

# Smart Bin

# SVV Project Testing Report



59090011 Napat Traikityanukul  
59090028 Samita Suttisirikul  
59090037 Titaporn Chaisilwattana



## Outline

---

What is our project: ***Smart Bin***

Testing Activities:

**Backend Testing: Unit Testing**

**User Interface Automate Testing: Integration**

- Cavy



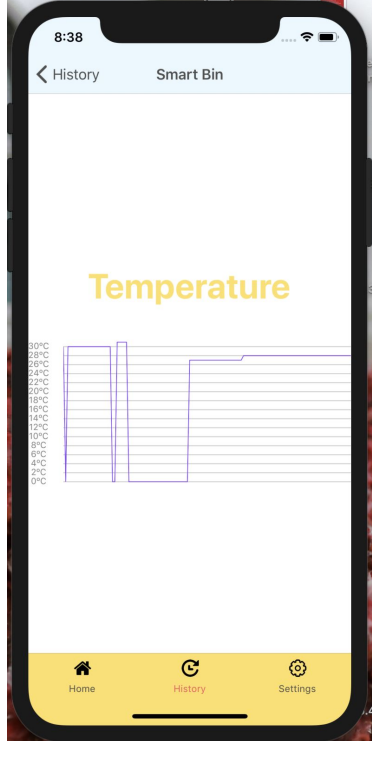
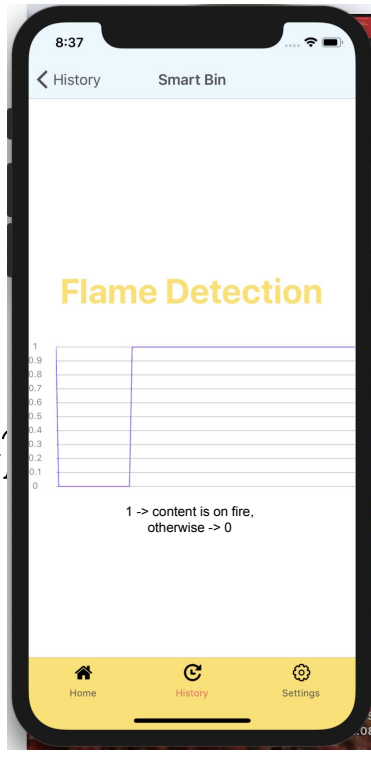
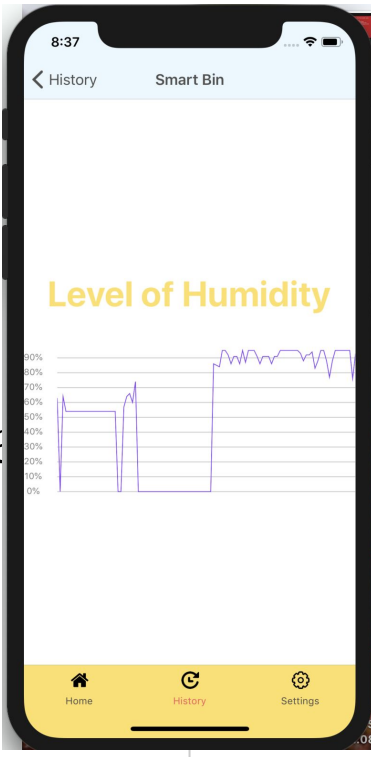
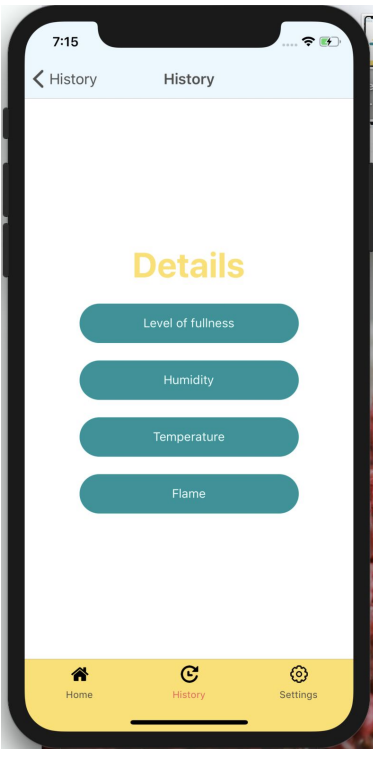
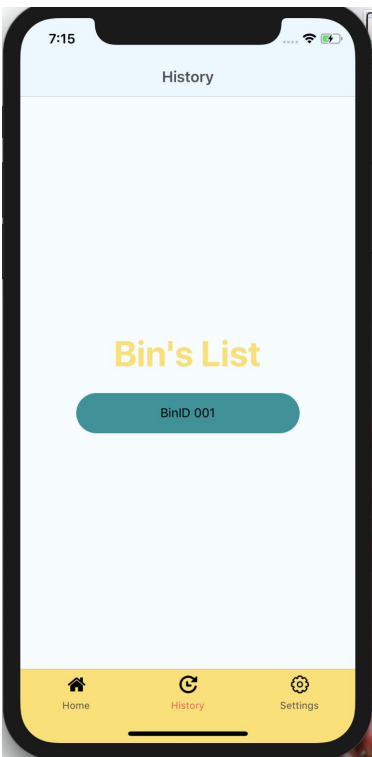
# Smart Bin

