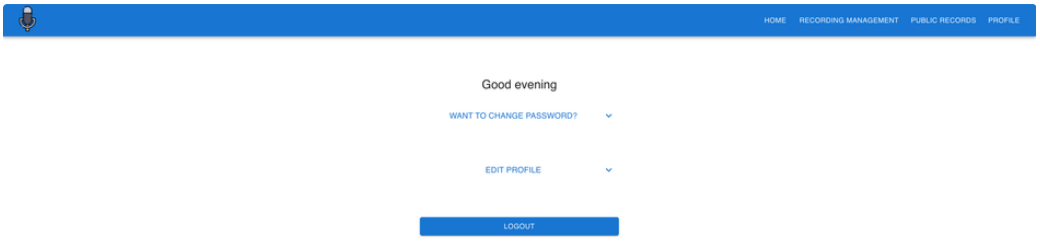


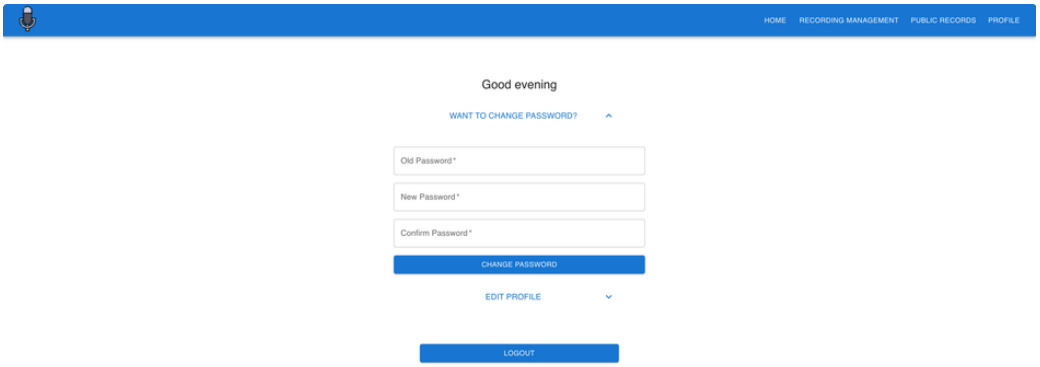
4. Sign in/up: where the user can sign in/up to gain access to the system.

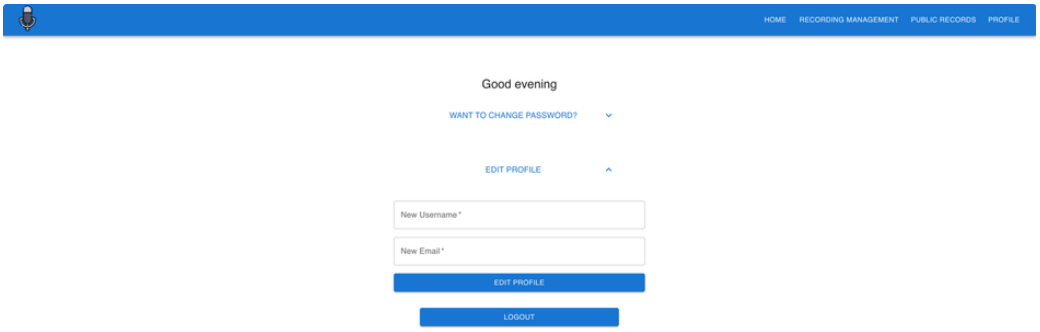




5. Profile: allowed user change password or change user name and email, and logout account.







Expected Schema:

- Post sign-in user

```
interface UserLogin {
  username: string;
  password: string;
}
```

- Post sign-up user

```
interface UserSignUp {
  username: string;
  email: string;
  password: string;
```

```
  isAdmin: boolean (default: False)
}
```

- Receive access token

```
interface UserLoginResponse {
  isAdmin: boolean;
  access: string;
  refresh: string;
}
```

- Recording

```
interface Recording {
  task_id: string;
  block_id: string;
  text: string;
  file: string;
}
```

- Change Password

```
interface ChangePasswordDto {
  current_password: string;
  new_password: string;
}
```

- Edit Profile

```
interface EditProfileDto {
  username: string;
  email: string;
}
```

Expected Endpoints

- user sign up `POST /api/user/signup`
- user sign in `POST /api/user/login`
- get all recording task `GET /api/task`
- add recording task `POST /api/task/{task_id}/{block_id}`
- delete recording task with id `DELETE /api/recording/{task_id}/{block_id}`
- user change password `PUT /api/user/change-password`
- user edit profile `PUT /api/user/edit-profile`