



CS 458
Software Verification & Validation
2019-2020 Spring

PROJECT #2

Berk Yıldız 21502040

Table of Contents

1. Implementation of Survey App.....	3
a. Summary of Implementation.....	3
b. Source Code.....	4
c. Appearance of Application	7
i. Android.....	7
ii. iOS.....	8
2. Capabilities of Appium.....	9
3. Test Cases & Automation Codes	10
4. Evaluation of Automation Experience.....	14
5. Comparison of Appium and Selenium.....	15

1. Implementation of Survey App

a. Summary of Implementation

Survey App is implemented by React Native which is a JavaScript framework developed by Facebook. The main advantage of React Native is that it is supportable for both Android and iOS. It makes possible to deploy same source code for both operating systems. Thanks to React Native, application is running on both Android and iOS with the same source code.

The application consists of five inputs: *Full Name*, *Birthday*, *City*, *Gender* and *Occupation*. *Full Name*, *City* and *Occupation* areas are **TextInput** component of React Native. *Birthday* is a **Datepicker** component and *Gender* is a **Picker** component of React Native. All components that exist in source code are compatible for both Android and iOS.

Full Name and *City* inputs just accept letters, dot and space character as an input. Any input which contains digit or symbols is invalid.

Submit button is initially disabled and inactive. When the input areas are supplied with proper data, *Submit* button becomes active and touchable. If user enters an invalid data after entering proper data, *Submit* button is disabled again.

If everything goes well and the form is submitted, the user receives a message that the submission was completed successfully.

Source code also includes alert messages for invalid inputs of *Full Name* and *City* areas for debugging purposes. However these are not executed in the regular flow of application because of the inactiveness of *Submit* button.

Appearance of the app may differ in iOS and Android, but its functionality is exactly the same on both operating systems.

b. Source Code

```
import React, { Component } from 'react';
import DatePicker from 'react-native-datepicker';

import {
  StyleSheet,
  AppRegistry,
  TextInput,
  View,
  Alert,
  Button,
  Picker,
  Text
} from 'react-native';

export default class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      NameInput: '',
      BirthdayInput: '',
      CityInput: '',
      GenderInput: '',
      OccupationInput: '',
      warningMsg: 'Use only letters'
    };
  }

  validateInput = (input) => {
    var re = /^[A-Za-z. ]+$/;
    return re.test(input);
  };

  validateInput2 = (input) => {
    var re = /^[A-Za-z ]+$/;
    return re.test(input);
  };

  CheckTextInput = () => {
    if (!this.validateInput(this.state.NameInput) ||
        !this.validateInput2(this.state.CityInput)) {
      if
        (!this.validateInput(this.state.NameInput) &&
         !this.validateInput2(this.state.CityInput)) {
        alert('Input Error: Inappropriate
character use for Full Name and City!');
      }
      else if
        (!this.validateInput(this.state.NameInput)) {
        alert('Input Error: Inappropriate
character use for Full Name!');
      }
      else if
        (!this.validateInput2(this.state.CityInput)) {
        alert('Input Error: Inappropriate
```

```

        character use for City!');
    }
    }
    else {
        alert('We received your answers. Thank
you!');
    }
};
render() {
    return (
        <View style={styles.MainContainer}>
            <TextInput
                accessibilityLabel="nameInput"
                placeholder="Full Name"
                onChangeText={NameInput => this.setState({
NameInput })}
                underlineColorAndroid="transparent"
                style={styles.TextInput}
                maxLength={250}
            />
            <Text>
                {this.state.warningMsg}
            </Text>
            <DatePicker
                accessibilityLabel="birthdayPicker"
                style={styles.TextInput}
                date={this.state.BirthdayInput}
                mode="date"
                placeholder="Birthday"
                format="DD-MM-YYYY"
                minDate="01-01-1901"
                maxDate="01-01-2021"
                confirmBtnText="Confirm"
                cancelBtnText="Cancel"
                customStyles={{
                    dateIcon: {
                        position: 'absolute',
                        left: 0,
                        top: 4,
                        marginLeft: 0
                    },
                    dateInput: {
                        marginLeft: 36
                    }
                }}
                onChange={(BirthdayInput) =>
{this.setState({BirthdayInput: BirthdayInput})}}
            />
            <TextInput
                accessibilityLabel="cityInput"
                placeholder="City"
                onChangeText={CityInput => this.setState({
CityInput })}
                underlineColorAndroid="transparent"
                style={styles.TextInput}
                maxLength={100}

