Develop a React based Frontend Web Application to Display the Trains Schedule

React App.Js file

```
import React, { Component } from 'react';
import { Tab, Tabs, TabList, TabPanel } from 'react-tabs';
import { withTranslation } from 'react-i18next';
import './App.css';
import Header from './components/Header/Header';
import SearchBar from './components/SearchBar/SearchBar';
import DataDisplay from './components/DataDisplay/DataDisplay';

const API = 'https://rata.digitraffic.fi/api/v1/';

class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      error: null,
      stations: [], // needed as sometimes origin or destination isn't a passenger station
      passengerStations: [], //used for displaying suggestions in search input
      todaysTrains: [],
      selectedStation: null,
      arrivalData: [],
      departureData: [],
      tabIndex: 0 // 0 = arrivals, 1 = departures
    };
  }

  componentDidMount() {
    this.fetchAll();
  }

  componentDidUpdate() {
    document.title = this.props.t('title'); // when user switches language
  }

  handleInputChange = selectedStation => {
    this.setState({ selectedStation });
    this.filterData(selectedStation);
  };

  fetchAll() {
    const dateNow = new Date().toISOString().slice(0, 10); // format of type 2019-02-12
    Promise.all([
      fetch(`${API}metadata/stations`).then(response => response.json()),
      fetch(`${API}trains/${dateNow}`).then(response => response.json())
    ]).then(
      allResponses => {
```

```
        const stations = allResponses[0].map(station => ({
          value: station.stationShortCode,
          label: station.stationName.includes(' asema')
            ? station.stationName.slice(0, -6)
            : station.stationName
        }));
        const passengerStations = allResponses[0]
          .filter(station => station.passengerTraffic === true)
          .map(station => ({
            value: station.stationShortCode,
            label: station.stationName.includes(' asema')
              ? station.stationName.slice(0, -6)
              : station.stationName
          }));
        const todaysTrains = allResponses[1];
        this.setState({ stations, passengerStations, todaysTrains });
      },
      error => {
        this.setState({ error });
      }
    );
  }

  filterData(selectedStation) {
    const { todaysTrains, stations } = this.state;
    const dateTimeNow = new Date().toJSON();
    const filteredData = todaysTrains
      .map(train => {
        const trainNumber = train.commuterLineID
          ? `Commuter train ${train.commuterLineID}`
          : `${train.trainType} ${train.trainNumber}`; //special case for Commuter trains
who have their own ID
        const originShortCode = train.timeTableRows[0].stationShortCode; // the origin
(Lähtöasema) is the first entry in the timeTable
        const origin = stations.find(
          station => station.value === originShortCode
        ).label; // retrieves the full name of station by short code
        const destinationShortCode =
          train.timeTableRows[train.timeTableRows.length - 1][
            'stationShortCode'
          ];
        const destination = stations.find(
          station => station.value === destinationShortCode
        ).label;

        let scheduledArrivalTime; // arrivals
        let actualArrivalTime;
        const arrivalTimeTable = {
          ...train.timeTableRows.filter(
            element =>
              element.stationShortCode === selectedStation.value &&
              element.type === 'ARRIVAL'
          )[0]
        };
```

```
      if (arrivalTimeTable) {
        if (arrivalTimeTable.hasOwnProperty('scheduledTime')) {
          scheduledArrivalTime = arrivalTimeTable.scheduledTime;
        }
        if (arrivalTimeTable.hasOwnProperty('actualTime')) {
          actualArrivalTime = arrivalTimeTable.actualTime;
        } else if (arrivalTimeTable.hasOwnProperty('liveEstimateTime')) {
          actualArrivalTime = arrivalTimeTable.liveEstimateTime;
        } else {
          actualArrivalTime = false;
        }
      }
      let scheduledDepartureTime; // departures
      let actualDepartureTime;
      const departureTimeTable = {
        ...train.timeTableRows.filter(
          element =>
            element.stationShortCode === selectedStation.value &&
            element.type === 'DEPARTURE'
        )[0]
      };
      if (departureTimeTable) {
        if (departureTimeTable.hasOwnProperty('scheduledTime')) {
          scheduledDepartureTime = departureTimeTable.scheduledTime;
        }
        if (departureTimeTable.hasOwnProperty('actualTime')) {
          actualDepartureTime = departureTimeTable.actualTime;
        } else if (departureTimeTable.hasOwnProperty('liveEstimateTime')) {
          actualDepartureTime = departureTimeTable.liveEstimateTime;
        } else {
          actualDepartureTime = false;
        }
      }

      return {
        ...train,
        trainNumber,
        origin,
        destination,
        scheduledArrivalTime,
        actualArrivalTime,
        scheduledDepartureTime,
        actualDepartureTime
      };
    })
    .filter(train => train.trainCategory !== 'Cargo') // removes cargo entries
    .filter(
      train =>
        train.actualArrivalTime > dateTimeNow ||
        train.scheduledArrivalTime > dateTimeNow ||
        train.actualDepartureTime > dateTimeNow ||
        train.scheduledDepartureTime > dateTimeNow
    ); // filters only time entries after "now"
  const arrivalData = filteredData
```

```
    .filter(entry => typeof entry.scheduledArrivalTime !== 'undefined')
    .map(entry => ({
      ...entry,
      actualTime: entry.actualArrivalTime,
      scheduledTime: entry.scheduledArrivalTime
    }));
  const departureData = filteredData
    .filter(entry => typeof entry.scheduledDepartureTime !== 'undefined')
    .map(entry => ({
      ...entry,
      actualTime: entry.actualDepartureTime,
      scheduledTime: entry.scheduledDepartureTime
    }));
  this.setState({ arrivalData, departureData });
}

render() {
  const {
    error,
    tabIndex,
    arrivalData,
    departureData,
    todaysTrains
  } = this.state;
  const { t } = this.props;
  const errorDisplay = (
    <div className="error">{error ? error.message : null}</div>
  );
  const content =
    todaysTrains.length === 0 ? (
      <p className="loading">{t('Loading')}...</p>
    ) : (
      <Tabs
        selectedIndex={tabIndex}
        onSelect={tabIndex => this.setState({ tabIndex })}
      >
        <TabList>
          <Tab>{t('Arrivals')}</Tab>
          <Tab>{t('Departures')}</Tab>
        </TabList>
        <TabPanel>
          <DataDisplay display="arrival" filteredData={arrivalData} />
          {errorDisplay}
        </TabPanel>
        <TabPanel>
          <DataDisplay display="departure" filteredData={departureData} />
          {errorDisplay}
        </TabPanel>
      </Tabs>
    );
  return (
    <div className="App">
      <Header />
      <SearchBar
```

```
          placeholder={t('Look for train station')}
          noOptionsMessage={inputValue => t('Not found')}
          options={this.state.passengerStations}
          onChange={this.handleInputChange}
        />
        {content}
      </div>
    );
  }
}

export default withTranslation()(App);
```

