Tour Guide Translator

PASSED

Tour Guide Translator is an app that allows tour guides to talk in their language and the tourists can listen to them in their language in real time. The tour guide creates a room for a certain trip and the tourists join the room. Tourists can also send questions to the tour guide in their language and they will be translated to the language of the tour guide. The app makes sure that every tourist in the trip will be able to understand the tour guide.

Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

Prerequisites

To use our project, you'll need to install Node.js to install our dependencies and run our scripts.

We built our project using a toolchain called Expo. You could either download the Expo XDE from their GitHub or use their command line interface. We prefer using the XDE because it has more relevant settings like "Development mode" for error handling and also supports using a phone simulator. For more information regarding installation of the XDE and using simulators, visit Expo Installation page.

Installing

For installing Expo command line interface type in your terminal:

```
npm install -g exp
```

To run the project on your own phone/tablet, navigate to Tour-Guide-Translator folder in the root of the project within your terminal and type:

```
npm install && exp start
```

Now to run the app on your phone, you must first download the Expo app from the App Store or Google Play Store and scan the given code (currently only android) or open the link provided in the terminal .

Keys for services

We are using services that need some authorization so you'll need to get your hands on some keys for billing and authorization purposes.

Amazon Web Services

We use Amazon Polly for Text to Speech.

First you'll need to login to the Amazon web services console. Then create a user with the AmazonPollyFullAccess permission and then attach an access key to that user. To use your key, navigate to our polly folder and input your keys to the config variable in both speak.js and voices.js.

Google Cloud Platform

We use Google for our translations so you'll need to get keys from them.

First login to Google Cloud Platform and create an new project. Navigate to APIs & Services -> Credentials. Under Credentials, you can press create credentials and create an API Key. Copy that key and put it in the translate.js file as the variable apiKey.

Firebase

Firebase hosts our database and handles authentication.

To setup your own database, you'll need to:

- Login to your firebase account
- Create a new project
- Enable email authentication under sign-in method
- Get your key from web setup and put it in the config variable in App.js.
- Put the database rules into Firebase. They can be found in rules.json.

Running the tests

We test our React Native components with Jest, using snapshot testing. The tests are located in the __tests__ folder and they are divided into two files, one for screen components and one for all other components. To run all tests use the command:

```
npm test
```

And to update the snapshots:

```
npm test -- -u
```

Coding style

Our coding rules are defined with ESLint and can be found in the .eslintrc file. The basis of our rules are from RallyCoding but we customized them to our liking. We use Prettier to format the code according to these rules each time we save a file. We recommend adding ESLint and Prettier extensions to your code editor. We use Visual Studio Code.

Deployment

To deploy we use CircleCI for continous integration. We use workflows with CircleCI to deploy automatically when we merge to our master branch. If our tests pass, CircleCI continues to deploy with Expo.

To get started, take a look at our CircleCI folder at the root of our project called .circleci. Inside the folder we have our config.yml which CircleCI uses to run our custom scripts for testing and deploying. The config file has two jobs: build and deploy. Both jobs run custom scripts which are located at the root of our project in a folder called circleScripts. As you can see, our workflow checks which branch we are pushing to and makes sure to deploy if we are on our master branch.

Our circleScripts mostly run our scripts from package.json in the right order. For instance we need to run expLogin first to be able to run expPublish.

To login to our Expo dummy account and publish from the command line, run from the Tour Guide Translator folder:

```
npm expLogin && npm expLogin
```

Contributing

There are two holy branches. First we have a master branch which needs approval from CircleCI and a code review from another developer to be able to push code to master. Then we have a Dev branch which is where we test newly merged code before making a pull request to the master branch.

We use branch-by-feature to make sure we dont interfere and break others code. When creating a new branch you must branch from Dev and make sure to pull from Dev frequently.

Authors

- **Bjartur Fannar Stefansson** - *Initial work* - bjarturfs
- **Gunnar Birnir Ólafssom** - *Initial work* - gunnarbirnir
- **Ævar Þór Gunnlaugsson** - *initial work* - aevartg