FINAL REPORT
INTERNSHIP

Author: Altair Cerniauskas – 635-069720

Program: Programmer Analyst / Internet Solutions Developer – LEA.9C

Host Organization: Mada Community Center

Address: 6875, boulevard Décarie Montreal, QC, Canada H3W 3E4

Phone: (514) 342-4969

Email: mada@madacenter.com

Stage Perid: from **Oct 25** to **Dec 19, 2018**

Montreal-QC 2018-Dec

Contents

1) Introduction

This report describes the work experience I gained during the completion of my internship which is part of the curriculum of the CDI College - Programmer Analyst course.

The internship took place at the Mada Community Center between October 25 and December 19, 2018. Founded in 1993, MADA is a volunteer-based organization that evolved from a small neighbourhood centre to Montreal's central address for fighting poverty in our community.

The objective of the internship is to know and collaborate with the activities that involve the Website (https://madacenter.com), and the tzedakah-pushke mobile application (https://tzedakah-pushke.madacenter.com) which belong to the

Mada Community Center.

2) Project

1ª part - Develop a Donation interactive module for Tzedakah-Pushke mobile application (https://tzedakah-pushke.madacenter.com).

2ª part - Develop a Recreational Activity interactive module for Tzedakah-Pushke mobile application (https://tzedakah-pushke.madacenter.com).

3ª part – Update and support for Mada Website (https://madacenter.com).

3) Computer System and Languages

Tzedakah-Pushke mobile application was created by React Most Wanted that is a react Starter Kit based on Create React App and Material-UI that uses Firebase. It is Best Practice Project PWA (Progressive Web Application).

React (also known as React.js or ReactJS) is a **JavaScript library** for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.

Mada Website was created by WordPress (HTML, CSS, JavaScript).

4) Documentation

https://reactjs.org/docs/getting-started.html

https://www.npmjs.com/package/firebase-tools

https://redux.js.org/#learn-redux-from-its-authors

https://javascript.info

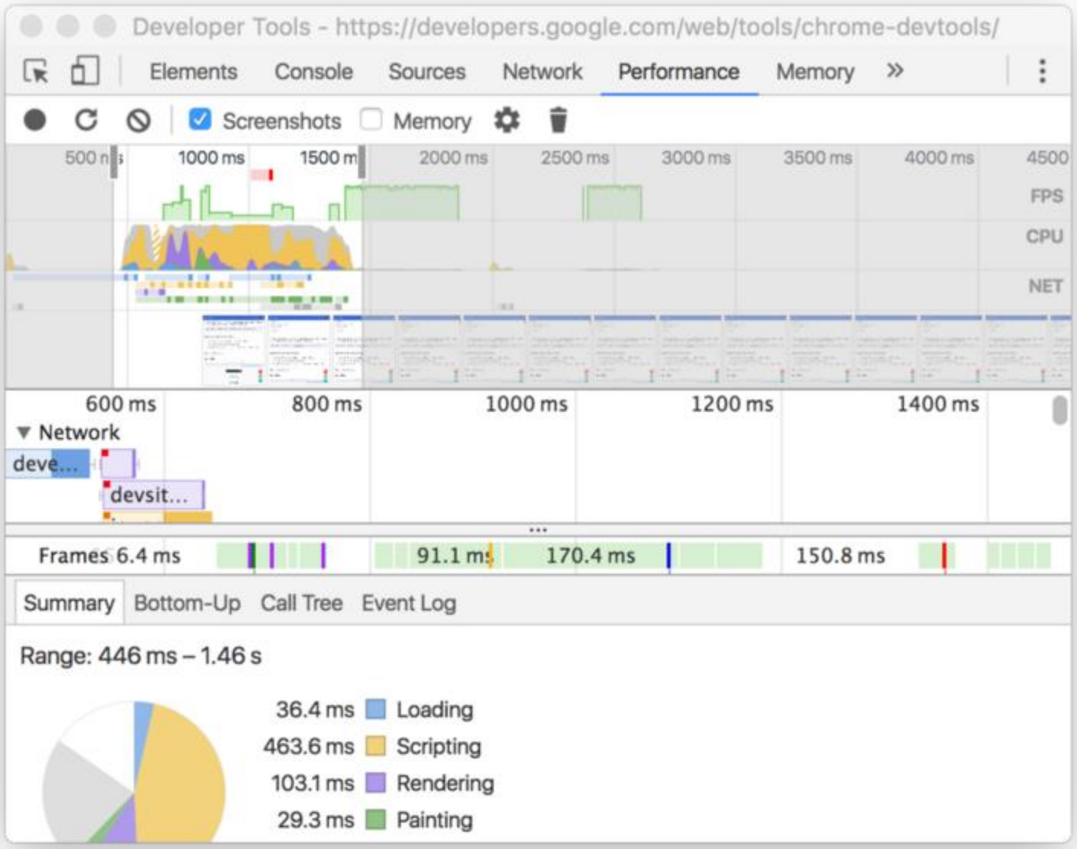
https://github.com/TarikHuber/react-most-wanted/wiki

https://github.com/peterh32/react-drag-drop-container

https://www.npmjs.com/package/react-unity-webgl

5) Process of Analysis

Chrome Developer Tools—Performance Analysis for React Apps



Range: 446 ms – 1.46 s

36.4 ms	Loading
463.6 ms	Scripting
103.1 ms	Rendering
29.3 ms	Painting

Chrome Developer Tools enable functionality from withing the Chrome browser which allows the developer to:

- Simulate different types of devices like phones and tablets
- Test responsive and device-specific viewports
- Sensor emulation
- Inspect and edit CSS and the DOM
- Inspect the contents of the console log
- View and debug Javascript code
- View network activity

- Profile Javascript CPU activity
- Identify and fix memory problems
- Debug Progressive web applications
- Inspect and manage storage, databases, and caches
- Inspect and delete cookies
- Inspect resources
- Understand security issues

6) Technique used in Project

Tzedakah-Pushke mobile application is a Progressive Web App, It must be:

- Progressive - Work for every user, regardless of browser choice, because they are built with progressive enhancement as a core tenet.
- Responsive - Fit any form factor, desktop, mobile, tablet, or whatever is next.
- Connectivity independent - Enhanced with service workers to work offline or on low quality networks.
- App-like - Use the app-shell model to provide app-style navigation and interactions.
- Fresh - Always up-to-date thanks to the service worker update process.
- Safe - Served via HTTPS to prevent snooping and ensure content has not been tampered with.

- Discoverable - Are identifiable as “applications” thanks to W3C manifests and service worker registration scope allowing search engines to find them.
- Re-engageable - Make re-engagement easy through features like push notifications.
- Installable - Allow users to “keep” apps they find most useful on their home screen without the hassle of an app store.
- Linkable - Easily share via URL and not require complex installation.

7) Technical Support Tasks

Technical support activities were limited to the app's development process.

There was no demand for external technical support because APP has not yet been launched. Possibly with the app laundering the company should create a customer support area. Some offices may have an IT department with one or two Technical Support Engineers or more depending on application demand.

8) Program Codes

FeedCard.js (just part of the code that call the Class: DragMoneyToDonate)

```
…
        <Dialog
          open={this.state.dialogConfirmationOpen}
          onClose={this.handleDialogConfirmationClose}
          aria-labelledby="alert-dialog-title"
          aria-describedby="alert-dialog-description"
        >
```

```
          <DialogTitle id="alert-dialog-title">{intl.formatMessage({ id:
'feed_donation_confirm_title' })}</DialogTitle>
          <DragMoneyToDonate onChangeAmount={this.handleChangeAmount}
totalDonate={this.state.amount}/>
          <DialogActions style={{ margin: 0 }}>
            <p>Confirm donation?    </p>
            <Button onClick={this.donateAction} color="primary">
              Ok
            </Button>
          </DialogActions>
        </Dialog>
….
```