```

```

        />
        <Text>
        {this.state.warningMsg}
        </Text>
        <Picker
        accessibilityLabel="genderSelect"
        selectedValue={this.state.GenderInput}
        style={styles.TextInput}
        onValueChange={(itemValue, itemIndex) =>
        this.setState({GenderInput: itemValue})
        }>
        <Picker.Item label="Gender" value="" />
        <Picker.Item label="Male" value="Male" />
        <Picker.Item label="Female" value="Female" />
        <Picker.Item label="Other" value="Other" />
        </Picker>
        <TextInput
        accessibilityLabel="occupationInput"
        placeholder="Occupation"
        onChangeText={OccupationInput => this.setState({
OccupationInput })}
        underlineColorAndroid="transparent"
        style={styles.TextInput}
        />
        <View style={{ marginTop: 35 }}>
        <Button
        accessibilityLabel="submitButton"
        title="Submit Form"
        onPress={this.CheckTextInput}
        color="#606070"

disabled={!this.validateInput(this.state.NameInput) ||
this.state.BirthdayInput == '' ||
!this.validateInput(this.state.CityInput) ||
this.state.OccupationInput == '' || this.state.GenderInput
== '' } ? true : false}
        />
        </View>
        </View>
        );
    }
}
const styles = StyleSheet.create({
  MainContainer: {
    flex: 1,
    margin: 35,
  },

  TextInput: {
    width: '100%',
    height: 40,
    borderWidth: 1,
    marginTop: 15,
  },
});

