Digital Divide App – Developer Documentation

Team Members

Full Stack: Andre Fernandez
Kristen Lane

Front-end: Trust Museta
Josiah Louis
Luc Nguyen

Back-end: Jeffrey Callender
Christopher Ehrhardt
Guillermo Castaneda

CEN-4910C-27087

April 22, 2021

Dr. Lisa Macon

1. Purpose.

   The purpose of this document is to assist developers in setting up the development environment for the Digital Divide App and to provide the starting set guidelines for its continued development.

2. Assumptions.

   This documentation assumes that the developer is knowledgeable in HyperText Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript, Express, and React, which serve as the front-end libraries required for building the app's user interface. They will allow the developer to dynamically generate and interact with the Document Object Model (DOM).

3. Requirements.

   The Digital Divide app was developed using a standard MERN stack, which consists of MongoDB, Express, React, Node.js.

   3.1 Node.js. The latest version includes npm.

   3.2 Express. Express is installed via npm. Express is used for the back-end/server-side of the app.

   3.3 React. React is used for the front-end/client-side of the app.

   3.4 MongoDB. For ease of development, considering the use of publicly available data used in the app, the development team decided to use MongoDB Atlas. However, it is recommended that MongoDB be locally installed for the development of additional features that require a greater level of security (e.g., user authentication). MongoDB Atlas access credentials for this project are stored in **/config.js**.

   3.5 Git. For version control.

4. Installation for local development

   4.1 Clone the repo.

   4.2 Install Node.js.

   4.3 Open a terminal instance on the repo's root folder and run:

```
$ npm install
$ cd client && npm install
```

   4.4 Update the MongoDB access credentials in **/config.js**:

```
module.exports = {
   jwtSecret: JWTSECRET || '123456',
   mongodburi: '' //your mongodb uri
}
```

   4.5 Configure the local server's URL in **/client/src/config.js**

```
module.exports = {
   BASE_URL: '' // server URL
}
```

   4.6 Finally, in a terminal instance on the repo's root folder:

```
$ cd client
$ yarn
$ yarn build
$ cd ..
$ npm start
```

4.7 Open a browser window and navigate to **http://localhost:5000**

5. Heroku deployment

For Heroku deployment, the following must be included in **/package.json**:

```
  ...
  "scripts": {
     ...
     "heroku-postbuild": "NPM_CONFIG_PRODUCTION=false npm install --prefix client && npm run
build --prefix client",
     ...
  },
  ...
```

Then, add the following in /app.js:

```
  if (process.env.NODE_ENV === 'production') {
     app.use(express.static(path.join(__dirname, 'client/build')));
     app.get('*', (req, res) => {
        res.sendFile(path.join(__dirname, '/client/build/index.html'))
     })
  }
```

6. Project Codebase.

The app follows current design standards that aim to separate logic, views, and data.

7. Project Structure.

```
.
├── Procfile
├── README.md
├── app.js
├── client
│   ├── jsconfig.json
│   ├── package.json
│   ├── public
│   │   ├── apple-icon.png
│   │   ├── favicon.ico
│   │   ├── index.html
│   │   └── manifest.json
│   ├── src
│   │   ├── Config.js
│   │   ├── assets
│   │   │   ├── App.css
│   │   │   ├── data.json
│   │   │   └── img
│   │   │       └── bg.png
│   │   ├── index.js
│   │   ├── layouts
│   │   │   ├── ChartCard.js
│   │   │   ├── Home.css
│   │   │   ├── Home.js
│   │   │   ├── Legend.js
│   │   │   ├── Maps.js
│   │   │   ├── SpeedCard.js
│   │   │   ├── SpeedTest.css
│   │   │   └── SpeedTest.js
│   │   └── logo.svg
```

```
│   └── yarn.lock
├── config.js
├── data.json
├── models
│   └── map.js
├── package-lock.json
├── package.json
└── routes
    └── map.js
```

7.1 Root

/app.js

Contains the routes to the middleware chain and acts as the server file. It is used to connect to the MongoDB database by passing the MongoDB connecting string to the mongoose component.

/data.json

The data used by the app to display the map tiles that represent the speed ranges. This data is generated with an API that directly accesses Ookla publicly available datasets. That API is not currently part of the app.

/package.json

Contains the app's name, version, description, keywords, dependencies and development dependencies required by the app.

7.2 Client (front-end)

/client/jsconfig.js

Instructs the JavaScript languages services how to validate and compile **.js** files. Defines the base URL path of the application and sets the default path to resolve non-relative module names.

/client/public

This folder automatically created by create-react-app. All files in the Public folder can be referenced from index.html using the **%PUBLIC_URL%** syntax.

/client/public/Favicon

The app's favicon.

/client/public/manifest.json

This file describes the app.

/client/public/index.html

Used as the app's template file. It links the `favicon.ico` and `manifest.json` files.

/client/src/assets

This folder contains the images and style sheets used by the app. For this project, it also contains the `data.json` with the app's map data.

/client/src/layouts

The app's UI layout structure that references the app's routers and models.

/client/src/layouts/ChartCard.js

Holds the properties (title, data, type, options) passed from `/client/src/layouts/Home.js` for the jitter and latency pieCharts displayed on the app's dashboard.

/client/src/layouts/Home.css

The stylesheet used to style the dashboard, including the map and legend display.

/client/src/layouts/Home.js

Contains JavaScript that defines and instantiates the front-end data for display in the browser. All back-end services are called from this file via HTTP requests using Axios to manage responses.

/client/src/layouts/Legend.js

Creates the map legend and defines the map features.

/client/src/layouts/Maps.js

App dashboard map display configuration and customization.

/client/src/Config.js

Speed range step and color scheme configuration.

The main map elements in the app refer to specific download speed ranges as steps. A color is assigned to each step in the download speed range. Additional steps may be created, and a unique color may be assigned to each step in the range. Naturally, more steps will provide a greater level of detail in the rendered map.

/client/src/index.js

The default entry point of the application. It loads the CSS styles and the dependencies required to render the application on the web browser.

/client/package.json

Contains the app's client's dependencies (development and optional) and scripts required by the app's front-end.

7.3 Server (back-end)

/models/map.js

The app uses a single model: the MapSchema with the data necessary to display the map overlay on the dashboard.

```
const MapSchema = mongoose.Schema({

  // The neighborhood ID
  NeighID:{
    type:String,
    required:true
  },

  // The map type. Constant as 'feature'
  type: {
    type: String,
    required: true
  },

  /**
   * The neighborhood properties
   * NeighID       - The neighborhood ID
   * NeighName     - The neighborhood name
   * avg_d_mbps_wt - Average download speed for the neighborhood
   * avg_u_mbps_wt - Average upload speed for the neighborhood
   * avg_lat_ms_wt - Average latency for the neighborhood
   * tests         - Speed tests requested in the neighborhood
   * devices       - Devices that requested speed tests in the neighborhood
   */
  properties: {
    type: Object,
    required: true
  },

  // Geographical coordinates that delimit the neighborhood limits
  geometry: {
    type: Object,
    required: true
  }
})
```

/routes/map.js

The routes in the file fetch the map data from **/data.json** and **/model/map.js**.

/python/manage.py

This Python script is used to fetch the publicly available geo-data from Ookla Open Data.