DragMoneyToDonate.js

```
import React from 'react';
import { withStyles } from '@material-ui/core/styles';
//import Card from '@material-ui/core/Card';
//import CardHeader from '@material-ui/core/CardHeader';
//import CardContent from '@material-ui/core/CardContent';
//import CardActions from '@material-ui/core/CardActions';
import DragElement from './DragElement';
import DropElement from './DropElement';
//import Button from '@material-ui/core/Button';


const styles = theme => ({
  card: {
    // minWidth: 275,
    maxWidth: 330,
    margin: 8,
    backgroundColor: '#6cbac9',
    textAlign: 'center',
    alignContent: 'center',
  },
  media: {
    height: 0,
    paddingTop: '56.25%', // 16:9
  },
  DragMoneyToDonate: {
    align: 'center',
    alignContent: 'center',
  },
  actions: {
    display: 'flex',
    alignContent: 'center',
  },
  text: {
    color: '#002d41',
  },
  text2: {
    color: '#002d41',
  },
});

class DragMoneyToDonate extends React.Component {

  state = {
    //donateimage: "/images/pushkacane.gif",
    onChangeAmount: this.props.onChangeAmount,
    totalDonate: this.props.totalDonate,
  };


  render() {
    //const {classes } = this.props
    //const {title } = this.state
    // we will use this as a custom drag element
    //const customElem = <button style={{marginTop:20,
marginLeft:20}}>Bananas!!</button>
    const contentBoxStyle = {
      backgroundColor: 'white',
      padding: '0 15px 0',
      //display: 'flex',
      //alignItems: 'center',
      //flexDirection: 'column',
    };
    const bigDragBoxStyle = {
      //float:'left',
      width: 215,
      height: 160,
      //alignContent:'center',
      display: 'flex',
      //fexWrap: 'wrap',
      alignItems: 'center',
      flexDirection: 'column',
      //border: '1px solid black',
    };
    const dragBoxStyle1 = {
      float:'left',
      width: 215,
      height: 50,
      marginTop: 0,
      textAlign: 'center'
      //border: '1px solid black',
    };
    const dragBoxStyle2 = {
      float:'left',
      width: 215,
      height: 50,
      marginTop: 0,
      textAlign: 'center'
      //border: '1px solid black',
    };
    const bigDropBoxStyle = {
      //float:'left',
      width: 215,
      height: 170,
      //alignContent:'center',
      display: 'flex',
      //fexWrap: 'wrap',
      alignItems: 'center',
      flexDirection: 'column',
      //border: '1px solid black',
    };
    const dropBoxStyle = {
      float:'left',
      width: 100,
      height: 170,
      marginTop: 0,
      textAlign: 'center',
      //border: '1px solid black',
    };


    return (

        <div className="contentBox" style={contentBoxStyle}>
            <div className="bigDragBox" style={bigDragBoxStyle}>
            <div className="dragBox" style={dragBoxStyle1}>
                <DragElement dragClone={true} dragElemOpacity={0.4} targetKey="donate"
alt="20$" value='20' image="images/20dolars.jpg"/>
                <DragElement dragClone={true} dragElemOpacity={0.4} targetKey="donate"
alt="50$" value='50' image="images/50dolars.jpg"/>
              </div>
              <div className="dragBox" style={dragBoxStyle1}>
                <DragElement dragClone={true} dragElemOpacity={0.4} targetKey="donate"
alt="5$" value='5' image="images/5dolars.jpg"/>
                <DragElement dragClone={true} dragElemOpacity={0.4} targetKey="donate"
alt="10$" value='10' image="images/10dolars.jpg"/>
              </div>
              <div className="DragBox" style={dragBoxStyle2}>
```

```
                <DragElement dragClone={true} dragElemOpacity={0.4} targetKey="donate"
alt="0.25$" value='0.25' image="images/25cents.png"/>
                <DragElement dragClone={true} dragElemOpacity={0.4} targetKey="donate"
alt="0.50$" value='0.50' image="images/50cents.png"/>
                <DragElement dragClone={true} dragElemOpacity={0.4} targetKey="donate"
alt="1$" value='1' image="images/1dolar.png"/>
                <DragElement dragClone={true} dragElemOpacity={0.4} targetKey="donate"
alt="2$" value='2' image="images/2dolars.png"/>
              </div>
            </div>
            <div className="bigDropBox" style={bigDropBoxStyle}>
              <div className="dropBox" style={dropBoxStyle}>
                  <DropElement
                    targetKey="donate"
                    name="donate"
                    src="/images/pushkacane.gif"
                    totalDonate={this.props.totalDonate}
                    onChangeTotalDonate={this.props.onChangeAmount}>
                  </DropElement>
              </div>
            </div>
        </div>
    )
  }
}

export default (withStyles(styles) (DragMoneyToDonate))
```

DragElement.js

```
import React from 'react';
import DragContainer from './DragContainer';

/*
    DragElement is a draggable element (in a DragContainer)
*/

export default class DragElement extends React.Component {
    landedOn(e) {
      console.log('I was dropped on ' + e.dropData.name)
      console.log({'onDrop event passed back to DragElement': e});
    }

    render() {
      // note use of render prop below, rather than child element
      return (
        <DragContainer
          targetKey={this.props.targetKey}
          dragClone={this.props.dragClone || false}
          dragData={{alt: this.props.alt, value: this.props.value}}
          customDragElement={this.props.customDragElement}
          onDrop={this.landedOn}
          render = {() => {
            return <img style={{ margin: 3 }} src={this.props.image}
alt={this.props.alt} height="45"/>
          }}
        />

      );
    }
  }
```