Index.js File

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import './i18n';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));


serviceWorker.unregister();
```

CSS style sheet for the app

```
.App {
  font-size: 14px;
}

.loading {
  margin: 20px 50px;
}

.error {
  margin: 20px 15px;
  color: red;
}

/* all react-tabs code was copied from node_modules/react-tabs/react-tabs.css then
overridden and edited here */
.react-tabs {
  -webkit-tap-highlight-color: transparent;
  margin-left: 30px;
```

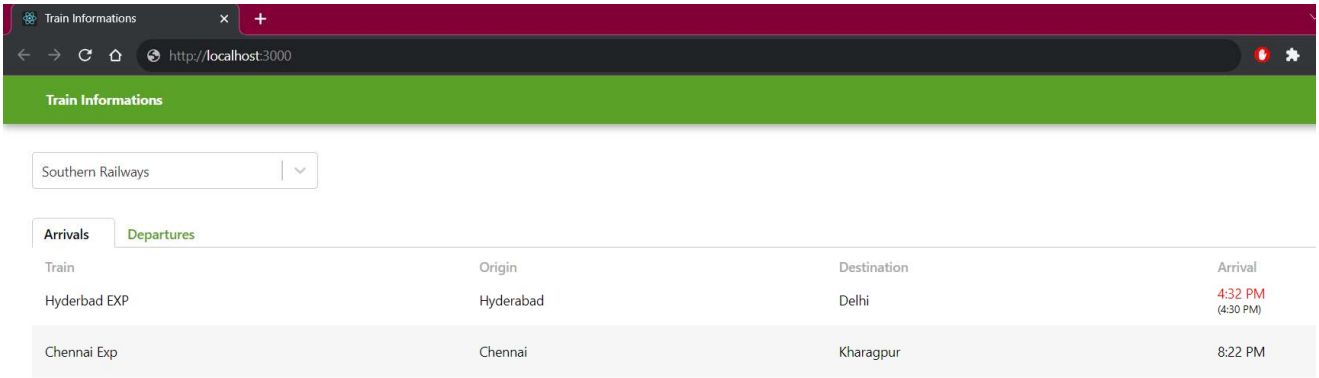
```
  /* max-width: 450px; */
  /* width: 450px; */
  width: 100%;
}

.react-tabs__tab-list {
  border-bottom: 1px solid #ccc;
  margin: 0 0 10px;
  padding: 0;
  color: #59a127;
  font-weight: 500;
}

.react-tabs__tab {
  min-width: 62px;
  display: inline-block;
  border: 1px solid transparent;
  border-bottom: none;
  bottom: -1px;
  position: relative;
  list-style: none;
  padding: 6px 12px;
  cursor: pointer;
}

