Lab 6 : Core APIs

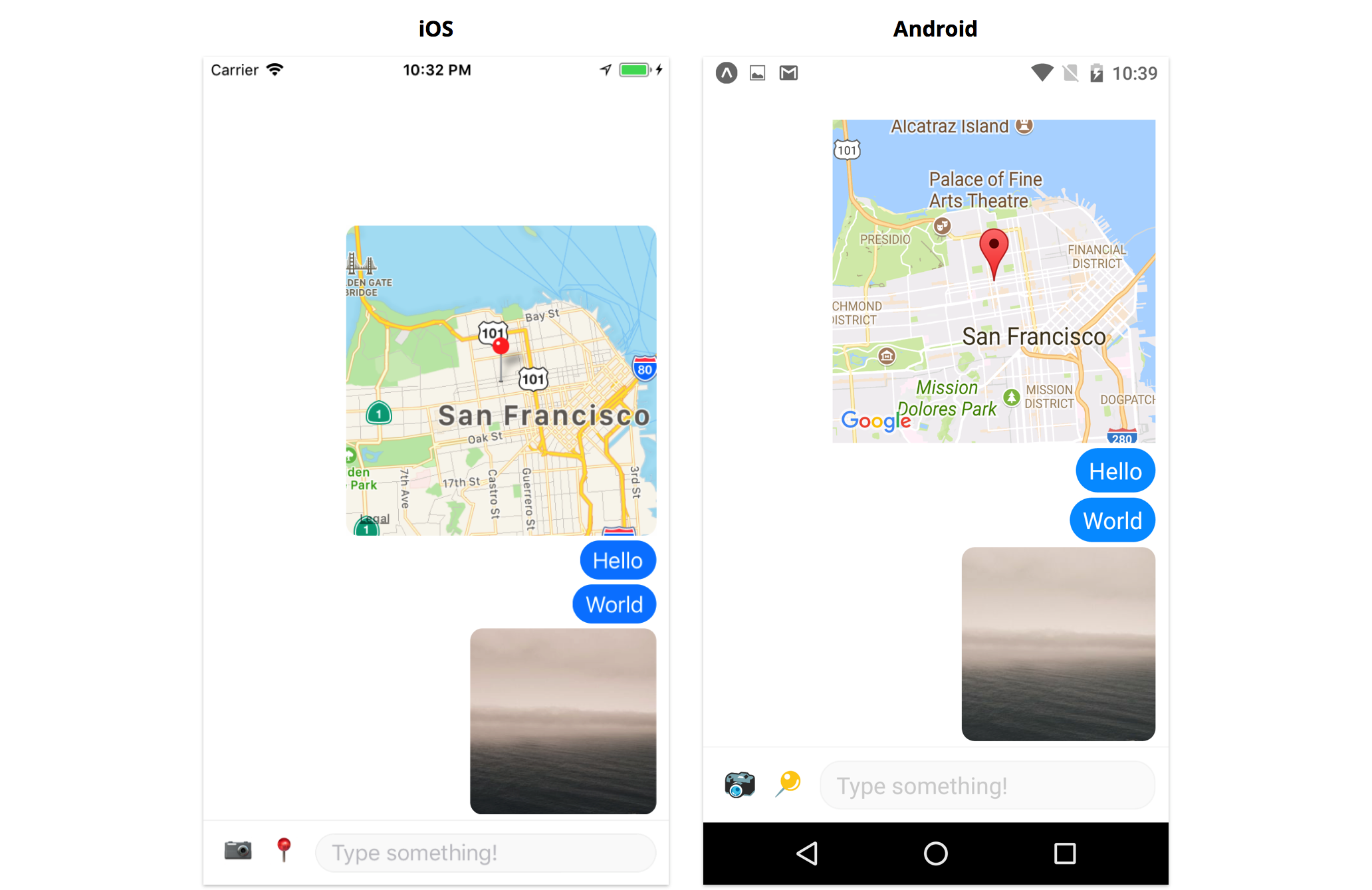
1. ให้นักศึกษาทำการสร้าง New Project โดยให้ Folder มีดังนี้ Mobile\<รหัสนักศึกษา>\Message

