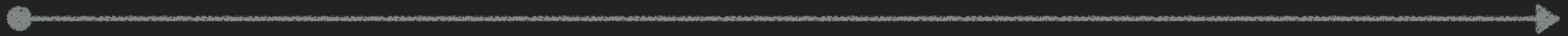
COCOAHEADS • AUG 2018

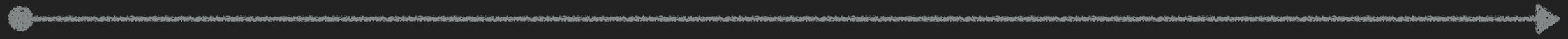
COCOAHEADS • AUG 2018

UI-TDD

SOFTWARE TESTING



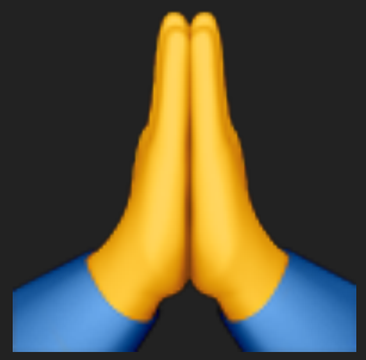
SOFTWARE TESTING



Repeatability

SOFTWARE TESTING

INFORMAL MANUAL TESTING

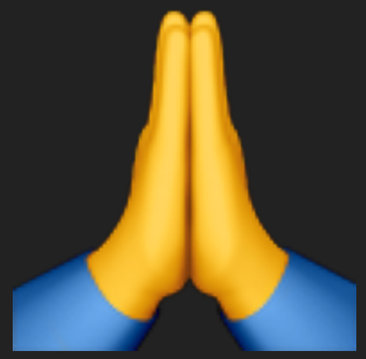


Repeatability

SOFTWARE TESTING

INFORMAL MANUAL TESTING

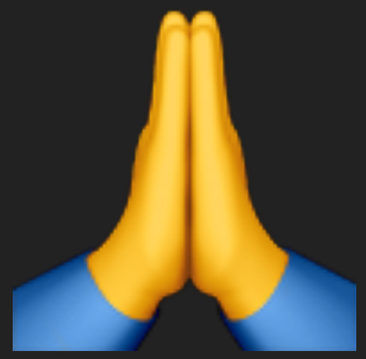
FORMAL MANUAL TESTING