```

c. Appearance of Application

i. Android

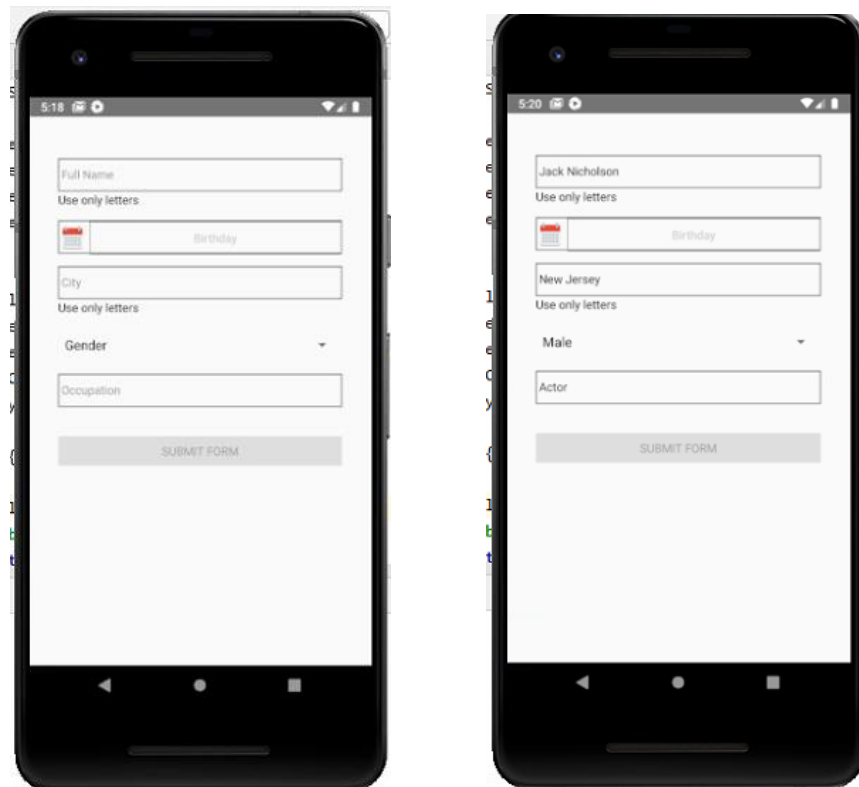


Figure 1: Android screens with inactive *Submit* button

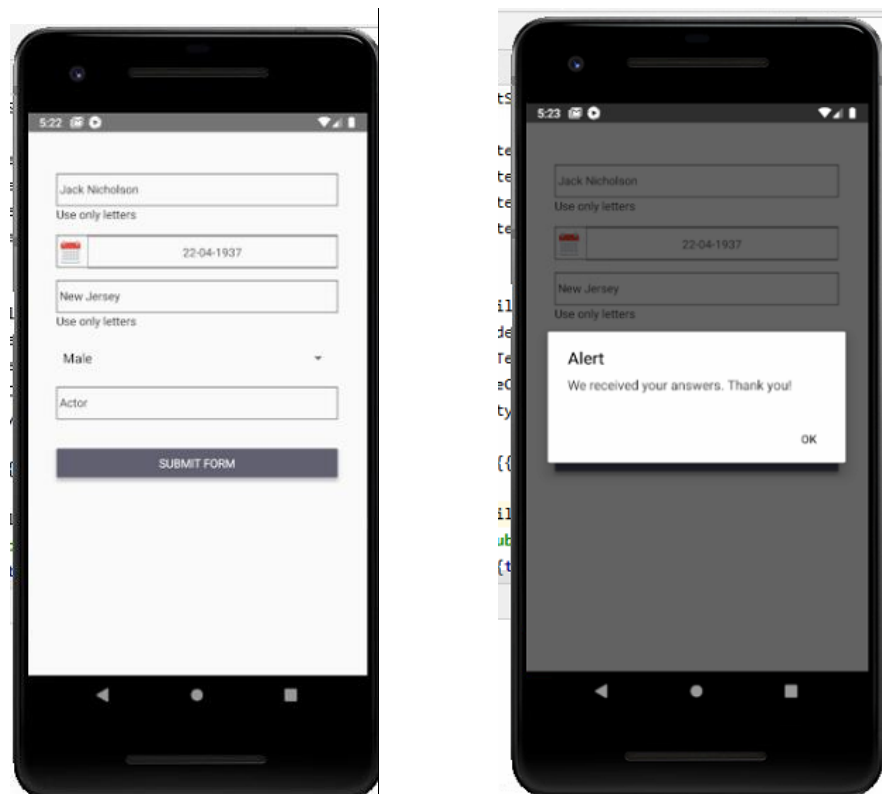


Figure 2: Active *Submit* button and successful submission

ii. iOS



Appetize.io 4:08 AM

Full Name
Use only letters

Birthday
28-02-2018

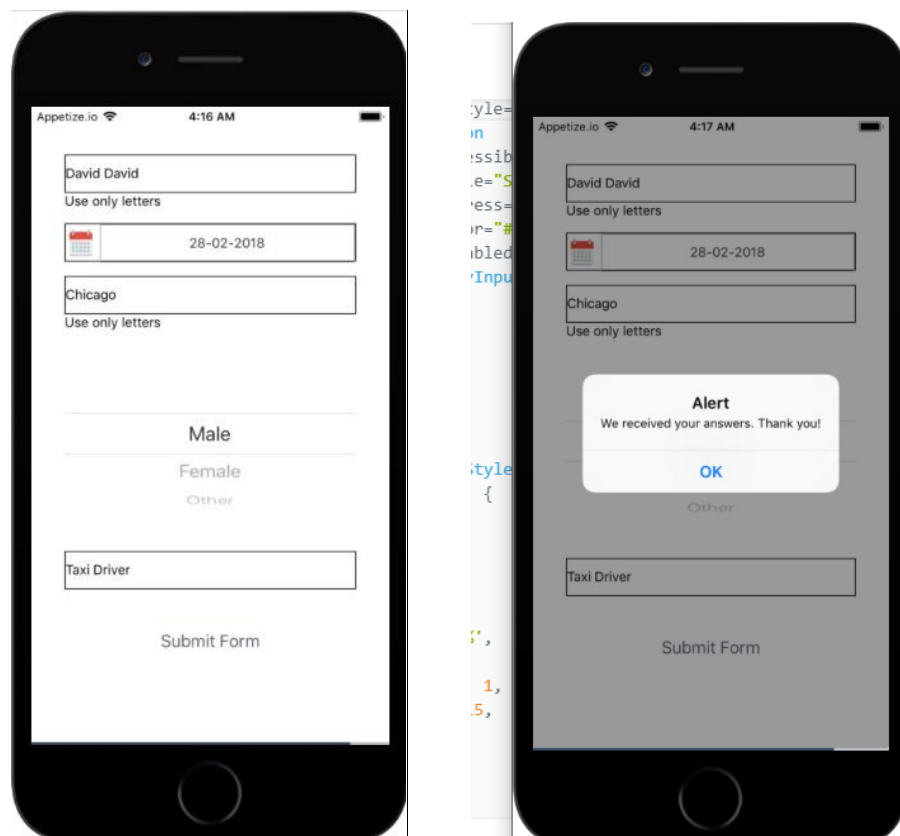
City
Use only letters

Chicago

Male
Female
Other

Submit Form

Figure 3: iOS screen with inactive *Submit* button



Appetize.io 4:16 AM

David David
Use only letters

28-02-2018

Chicago
Use only letters

Male
Female
Other

Taxi Driver

Submit Form

Alert
We received your answers. Thank you!
OK

Submit Form

Figure 4: Active *Submit* button and successful submission

2. Capabilities of Appium

- Appium can automate all mobile applications, regardless of its targeting operating system and which language it is written in.
- The existence of requested components on a mobile application can be verified by Appium.
- Giving correct error messages can be verified by Appium.
- It is possible to write automation in all dev tools by most of the languages like Java, PHP, Javascript, C# and more.
- Test cases for valid and invalid inputs for the elements of a webpage can be verified by Appium.
- There are several ways for selecting components on a mobile application by using accessibilityId, name, id, xpath.
- Appium is cross-platform which means it is possible to write automation code for Android, iOS and Windows with the same API.
- It supports several frameworks like NUnit, JUnit, TestNG etc.
- It provides huge number of methods to examine every possible scenario on a mobile phone.
- Setting the environment for automation is very simple for Appium.

3. Test Cases & Automation Codes

- ***Enter valid Full Name, Birthday, City, Gender & Occupation and click Submit Form button. Verify if user can submit the form and “We received your answers. Thank you!” message shows up.***

```
MobileElement element = (MobileElement)
driver.findElementByAccessibilityId("nameInput");

element.sendKeys("Jack Nicholson");

element = (MobileElement)
driver.findElementByAccessibilityId("birthdayPicker");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("ViewGroup(0)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("View(4)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("cityInput");

element.sendKeys("New Jersey");

element = (MobileElement)
driver.findElementByAccessibilityId("genderSelect");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("CheckedTextView(1)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("occupationInput");

element.sendKeys("Actor");

element = (MobileElement)
driver.findElementByAccessibilityId("submitButton");

element.click();

String message = driver.switchTo().alert().getText();

if(message.equals("We received your answers. Thank you!"))
System.out.println("Test for valid inputs is successful.");

else System.out.println("Test for valid inputs fails!");
```