DragContainer.js

```
import React from 'react';
import PropTypes from 'prop-types';


function usesLeftButton(e) {
  const button = e.buttons || e.which || e.button;
  return button === 1;
}

function getFixedOffset() {
  // When browser window is zoomed, IOS browsers will offset "location:fixed" component
coordinates
  // from the actual window coordinates
  let fixedElem = document.createElement('div');
  fixedElem.style.cssText = 'position:fixed; top: 0; left: 0';
  document.body.appendChild(fixedElem);
  const rect = fixedElem.getBoundingClientRect();
  document.body.removeChild(fixedElem);
  return [rect.left, rect.top]
}

function isZoomed() {
  // somewhat arbitrary figure to decide whether we need to use getFixedOffset (above)
or not
  return Math.abs(1 - (document.body.clientWidth / window.innerWidth)) > 0.02;
}

class DragContainer extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      leftOffset: 0,
      topOffset: 0,
      left: 0,
      top: 0,
      clicked: false,
      dragging: false,
    };

    // The DOM elem we're dragging, and the elements we're dragging over.
    this.dragElem = null;
    this.containerElem = null;
    this.sourceElem = null;
    this.currentTarget = null;
    this.prevTarget = null;

    this._isMounted = true;

    // offset factors that occur when dragging in a zoomed-in IOS browser
    this.fixedOffsetLeft = 0;
    this.fixedOffsetTop = 0;

    // scrolling at edge of window
    this.scrollTimer = null;
    this.xScroll = 0;
    this.yScroll = 0;
  }

  componentDidMount() {
    // set draggable attribute 'false' on any images, to prevent conflicts w browser
native dragging
    const imgs = this.containerElem.getElementsByTagName('IMG');
    for (let i = 0; i < imgs.length; i += 1) {
      imgs[i].setAttribute('draggable', 'false');
    }

    // capture events
    if (this.props.dragHandleClassName) {
      // if drag handles
      const elems =
this.containerElem.getElementsByClassName(this.props.dragHandleClassName);
      for (let i = 0; i < elems.length; i += 1) {
        this.addListeners(elems[i]);
        elems[i].style.cursor = 'move';
      }
    } else {
      // ... or not
      this.addListeners(this.containerElem);
      this.containerElem.style.cursor = 'move';
    }
  }

  componentWillUnmount() {
    this._isMounted = false;
  }

  addListeners = (elem) => {
    elem.addEventListener('mousedown', (e) => { this.handleMouseDown(e); }, false);
    elem.addEventListener('touchstart', (e) => { this.handleTouchStart(e); }, false);
    // must add touchmove listener here in order for preventDefault() to work, to
prevent scrolling during drag..
    elem.addEventListener('touchmove', this.handleTouchMove, { passive: false });
    elem.addEventListener('touchend', this.handleTouchEnd);
  };

  buildCustomEvent = (eventName, extraData = {}) => {
    let e;
    if (typeof window.CustomEvent !== 'function') {
      // we are in IE 11 and must use old-style method of creating event
      e = document.createEvent('CustomEvent');
      e.initCustomEvent(eventName, true, true, {});
    } else {
      e = new CustomEvent(eventName, { bubbles: true, cancelable: true });
    }
    // Add useful data to the event
    Object.assign(e, {
      dragData: this.props.dragData,
      dragElem: this.dragElem,
      containerElem: this.containerElem,
      sourceElem: this.sourceElem,
    }, extraData);
    return e;
  };

  setCurrentTarget = (x, y) => {
    // drop the z-index to get this elem out of the way, figure out what we're dragging
over, then reset the z index
    this.dragElem.style.zIndex = -1;
    const target = document.elementFromPoint(x, y) || document.body;
    this.dragElem.style.zIndex = this.props.zIndex;
    // prevent it from selecting itself as the target
    this.currentTarget = this.dragElem.contains(target) ? document.body : target;
  };

  setFixedOffset = () => {
    if (isZoomed()){
      [this.fixedOffsetLeft, this.fixedOffsetTop] = getFixedOffset();
    }
  };

  doScroll = () => {
    window.scrollBy(this.xScroll, this.yScroll)
    this.setFixedOffset();
  };

  startScrolling = (x, y) => {
    [this.xScroll, this.yScroll] = [x,y];
    if(!this.scrollTimer){
      this.scrollTimer = setInterval(this.doScroll, 50);
    }
  };

  stopScrolling = () => {
    clearInterval(this.scrollTimer);
    this.scrollTimer = null;
  };

  generateEnterLeaveEvents = (x, y) => {
    // generate events as we enter and leave elements while dragging
    const prefix = this.props.targetKey;
    this.setCurrentTarget(x, y);
    if (this.currentTarget !== this.prevTarget) {
      if (this.prevTarget) {
this.prevTarget.dispatchEvent(this.buildCustomEvent(`${prefix}DragLeave`)); }
      if (this.currentTarget) {
this.currentTarget.dispatchEvent(this.buildCustomEvent(`${prefix}DragEnter`)); }
    }
    this.prevTarget = this.currentTarget;
  };

  generateDropEvent = (x, y) => {
    // generate a drop event in whatever we're currently dragging over
    this.setCurrentTarget(x, y);
    const customEvent = this.buildCustomEvent(`${this.props.targetKey}Drop`, { x, y });
    this.currentTarget.dispatchEvent(customEvent);
  };

  // Start the Drag
  handleMouseDown = (e) => {
    if (usesLeftButton(e) && !this.props.noDragging) {
      document.addEventListener('mousemove', this.handleMouseMove);
      document.addEventListener('mouseup', this.handleMouseUp);
      this.startDrag(e.clientX, e.clientY);
    }
  };
  handleTouchStart = (e) => {
    if (!this.props.noDragging) {
      e.stopPropagation();
      this.setFixedOffset();
      this.startDrag(e.targetTouches[0].clientX, e.targetTouches[0].clientY);
    }
  };

  startDrag = (clickX, clickY) => {
    navigator.vibrate(60);
    document.addEventListener(`${this.props.targetKey}Dropped`, this.props.onDrop);
    const rect = this.containerElem.getBoundingClientRect();
    this.setState({
      clicked: true,
      leftOffset: rect.left - clickX,
      topOffset: rect.top - clickY,
      left: rect.left,
      top: rect.top,
    });