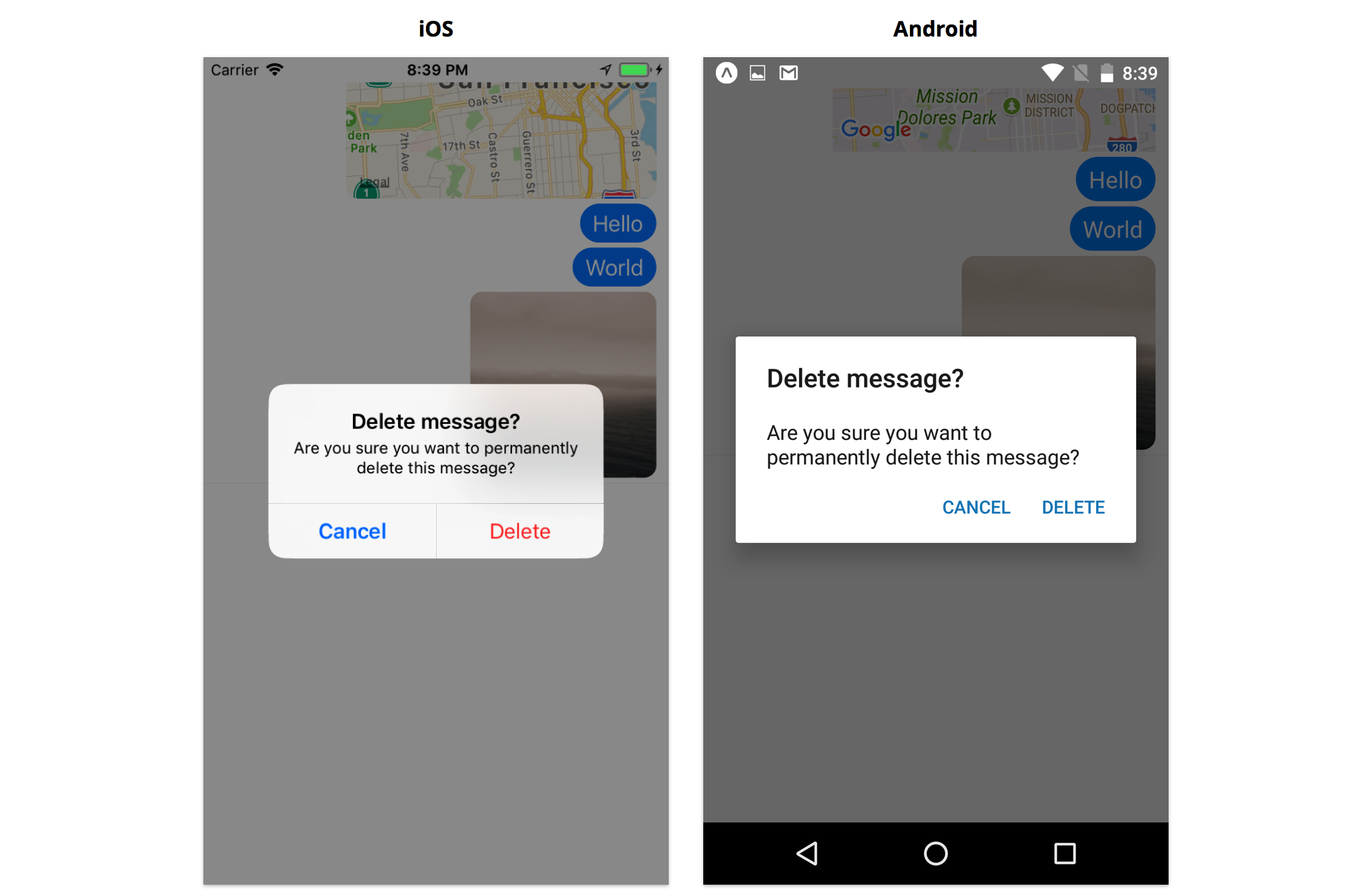
Expo init <path\folder\StudentID\Project name>