Repeatability

SOFTWARE TESTING

INFORMAL MANUAL TESTING



FORMAL MANUAL TESTING

AUTOMATED TESTING

Repeatability

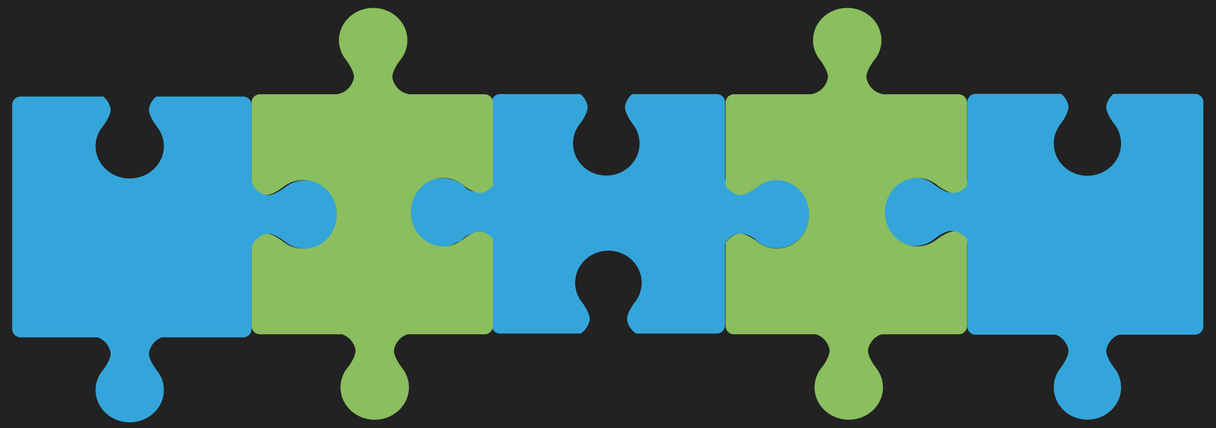
SOFTWARE TESTING

AUTOMATED

SOFTWARE TESTING