    this.props.onDragStart(this.props.dragData);
  };

  // During Drag
  handleMouseMove = (e) => {
    if (!this.props.noDragging) {
      e.preventDefault();
      if (this.state.clicked) {
        this.drag(e.clientX, e.clientY);
        window.getSelection().removeAllRanges(); // prevent firefox native-drag issue
when image is highlighted
      }
    }
  };

  handleTouchMove = (e) => {
    if (!this.props.noDragging) {
      e.preventDefault();  // prevents window scrolling
      if (this.state.clicked) {
        this.drag(e.targetTouches[0].clientX, e.targetTouches[0].clientY);
      }
    }
  };

  getOffscreenCoordinates = (x, y) => {
    // are we offscreen (or very close, anyway)? if so by how much?
    const LEFTEDGE = 10
    const RIGHTEDGE = window.innerWidth - 10
    const TOPEDGE = 10
    const BOTTOMEDGE = window.innerHeight - 10
    const xOff = x < LEFTEDGE ? x - LEFTEDGE : x > RIGHTEDGE ? x - RIGHTEDGE : 0;
    const yOff = y < TOPEDGE ? y - TOPEDGE : y > BOTTOMEDGE? y - BOTTOMEDGE : 0;
    return yOff || xOff ? [xOff, yOff] : false;
  };

  drag = (x, y) => {
    this.generateEnterLeaveEvents(x, y);
    const stateChanges = { dragging: true };
    const offScreen = this.getOffscreenCoordinates(x, y);
    if (offScreen) {
      this.startScrolling(...offScreen)
    } else {
      this.stopScrolling();
      if (!this.props.yOnly) { stateChanges.left = (this.state.leftOffset + x) -
this.fixedOffsetLeft; }
      if (!this.props.xOnly) { stateChanges.top = (this.state.topOffset + y) -
this.fixedOffsetTop; }
    }
    this.setState(stateChanges);
    this.props.onDrag(this.props.dragData, this.currentTarget, x, y);
  };

  // Drop
  handleMouseUp = (e) => {
    this.setState({ clicked: false });
    if (this.state.dragging) {
      document.removeEventListener('mousemove', this.handleMouseMove);
      document.removeEventListener('mouseup', this.handleMouseUp);
      this.drop(e.clientX, e.clientY);
      window.getSelection().removeAllRanges(); // prevent weird-looking highlights
    }
  };

  handleTouchEnd = (e) => {
    this.setState({ clicked: false });
    if (this.state.dragging) {
      this.drop(e.changedTouches[0].clientX, e.changedTouches[0].clientY);
    }
  };

  drop = (x, y) => {
    this.stopScrolling();
    this.generateDropEvent(x, y);
    document.removeEventListener(`${this.props.targetKey}Dropped`, this.props.onDrop);
    this._isMounted && this.setState({ dragging: false });
    this.props.onDragEnd(this.props.dragData, this.currentTarget, x, y);
  };

  getDisplayMode = () => {
    if (this.state.dragging && !this.props.dragClone && !this.props.customDragElement)
{
      if (this.props.disappearDraggedElement) {
        return 'disappeared'
      }
      return 'hidden'
    }
    return 'normal'
  };

  render() {
    const content = this.props.render ? this.props.render(this.state) :
this.props.children;
    const displayMode = this.getDisplayMode();

    // dragging will be applied to the "ghost" element
    let ghostContent;
    if (this.props.customDragElement) {
      ghostContent = this.props.customDragElement;
    } else {
      ghostContent = content;   // dragging a clone
    }

    const ghostStyles = {
      position: 'fixed',
      cursor: 'move',
      left: this.state.left,
      top: this.state.top,
      zIndex: this.props.zIndex,
      opacity: this.props.dragElemOpacity,
      display: this.state.dragging ? 'block' : 'none',
    };

    const ghost = (
      <div className="ddcontainerghost" style={ghostStyles} ref={(c) => { this.dragElem
= c; }}>
        {ghostContent}
      </div>
    );

    const containerStyles = {
      position: displayMode === 'disappeared' ? 'absolute' : 'relative',
      display: 'inline-block',
    };
    const sourceElemStyles = {
      display: displayMode === 'disappeared' ? 'none' : 'inherit',
      visibility: displayMode === 'hidden' ? 'hidden' : 'inherit',
    };

    return (
      <div className="ddcontainer" style={containerStyles} ref={(c) => {
this.containerElem = c; }}>
        <span className="ddcontainersource" style={sourceElemStyles} ref={(c) => {
this.sourceElem = c; }}>
          {content}
        </span>
        {ghost}
      </div>
    );
  }
}

DragContainer.propTypes = {
  children: PropTypes.node,

  // Determines what you can drop on
  targetKey: PropTypes.string,

  // If provided, we'll drag this instead of the actual object. Takes priority over
dragClone if both are set
  customDragElement: PropTypes.oneOfType([PropTypes.string, PropTypes.node]),

  // Makes the dragged element completely disappear while dragging so that it takes up
no space
  disappearDraggedElement: PropTypes.bool,

  // If true, then we will drag a clone of the object instead of the object itself. See
also customDragElement
  dragClone: PropTypes.bool,

  // ghost will display with this opacity
  dragElemOpacity: PropTypes.number,

  // We will pass this data to the target when you drag or drop over it
  dragData: PropTypes.object,

  // If included, we'll only let you drag by grabbing elements with this className
  dragHandleClassName: PropTypes.string,

  // if True, then dragging is turned off
  noDragging: PropTypes.bool,
  // callbacks (optional):
  onDrop: PropTypes.func,
  onDrag: PropTypes.func,
  onDragEnd: PropTypes.func,
  onDragStart: PropTypes.func,

  // Enable a render prop
  render: PropTypes.func,

  // Constrain dragging to the x or y directions only
  xOnly: PropTypes.bool,
  yOnly: PropTypes.bool,

  // Defaults to 1000 while dragging, but you can customize it if you need it to go
higher
  zIndex: PropTypes.number,
};

DragContainer.defaultProps = {
  targetKey: 'ddc',
  children: null,
  customDragElement: null,
  disappearDraggedElement: false,
  dragClone: false,
  dragElemOpacity: 0.9,
  dragData: {},
  dragHandleClassName: '',
  onDragStart: () => {},
  onDrag: () => {},
  onDragEnd: () => {},
  onDrop: () => {},
  noDragging: false,
  render: null,
  xOnly: false,
  yOnly: false,
  zIndex: 1000,
};

export default DragContainer;
```