1. ให้นักศึกษาเขียนโปรแกรมเพื่อรับส่งข้อความและรูปภาพ (ดังแสดงตามรูปภาพข้างล่างนี้) โดยการปฏิบัติครั้งนี้ให้นักศึกษาทำการสร้าง NetInfo, StatusBar, MessageList, Alert

Hint: (Component) View, Text, Image

(APIs) NetInfo, MessageList, Statusbar, Alert





**Source Code**

**App.js**

import { StyleSheet, View } from 'react-native';

import React from 'react';

import Status from './components/Status';

import MessageList from './components/MessageList'

import { createTextMessage, createImageMessage, createLocationMessage } from './utils/MessageUtils';

export default class App extends React.Component {

  renderMessageList() {

    let mList = this.createMessageList()

    let iList = [createImageMessage("https://pbs.twimg.com/media/Ehe-7nlU4AIqhUU.jpg")]

    let lList = [createLocationMessage({

      latitude: 37.78825,

      longitude: -122.4324

    })]

return (

      <View style={styles.content}>

        <MessageList messages={mList}/>

        <MessageList messages={iList}/>

        <MessageList messages={lList}/>

      </View>);

  }

  renderInputMethodEditor() {

    return (

      <View style={styles.inputMethodEditor}></View>);

  }

  renderToolbar() {

    return (

      <View style={styles.toolbar}></View>);

  }

  createMessageList(){

    let message = ['Hello', 'World']

    let mList = []

    for(let i = 0; i < message.length; i++){

      mList.push(createTextMessage(message[i]))

    }

    return mList

  }

  render() {

    return (

      <View style={styles.container}>

      <Status />

      {this.renderMessageList()}

      {/\* {this.renderToolbar()}

      {this.renderInputMethodEditor()} \*/}

      {/\* ตรงส่วนนี้ปิดไว้เพราะยังไม่ได้ใช้งานในสัปดาห์นี้ และทำให้หน้าจอแสดงผลได้ไม่สวยงามครับ \*/}

      </View>

    );

  }

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: 'white',

  },

content: {

    flex: 1,

    backgroundColor: 'white',

  },

  inputMethodEditor: {

    flex: 1,

    backgroundColor: 'white',

  },

  toolbar: {

    borderTopWidth: 1,

    borderTopColor: 'rgba(0,0,0,0.04)', backgroundColor: 'white',

  },

})

**Status.js**

import Constants from 'expo-constants';

import NetInfo from '@react-native-community/netinfo';

import { Platform, StatusBar, StyleSheet, Text, View, } from 'react-native';

import React from 'react';

const statusHeight = (Platform.OS === 'ios' ? Constants.statusBarHeight : 0);

const handler = (status) => {

    console.log('Network status changed', status);

};

const subscription = NetInfo.addEventListener(handler);

export default class Status extends React.Component {

    state = {

        isConnected: null,

    };

    async componentDidMount() {

        this.subscription =

            NetInfo.addEventListener(this.handleChange);

        const { isConnected } = await NetInfo.fetch();

        this.setState({ isConnected });

    }

componentWillUnmount() {

        this.subscription()

    }

    handleChange = ({ isConnected }) => {

        this.setState({ isConnected });

    };

    render() {

        const { isConnected } = this.state;

        const backgroundColor = isConnected ? 'white' : 'red';

        const statusBar = (

            <StatusBar backgroundColor={backgroundColor} barStyle={isConnected ? 'dark-content' : 'light-content'} animated={false} />

        );

        const messageContainer = (

            <View style={styles.messageContainer} pointerEvents={'none'}>

                {statusBar}

                {!isConnected && (

                    <View style={styles.bubble}>

                        <Text style={styles.text}>No network connection</Text>

                    </View>

                )}

            </View>

        );

        if (Platform.OS === 'ios') {

            return (

                <View style={[styles.status, { backgroundColor }]}>

                    {messageContainer}

                </View>

            );

        }

        return messageContainer;