- ***Leave Full Name blank and enter valid Birthday, City, Gender & Occupation. Verify if Submit Form button is disabled.***

```
element = (MobileElement)
driver.findElementByAccessibilityId("birthdayPicker");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("ViewGroup(0)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("View(4)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("cityInput");

element.sendKeys("New Jersey");

element = (MobileElement)
driver.findElementByAccessibilityId("genderSelect");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("CheckedTextView(1)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("occupationInput");

element.sendKeys("Actor");

element = (MobileElement)
driver.findElementByAccessibilityId("submitButton");

element.click();

if(element.isEnabled()) System.out.println("Submit Button is
enabled. Test fails!");

else System.out.println("Submit Button is still disabled. Test
passes.");
```

- ***Leave Gender blank and enter valid Full Name, Birthday, City & Occupation. Verify if Submit Form button is disabled.***

```
element = (MobileElement)
driver.findElementByAccessibilityId("nameInput");

element.sendKeys("Jack Nicholson");

element = (MobileElement)
driver.findElementByAccessibilityId("birthdayPicker");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("ViewGroup(0)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("View(4)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("cityInput");

element.sendKeys("New Jersey");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("occupationInput");

element.sendKeys("Actor");

element = (MobileElement)
driver.findElementByAccessibilityId("submitButton");

element.click();

if(element.isEnabled()) System.out.println("Submit Button
enabled. Test fails!");

else System.out.println("Submit Button is still disabled. Test
passes.");
```

- ***Enter invalid Full Name by having input with numbers and enter valid Birthday, City, Gender & Occupation. Verify if Submit Form button is disabled.***

```
element = (MobileElement)
driver.findElementByAccessibilityId("nameInput");

element.sendKeys("rebellion_33");

element = (MobileElement)
driver.findElementByAccessibilityId("birthdayPicker");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("ViewGroup(0)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("View(4)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("cityInput");

element.sendKeys("New Jersey");

element = (MobileElement)
driver.findElementByAccessibilityId("genderSelect");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("CheckedTextView(1)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("occupationInput");

element.sendKeys("Actor");

element = (MobileElement)
driver.findElementByAccessibilityId("submitButton");

element.click();

if(element.isEnabled()) System.out.println("Submit Button
enabled. Test fails!");

else System.out.println("Submit Button is still disabled. Test
passes.");
```

- ***Do not choose Birthday and enter valid Full Name, City, Gender & Occupation.***

Verify if Submit Form button is disabled.

```

element = (MobileElement)
driver.findElementByAccessibilityId("nameInput");

element.sendKeys("Jack Nicholson");

element = (MobileElement)
driver.findElementByAccessibilityId("cityInput");

element.sendKeys("New Jersey");

element = (MobileElement)
driver.findElementByAccessibilityId("genderSelect");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("CheckedTextView(1)");

element.click();

element = (MobileElement)
driver.findElementByAccessibilityId("occupationInput");

element.sendKeys("Actor");

element = (MobileElement)
driver.findElementByAccessibilityId("submitButton");

element.click();

if(element.isEnabled()) System.out.println("Submit Button
enabled. Test fails!");

else System.out.println("Submit Button is still disabled. Test
passes.");

```

4. Evaluation of Automation Experience

Before I started the project, I had no idea about implementation of mobile automation, even though I knew about it as a concept. After finishing the first project, I had an idea of the coding side of automation, but I could not predict what differences mobile automation would have. When I started to follow documentation and tutorials of Appium, the idea of mobile automation became clear in my mind and I surprised for the similarity with web automation. After I started to practice mobile automation, I realized how can it operates the testing process on mobile environment more accurate and faster and understood having knowledge of this topic can be really useful for my self-development. In general, I can say that I had a very nice experience for mobile automation and I am very pleased to have a basic idea about mobile automation.

5. Comparison of Appium and Selenium

Appium and Selenium have similar capabilities like they are both cross-platform products which can work on all environments in their domains. Automation by using Appium is just mobile version of automation with Selenium. They have similar approaches, characteristics and methods. The major difference is just their working platforms (mobile and web).