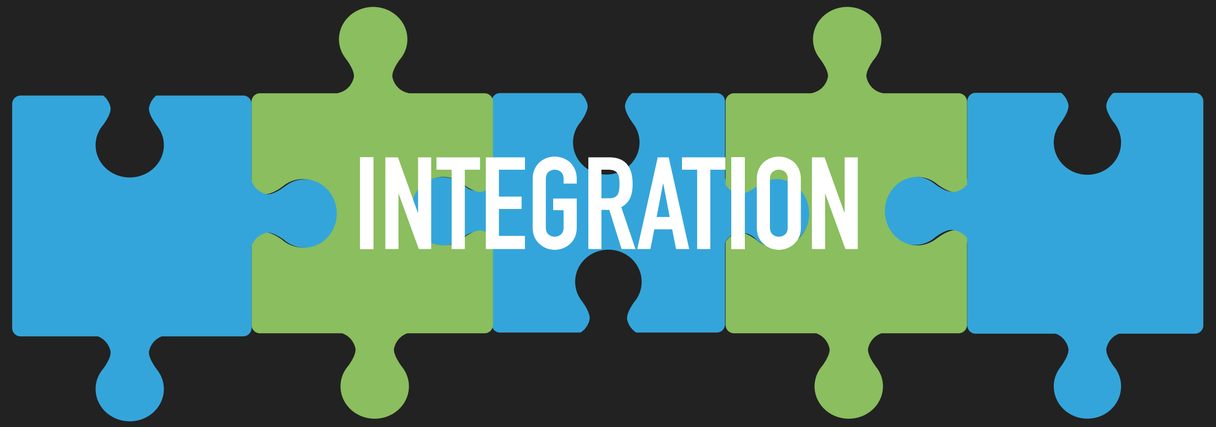
AUTOMATED

SOFTWARE TESTING



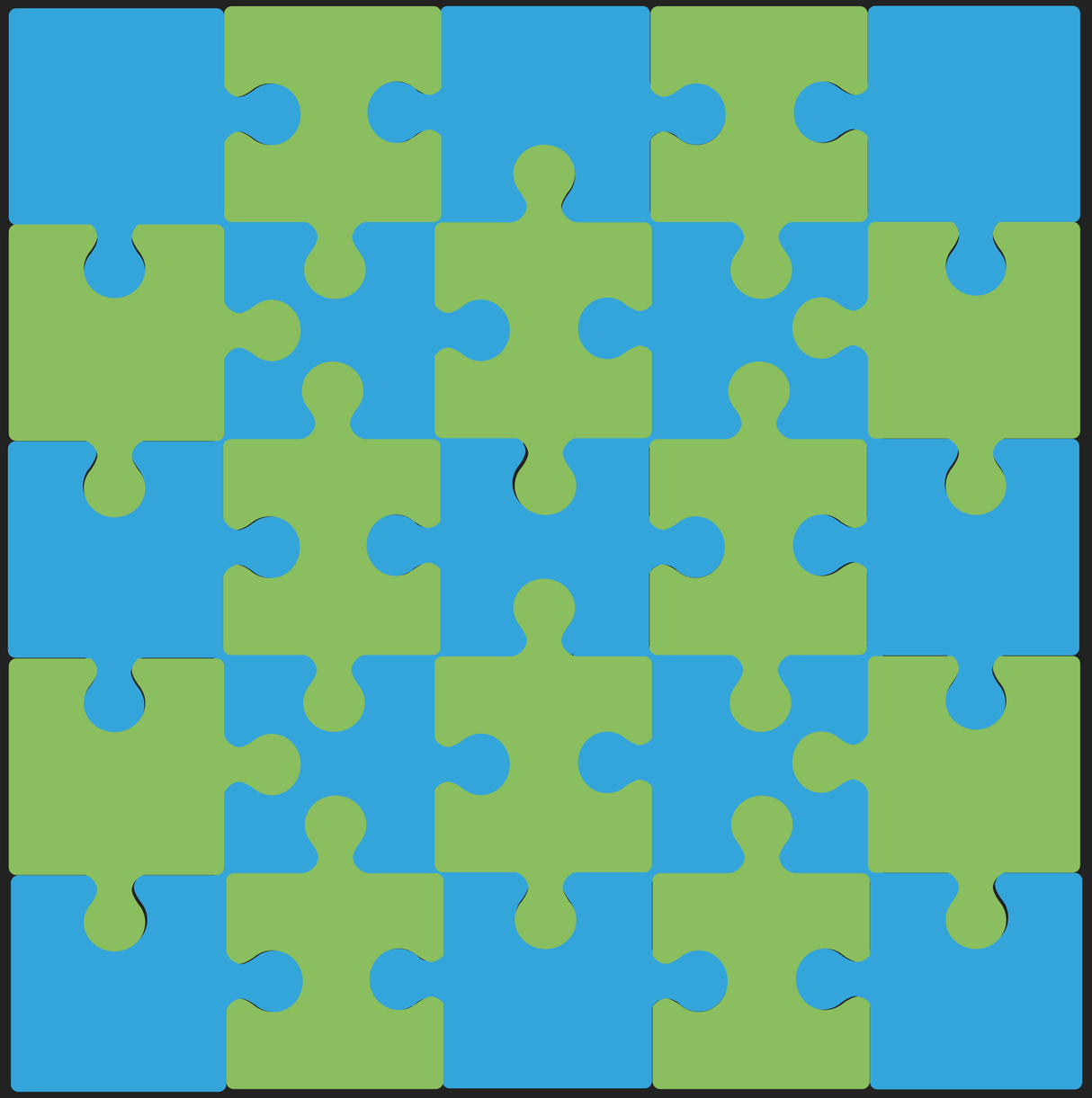
AUTOMATED

SOFTWARE TESTING



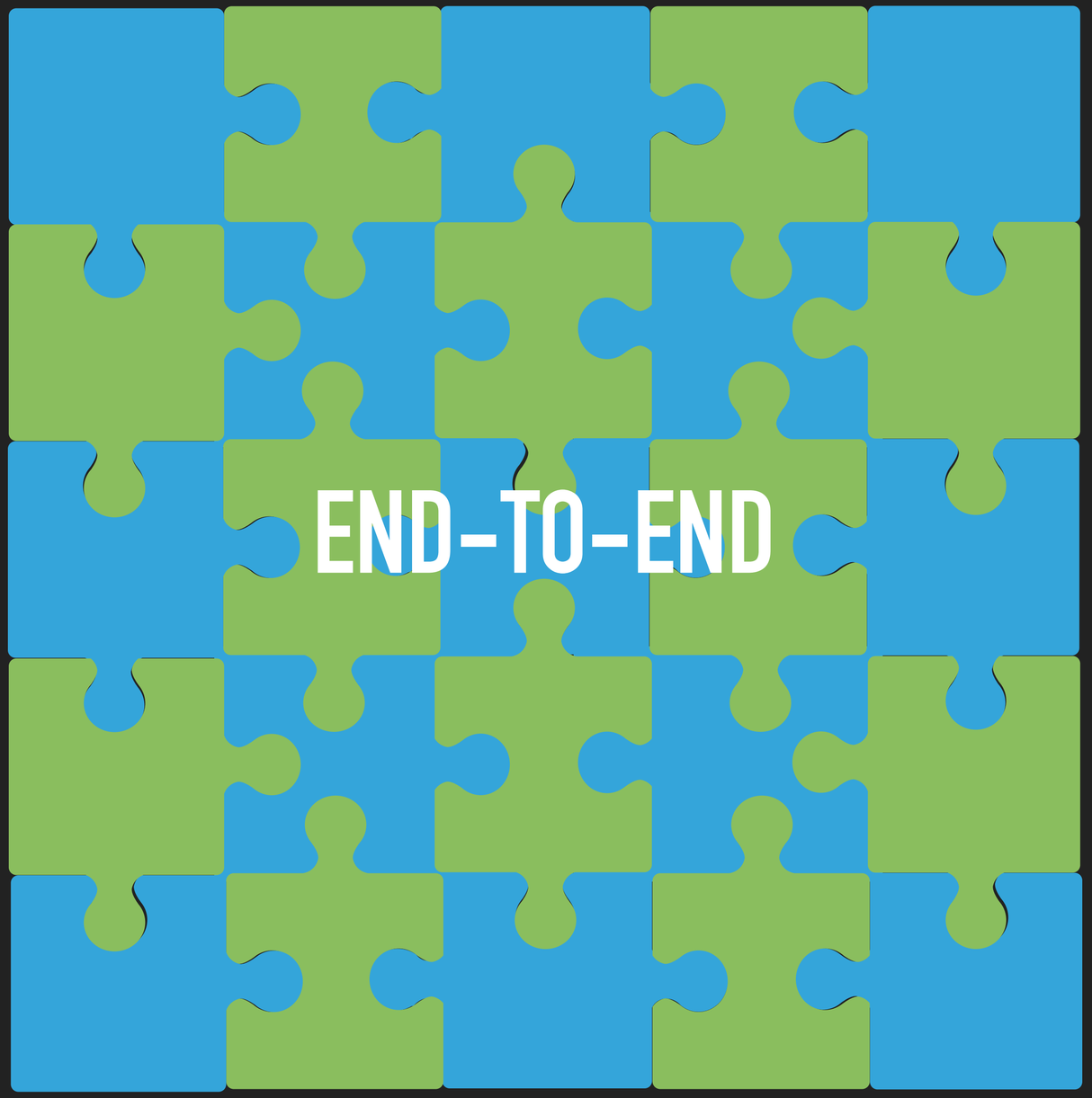
AUTOMATED

SOFTWARE TESTING