Easy to manage!

Monitor your bin realtime!

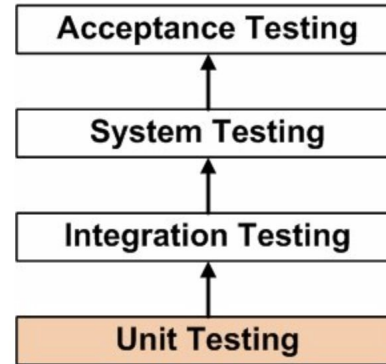
View your bin history!





# Unit test

- What is it?
- When is it performed?
- Who performs it?





# Testing View Level of Fullness Function

## ● Test Design

Description	User should be able to view correct level of fullness of each bin in real time.
Precondition	User already added a Smart Bin into their account.
O/S	MacOS Mojave
Steps	<ol style="list-style-type: none"><li>1. Select the bin that want to view status of.</li><li>2. System outputs graph of fullness of that bin.</li></ol>
Test Data Set	<ol style="list-style-type: none"><li>1.itemUltra[22,23,22,3]</li><li>2.itemUltra[22,22,23,3,22]</li><li>3.itemUltra[]</li></ol>



# Testing View Level of Fullness Function

- Test Cases, Execution, Result

S/N	Test Cases	Input Data	Expected Results	Actual Results	Pass/Fail
1	User throws in thrash, which make the bin full	<u>itemUltra</u> [22,23,22,3]	User able to view the chart showing real time data, and see the last data on the chart as 3 cm	The chart shows real time data as the user throws garbage into the bin and it's full(the latest data is <5)	pass
2	User empty the bin, which make the bin empty	<u>itemUltra</u> [22,22,23,3,22]	User able to view the chart showing real time data, and see the last data on the chart as 22 cm, which means bin is empty again after it was full	The chart shows real time data as the user just empty the bin	pass
3	User just opens the Smart bin for the first time	itemUltra[]	User able to view the chart showing chart with no data added yet	The chart shows nothing yet.	pass





# Cavy as UI

: a cross-platform integration automate test framework for React-Native

## ***Automate Testing***

: the use of software separate from the software being tested to control the execution of tests and the comparison of actual outcomes with predicted outcomes.

# Why Cavy?

React Native had only a handful of testing approaches available:



1. Unit testing components ([Jest](#)).
2. Shallow-render testing components ([enzyme](#)).
3. Testing within your native environment, using native JS hooks ([Appium](#)).
4. Testing completely within your native environment ([XCTest](#)).

**Cavy** fits in between shallow-render testing and testing within our native environment.



## Testing Register Page

- User registers filling in username, email and password.
- After user presses the 'SIGN UP' button, it will link to Home page.

## Index.js

```
1  import {AppRegistry} from 'react-native';
2  import App from './App';
3  import {name as appName} from './app.json';
4  import {Tester, TestHookStore} from 'cavy';
5  import LoginSpec from './spec/LoginSpec';
6  import React, {Component} from 'react';
7
8  const testHookStore = new TestHookStore();
9
10 export default class AppWrapper extends Component {
11   render(){
12     return (
13       <Tester specs={[LoginSpec]} store={testHookStore} waitTime={1000}>
14       <App/>
15       </Tester>
16     );
17   }
18 }
19 // AppRegistry.registerComponent(appName, () => App);
20 AppRegistry.registerComponent(appName, () => AppWrapper);
```