    }

}

const styles = StyleSheet.create({

    status: {

        zIndex: 1,

        height: statusHeight,

    },

    messageContainer: {

        zIndex: 1,

        position: 'absolute',

        top: statusHeight + 20,

        right: 0,

        left: 0,

        height: 80,

        alignItems: 'center',

    },

    bubble: {

        paddingHorizontal: 20,

        paddingVertical: 10,

        borderRadius: 20,

        backgroundColor: 'red',

    },

    text: {

        color: 'white',

    }

});

**MessageList.js**

import React from 'react';

import PropTypes from 'prop-types';

import { StyleSheet, View, Text, FlatList, TouchableOpacity, Alert, Image } from 'react-native';

import { MessageShape } from '../utils/MessageUtils';

import MapView from 'react-native-maps';

const keyExtractor = item => item.id.toString();

export default class MessageList extends React.Component {

    static propTypes = {

        messages: PropTypes.number.isRequired,

        onPressMessage: PropTypes.number.isRequired,

    };

    state = {

        messages: this.props.messages,

    }

deleteAlert = (id) =>

        Alert.alert(

            "Delete message?",

            "Are you sure you want to permanently delete this message?",

            [

                {

                    text: "CANCEL",

                    onPress: () => console.log("Cancel Pressed"),

                    style: "cancel"

                },

                {

                    text: "DELETE", onPress: () => {

                        this.setState({

                            messages: this.state.messages.filter(obj => obj.id != id)

                        })

                    }

                }

            ],

            { cancelable: false }

        );

    render() {

        const TextMessage = ({ text }) => {

            return (

                <View style={styles.messageBubble}>

                    <Text style={styles.textMessage}>{text}</Text>

                </View>

            );

        }

        const ImageMessage = ({ uri }) => (

            <Image

                style={styles.imageMessage}

                source={{

                    uri: uri

                }}

            />

        );

const MapMessage = ({ coordinate }) => (

            <Image

                style={styles.mapMessage}

                source={require("../assets/map.jpg")}

            />

            /\* เนื่องจากโจทย์ในเอกสารประกอบแลป ระบุว่าให้เขียนโปรแกรมเพื่อรับส่งข้อความและรูปภาพ จึงเข้าใจได้ว่าส่วนของแผนที่ไว้ทำในแลปชั่วโมงหน้าครับ

            ตอนนี้ผมเลยเอารูปแผนที่ที่ได้เตรียมไว้มาใส่แทนที่ไว้ก่อนครับ\*/

        );

const renderItem = ({ item }) => {

            let itemBlock;

            if (item.type === 'text') {

                itemBlock = (<TextMessage text={item.text} />)

            }

            else if (item.type === 'image') {

                itemBlock = (<ImageMessage uri={item.uri} />)

            } else {

                itemBlock = (<MapMessage coordinate={item.coordinate} />)

            }

            return (

                <TouchableOpacity onPress={() => { this.deleteAlert(item.id) }}>

                    <View style={styles.message}>

                        {itemBlock}

                    </View>

                </TouchableOpacity>

            )

        }

        return (

            <FlatList

                data={this.state.messages}

                renderItem={renderItem}

                keyExtractor={keyExtractor}

            />

        );

    }

}

const styles = StyleSheet.create({

    message: {

        marginTop: 8,

        marginRight: 10,

        marginLeft: 10,

        paddingHorizontal: 10,

        paddingVertical: 0,

    },

    imageMessage: {

        width: "50%",

        resizeMode: "contain",

        height: undefined,

        aspectRatio: 1,

        alignSelf: 'flex-end',

    },

textMessage: {

        borderRadius: 5,

        flexDirection: 'row',

        flex: 1,

        color: "white",

        justifyContent: 'center',

        alignSelf: 'center',

        fontSize: 20

    },

    messageBubble: {

        paddingVertical: 10,

        paddingHorizontal: 10,

        backgroundColor: 'rgb(16,135,255)',

        borderRadius: 20,

        width: '30%',

        justifyContent: 'center',

        alignSelf: 'flex-end'