.react-tabs__tab--selected {
  background: #fff;
  border-color: #ccc;
  color: black;
  border-radius: 5px 5px 0 0;
}

.react-tabs__tab--disabled {
  color: GrayText;
  cursor: default;
}

.react-tabs__tab:focus {
  box-shadow: 0 0 5px hsl(208, 99%, 50%);
  border-color: hsl(208, 99%, 50%);
  outline: none;
}

.react-tabs__tab:focus:after {
  content: '';
  position: absolute;
  height: 5px;
  left: -4px;
  right: -4px;
  bottom: -5px;
  background: #fff;
}

.react-tabs__tab-panel {
```

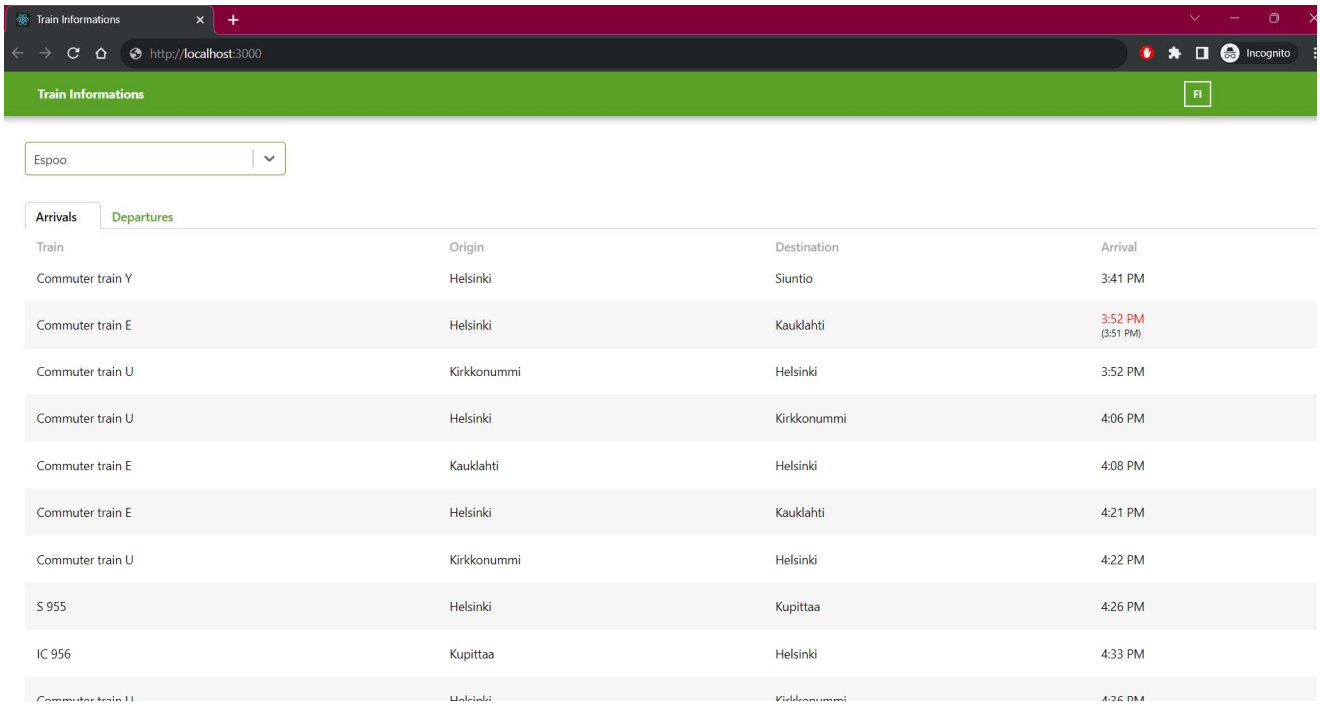
```
  display: none;
}

.react-tabs__tab-panel--selected {
  display: block;
}
```

Source code for loading HTML page

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-fit=no"
    />
    <meta name="theme-color" content="#000000" />

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <title>Train Schedule System </title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>

  </body>
</html>
```

Screenshots of the Program Running




Train Informations
http://localhost:3000
Incognito
Train Informations
FI
Espoo
Arrivals
Departures
Train
Origin
Destination
Arrival
Commuter train Y
Helsinki
Siuntio
3:41 PM
Commuter train E
Helsinki
Kauklahti
3:52 PM
(3:51 PM)
Commuter train U
Kirkkonummi
Helsinki
3:52 PM
Commuter train U
Helsinki
Kirkkonummi
4:06 PM
Commuter train E
Kauklahti
Helsinki
4:08 PM
Commuter train E
Helsinki
Kauklahti
4:21 PM
Commuter train U
Kirkkonummi
Helsinki
4:22 PM
S 955
Helsinki
Kupittaa
4:26 PM
IC 956
Kupittaa
Helsinki
4:33 PM