## Register.js

```
import React from "react";
import {
  View,
  Button,
  TextInput,
  StyleSheet,
  TouchableOpacity,
  Text,
  Image
} from "react-native";
import { hook } from 'cavy';
import { auth,db } from "../utils/firebase";
```



```
render() {
  return (
    <View style={styles.container}>
      <Image
        source={require("/Users/samitasutti/SB/assets/images/logo.png")}
        style={{ margin: 50, width: 150, height: 150 }}
      />
      <Text style={styles.text}>SIGN UP</Text>
      <View style={styles.inputContainer}>
        <TextInput
          ref={this.props.generateTestHook('Username.TextInput')}
          style={styles.inputs}
          placeholder="Username"
          autoCapitalize="none"
          //placeholderTextColor="#696969"
          onChangeText={val => this.onChangeText("name", val)}
        />
      </View>
      <View style={styles.inputContainer}>
        <TextInput
          ref={this.props.generateTestHook('Email2.TextInput')}
          style={styles.inputs}
          placeholder="Email"
          autoCapitalize="none"
          //placeholderTextColor="#696969"
          onChangeText={val => this.onChangeText("email", val)}
        />
      </View>
      <View style={styles.inputContainer}>
        <TextInput
          ref={this.props.generateTestHook('Password2.TextInput')}
          style={styles.inputs}
          placeholder="Password"
          secureTextEntry={true}
          autoCapitalize="none"
          //placeholderTextColor="#696969"
          onChangeText={val => this.onChangeText("password", val)}
        />
      </View>
    </View>
  );
}
```

```
<TouchableOpacity
  ref={this.props.generateTestHook('Signup.Button')}
  style={[styles.buttonContainer, styles.loginButton]}
  onPress={() => this.signUp(this.state.name, this.state.email, this.state.password)}
>
  <Text style={styles.loginText}>Register</Text>
</TouchableOpacity>
<TouchableOpacity
  onPress={() => this.props.navigation.navigate("Login")}
>
  <Text style={styles.buttonText}>Already have an account?</Text>
</TouchableOpacity>
</View>
```

```
export default hook(SignUp);
```

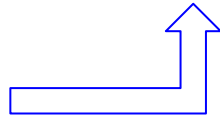
“

Register.js: Attach hooks to components that we want to access and manipulate in the tests

## Index.js

```
1  import {AppRegistry} from 'react-native';
2  import App from './App';
3  import {name as appName} from './app.json';
4  import {Tester, TestHookStore} from 'cavy';
5  import LoginSpec from './spec/LoginSpec';
6  import React, { Component } from 'react';
7
8  const testHookStore = new TestHookStore();
9
10 export default class AppWrapper extends Component {
11   render(){
12     return (
13       <Tester specs={[LoginSpec]} store={testHookStore} waitTime={1000}>
14         <App/>
15       </Tester>
16     );
17   }
18 }
19 // AppRegistry.registerComponent(appName, () => App);
20 AppRegistry.registerComponent(appName, () => AppWrapper);
```

Running test



Command Line/Terminal



```
samitas-mbp:SB samitasutti$ react-native run-ios
```

```
export default function(spec){

  spec.describe('Test Register', function(){
    spec.it('Test Register Page', async function(){
      await spec.press ('Register.Button');
      await spec.pause (2000);
      await spec.fillIn ('Username.TextInput','m');
      await spec.pause (2000);
      await spec.fillIn ('Email2.TextInput','meee@gmail.com');
      await spec.pause (2000);
      await spec.fillIn ('Password2.TextInput','123456');
      await spec.pause (2000);
      await spec.press ('Signup.Button');
      await spec.pause (2000);
      await spec.exists('Add.Button');
    });
  });
}
```

## LoginSpec.js

### Result

from  
console  
log



Sign up meee@gmail.com 123456

Register.js:25

Test Register: Test Register Page 

TestRunner.js:95

Cavy test suite stopped at Mon Jun 03 2019 14:49:18  
GMT+0700 (Indochina Time), duration: 11.092 seconds.

TestRunner.js:50





```
spec.describe('Test Register2', function(){
  spec.it('Test Register2 Page', async function(){
    await spec.press ('Register.Button');
    await spec.pause (2000);
    await spec.fillIn ('Username.TextInput','');
    await spec.pause (2000);
    await spec.fillIn ('Email2.TextInput','');
    await spec.pause (2000);
    await spec.fillIn ('Password2.TextInput','');
    await spec.pause (2000);
    await spec.press ('Signup.Button');
    await spec.pause (2000);
    await spec.exists('Add.Button');
```

## LoginSpec.js

Result from console log



Sign up	<u>Register.js:25</u>
Error:	<u>Register.js:38</u>
<pre> M {code: "auth/invalid-email", message: "The email address is badly formatted."} </pre>	
 ▶ Test Register2: Test Register2 Page  Could not find component with identifier Add.Button	<u>YellowBox.js:67</u>
Cavy test suite stopped at Mon Jun 03 2019 14:51:15 GMT+0700 (Indochina Time), duration: 11.754 seconds.	<u>TestRunner.js:50</u>



# Thank you.

*Any questions ?*