    },

    mapMessage: {

        width: "80%",

        resizeMode: "contain",

        height: undefined,

        aspectRatio: 1,

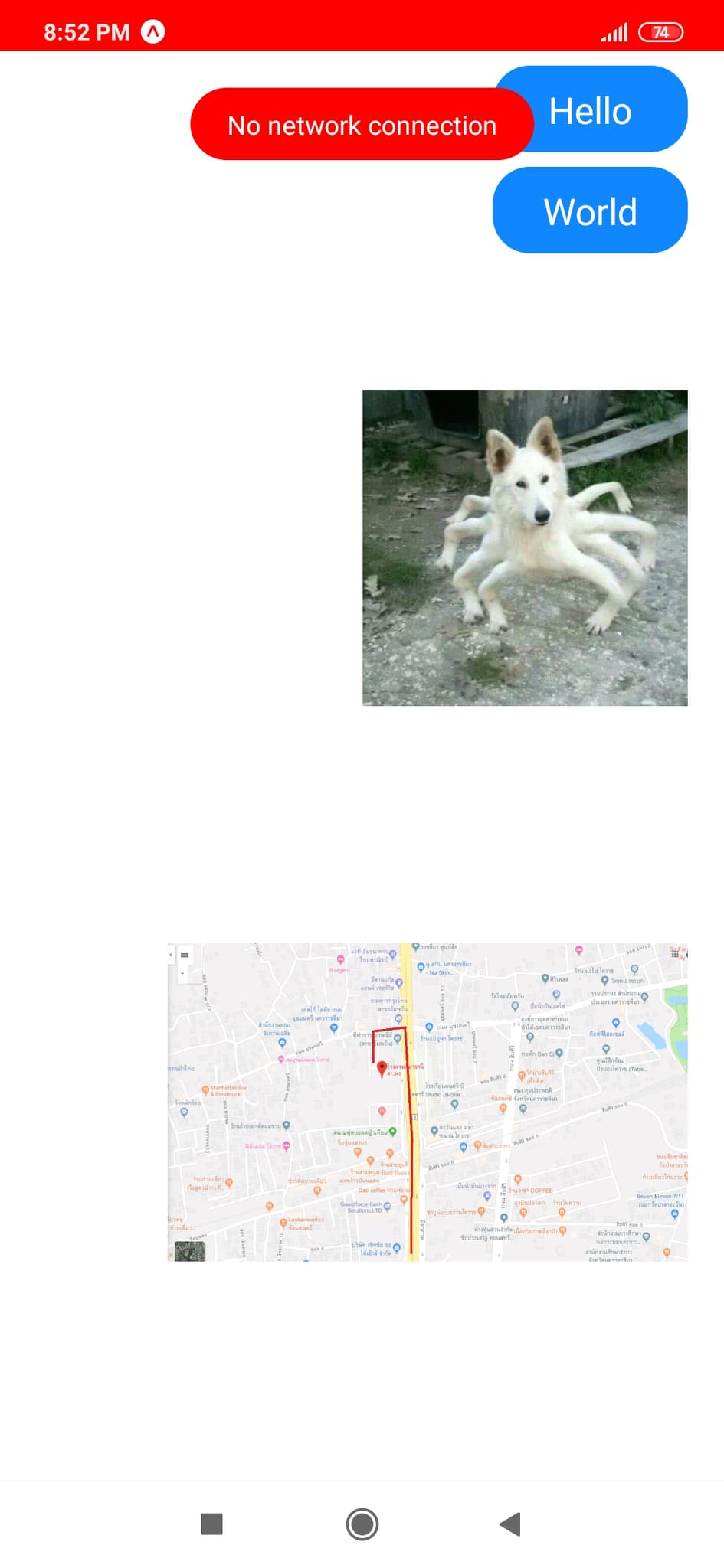
        alignSelf: 'flex-end',

    }

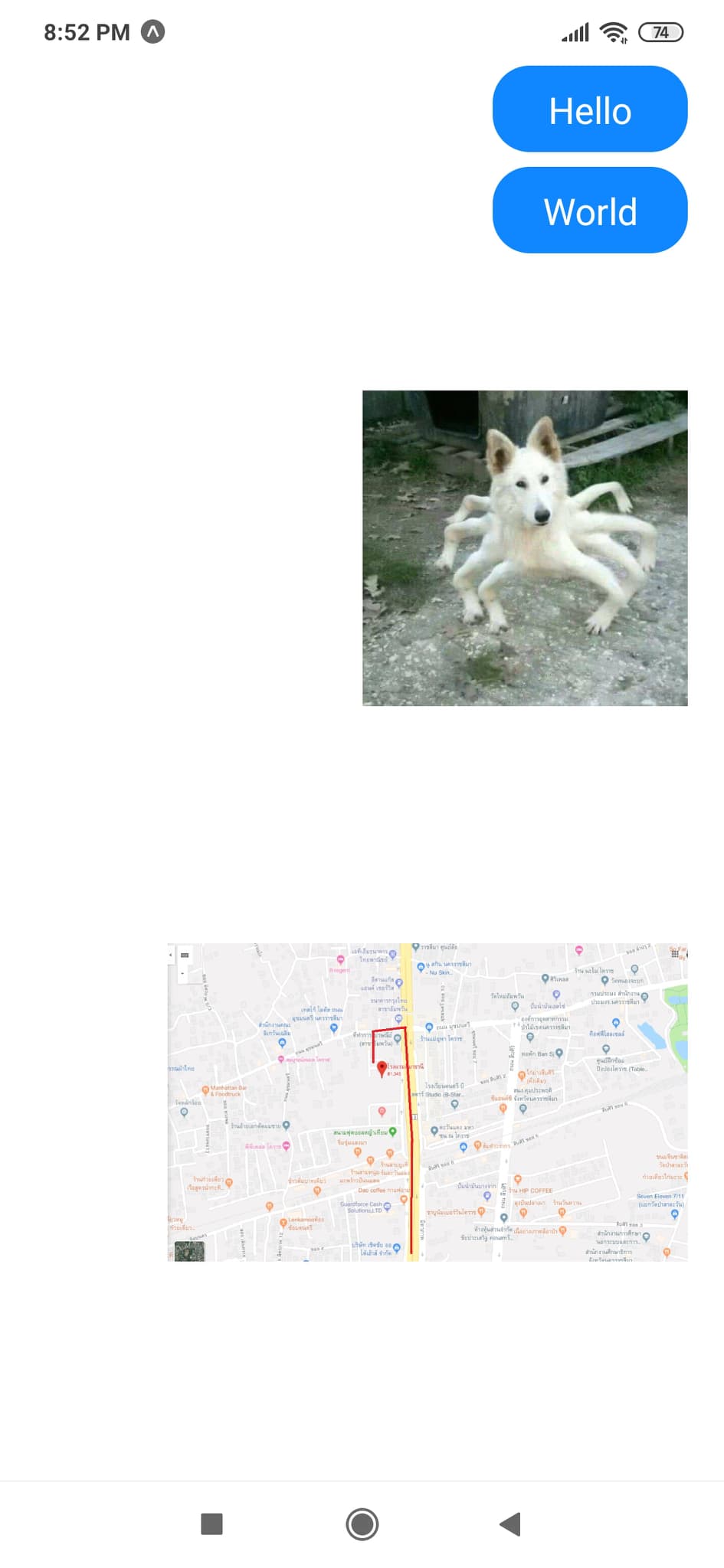
});