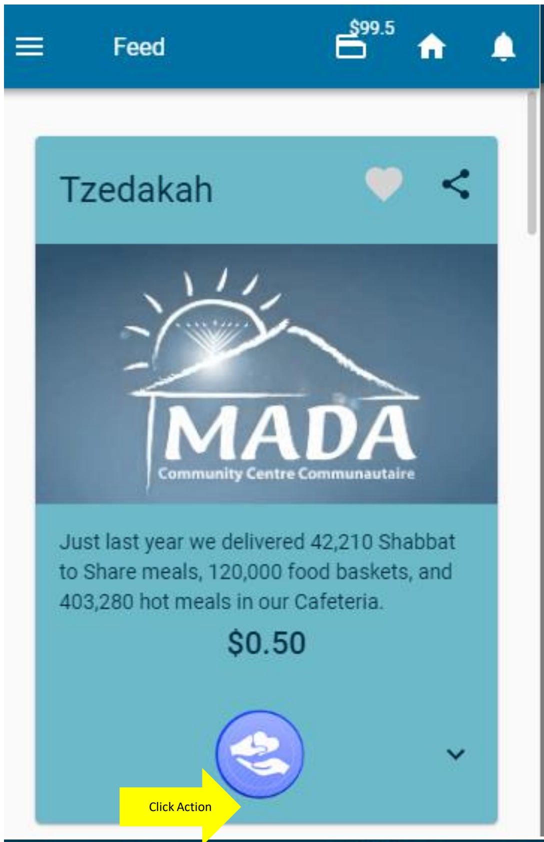
DropElement.js

```
import React from 'react';
//import Sound from 'react-sound';
import ReactAudioPlayer from 'react-audio-player';
import DropContainer from './DropContainer';

//import Button from '@material-ui/core/Button';


/*
    DropElement is set up as a drop target. Required props are targetKey and name.
*/
export default class DropElement extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
           // donateimage: "/images/pushkacanehappy.gif",
            thankYouMessage: '',
            src: this.props.src,
            onChangeTotalDonate: this.props.onChangeTotalDonate,
            totalDonate: this.props.totalDonate,

        };
    }

    //componentWillUnmount() {
    //    clearTimeout(this.timer);
    //}

    dropped = (e) => {
        //e.containerElem.style.visibility="hidden"; // This command hide the element
dropped
        this.setState({src: "/images/pushkacanehappy.gif"});
        this.setState({thankYouMessage: `Thanks !!!`})
        navigator.vibrate(100);
        this.props.onChangeTotalDonate(parseFloat(`${this.props.totalDonate}`) +
parseFloat(`${e.dragData.value}`));
        setTimeout(() => this.setState({src: "/images/pushkacane.gif"}), 5000);
        setTimeout(() => this.setState({thankYouMessage: ``}), 5000);
        console.log({'onHit event passed to target DropElement:':e});
        this.play();
    };

    play = () => {
        return (
            <ReactAudioPlayer
            src="/images/Yeah.mp3"
            autoPlay
            controls
            />
        );
    }
    render() {
        return (
        <DropContainer
            onHit={this.dropped}
            targetKey={this.props.targetKey}
            dropData={{name: this.props.name}}
            onChangeTotalDonate={this.props.onChangeTotalDonate}
            totalDonate={this.props.totalDonate}
            play={this.play}
        >
            <div className='dropElement'>
                <div style={{minHeight: 24, fontStyle: 'italic'}}>
                    {this.state.thankYouMessage}
                </div>
                <img src={this.state.src} alt="Donate" width="100"/>
                <p>${this.props.totalDonate}</p>
            </div>
        </DropContainer>
        );
    }
}
```

DropContainer.js

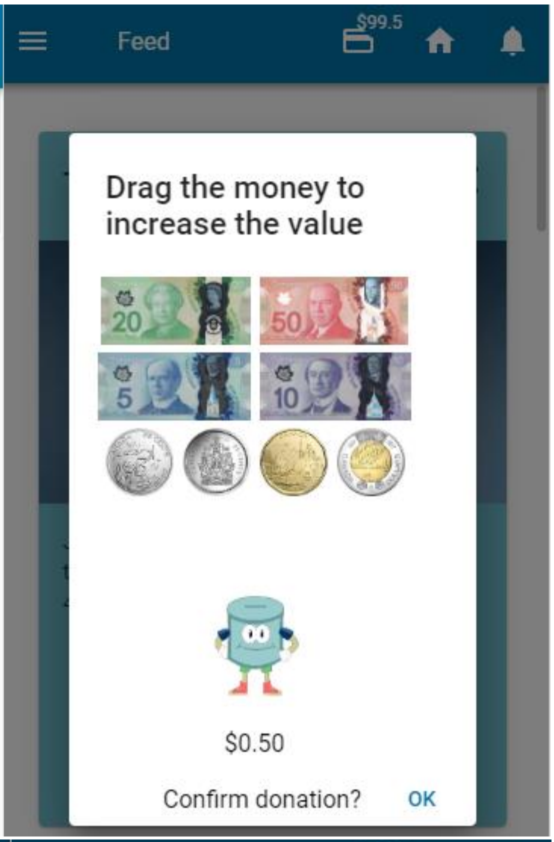
```
import React from 'react';
import PropTypes from 'prop-types';


class DropContainer extends React.Component {
  constructor(props) {
    super(props);
    this.elem = null;
    this.state = {highlighted: false};
  }

  componentDidMount() {
    this.elem.addEventListener(`${this.props.targetKey}DragEnter`,
this.handleDragEnter, false);
    this.elem.addEventListener(`${this.props.targetKey}DragLeave`,
this.handleDragLeave, false);
    this.elem.addEventListener(`${this.props.targetKey}Drop`, this.handleDrop, false);
  }

  createEvent(eventName, eventData) {
    // utility to create an event
    let e;
    if (typeof window.CustomEvent !== 'function') {
      // we are in IE 11 and must use old-style method of creating event
      e = document.createEvent('CustomEvent');
      e.initCustomEvent(eventName, true, true, {});
    } else {
      e = new CustomEvent(eventName, { bubbles: true, cancelable: true });
    }
    Object.assign(e, eventData);
    return e;
  }

  handleDrop = (e) => {
    // tell the drop source about the drop, then do the callback
    const evt = this.createEvent(
      `${this.props.targetKey}Dropped`,
      {
        dragData: e.dragData,
        dropElem: this.elem,
        dropData: this.props.dropData,
      },
    );
    e.containerElem.dispatchEvent(evt);
    this.props.onHit(e);
    this.setState({highlighted: false})
  }

  handleDragEnter = (e) => {
    console.log('enter')
    const _e = e;
    this.props.highlightClassName && this.setState({highlighted: true})
    this.props.onDragEnter(_e);
  }

  handleDragLeave = (e) => {
    const _e = e;
    this.props.highlightClassName && this.setState({highlighted: false})
    this.props.onDragLeave(_e);
  }


  render() {
    const classNames = 'DropContainer ' +  (this.state.highlighted ?
this.props.highlightClassName : '');
    return (
      <span ref={(t) => { this.elem = t; }} className={classNames}>
        {this.props.render ? this.props.render() : this.props.children}
      </span>
    );
  }
}

DropContainer.propTypes = {
  children: PropTypes.node,
  render: PropTypes.func,
  highlightClassName: PropTypes.string,

  // needs to match the targetKey in the DragContainer -- matched via the
enter/leave/drop event names, above
  targetKey: PropTypes.string,

  // data that gets sent back to the DragContainer and shows up in its onDrop()
callback event
  dropData: PropTypes.object,

  // callbacks
  onDragEnter: PropTypes.func,
  onDragLeave: PropTypes.func,
  onHit: PropTypes.func,
};

DropContainer.defaultProps = {
  children: null,
  targetKey: 'ddc',
  onDragEnter: () => {},
  onDragLeave: () => {},
  onHit: () => () => {},
  dropData: {},
  highlightClassName: 'highlighted',
  render: null,
};

export default DropContainer;
```