**ผลการทดลอง**

**กรณีที่ไม่มีการเชื่อมต่อกับอินเทอร์เน็ต**

****

**กรณีที่มีการเชื่อมต่ออินเทอร์เน็ต (รูปแบบปกติ)**

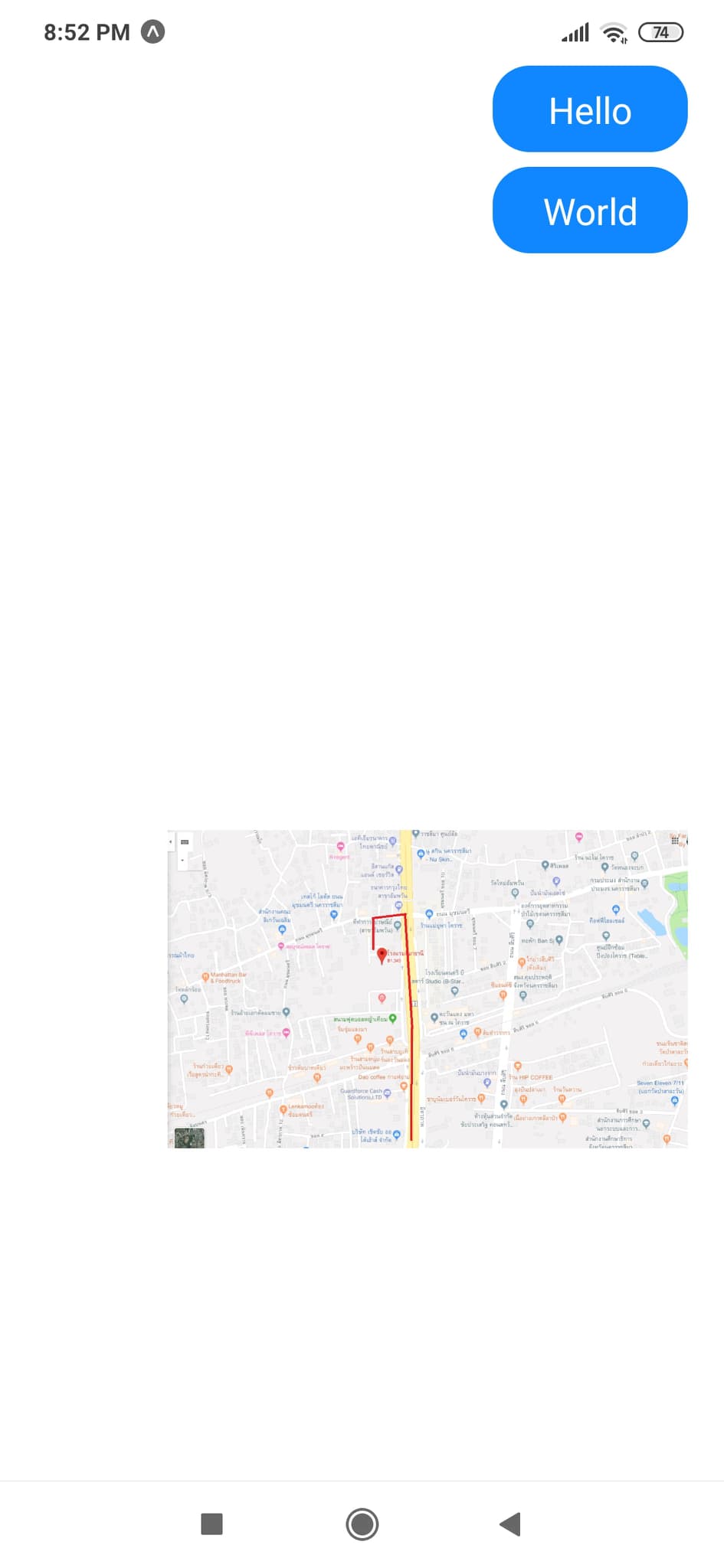
****

**กรณีที่กดที่ข้อความหรือรูปภาพที่ส่ง (ในกรณีนี้กดที่รูปภาพ)**

**รูปภาพประกอบด้วย ข้อความ

คำอธิบายที่สร้างโดยอัตโนมัติ**

จะมี Alert ที่ได้ตั้งค่าไว้ให้แจ้งเตือนขึ้นมาหลังจากที่กดที่ข้อความหรือรูปภาพที่ส่งขึ้นมา เพื่อยืนยันกับเราว่าต้องการจะลบข้อความหรือรูปภาพที่ส่งไปมั้ย

**กรณีที่กด DELETE ใน Alert ที่แจ้งเตือนขึ้นมาหลังจากกดที่ข้อความหรือรูปภาพที่ส่ง**

หลังจากกดที่ DELETE จะทำการลบข้อความหรือรูปภาพที่เราได้ส่งไป