9) Reports



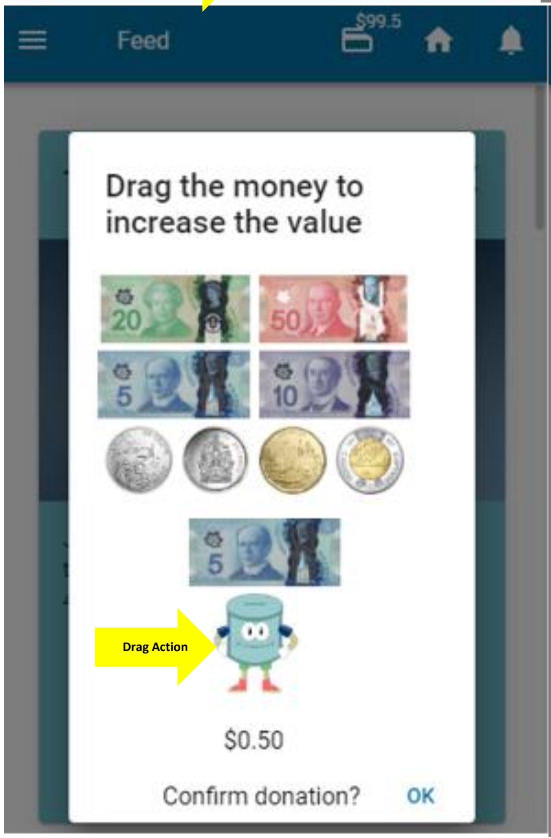
Just last year we delivered 42,210 Shabbat to Share meals, 120,000 food baskets, and 403,280 hot meals in our Cafeteria.

$0.50



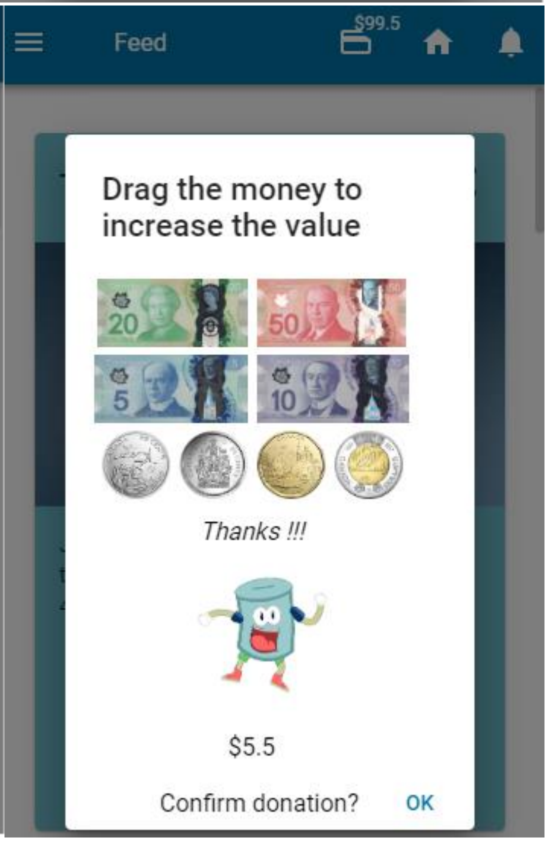
Drag the money to increase the value

$0.50



Drag the money to increase the value

$0.50



Drag the money to increase the value

Thanks !!!

$5.5

10)Technical Support Offered

1ª part of Project (Develop a Donation interactive module for app): The project proposal was accepted by the app's responsible manager and the project was implemented.

2ª part of Project (Develop a Recreational Activity interactive module for app): The project proposal was accepted by the app's responsible manager but its implementation was delayed for the app's second phase of development.

3ª part of Project (Update and support for Mada Website): Despite the constant demand, this part of the project does not require a very high technical knowledge since the Website was developed in WordPress.

11)Challenges

It was my first contact with the React library. So the challenge was great, in a short time understand how an application is structured in React and help in its implementation. Despite the difficulties, the result was rewarding.

12)Conclusion

I was fortunate enough to do the internship right in the implementation period of the tzedakah-pushke mobile application (https://tzedakah-pushke.madacenter.com). This application together with the Mada Community Center website (https://madacenter.com), aims to publicize the Institution. This way I had the opportunity to collaborate and learn a lot with this implementation.

In the first part of the project was where I spent most of the internship time. I had to know the structure of the app, know the resources used to then really begin the development of the project. And the best, the project was accepted by the responsible manager of the app and it was implemented.

In the second part of the project the objective was to present a solution to make the app more attractive because the focus of the app is to also promote a children's audience and have an educational proposal. The idea was to incorporate Unity development in the app that was developed in React. The alternative has been tested and approved but this will be implemented in the second phase of the project.

And finally the third phase of the project was to carry out the daily activities of the IT department as support and update the Mada website. In general the internship was very useful and I felt very gratifying to participate in the professional development of an application. I would like to thank everyone who gave me this opportunity.