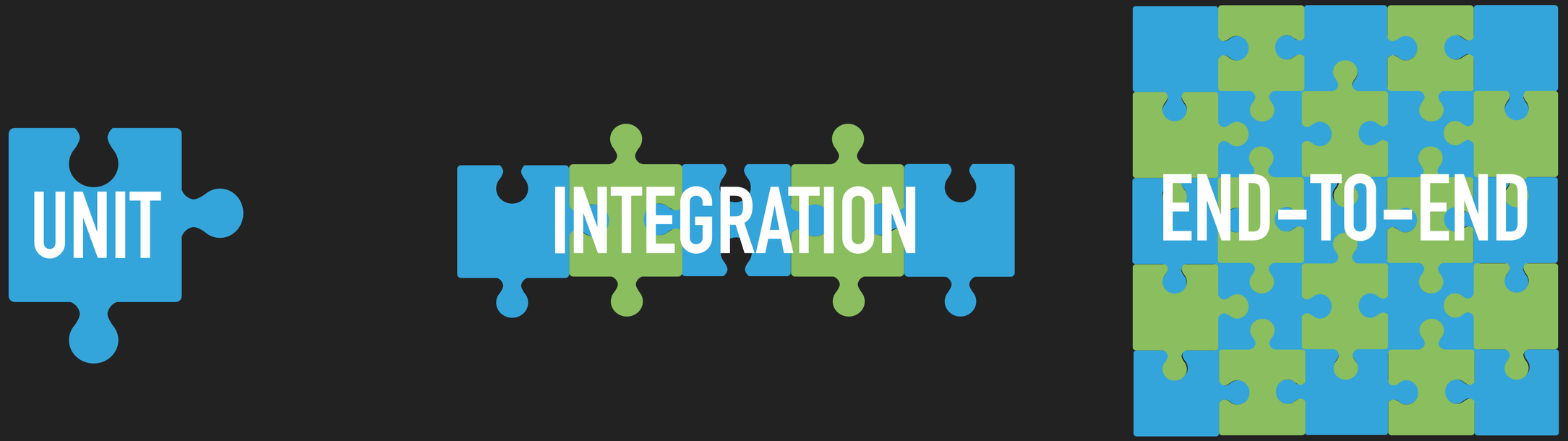
AUTOMATED

SOFTWARE TESTING



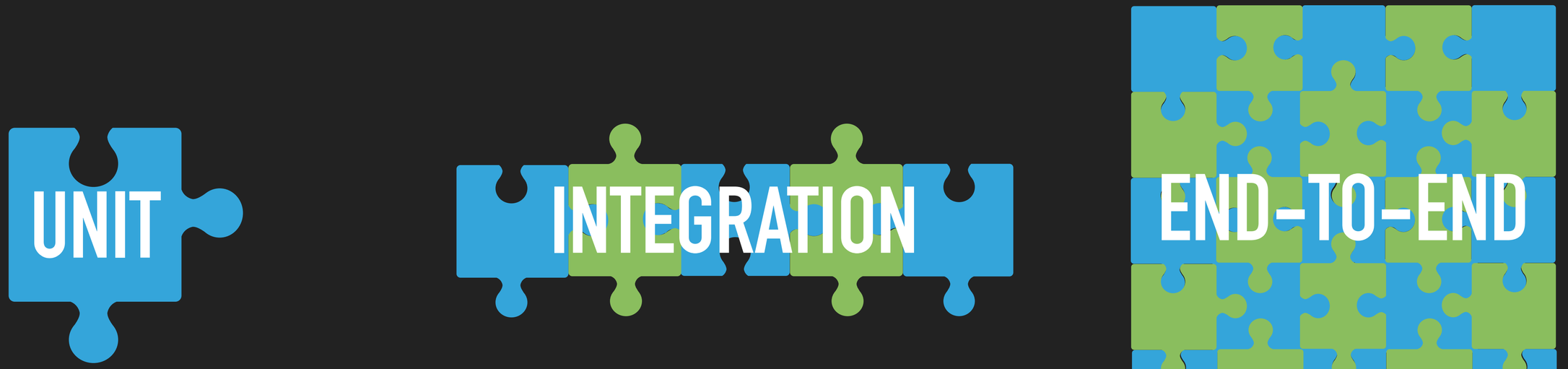
AUTOMATED

SOFTWARE TESTING



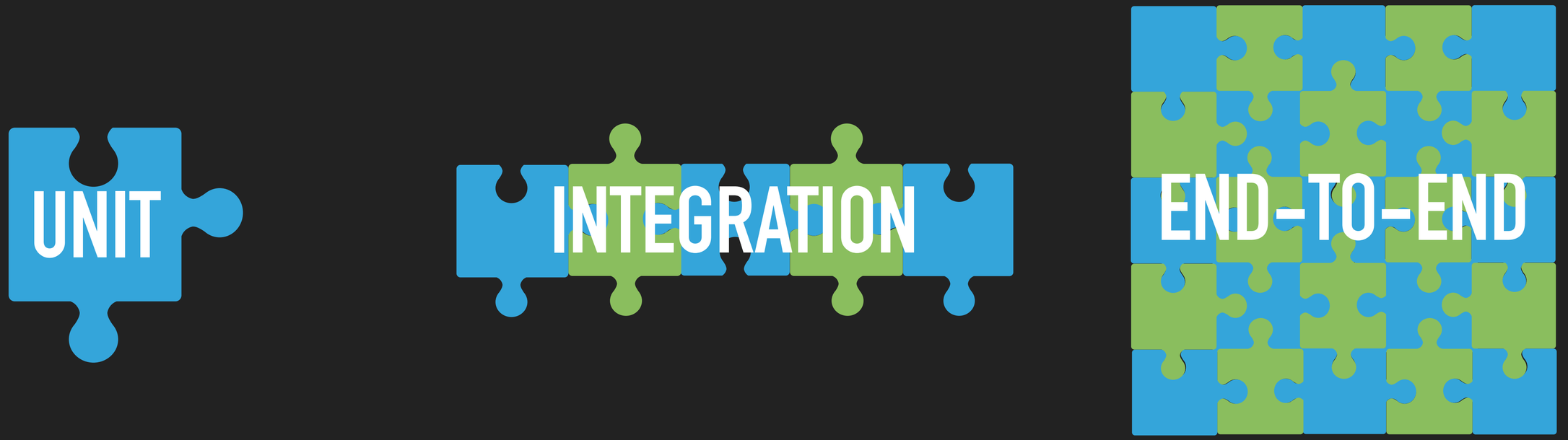
AUTOMATED

SOFTWARE TESTING



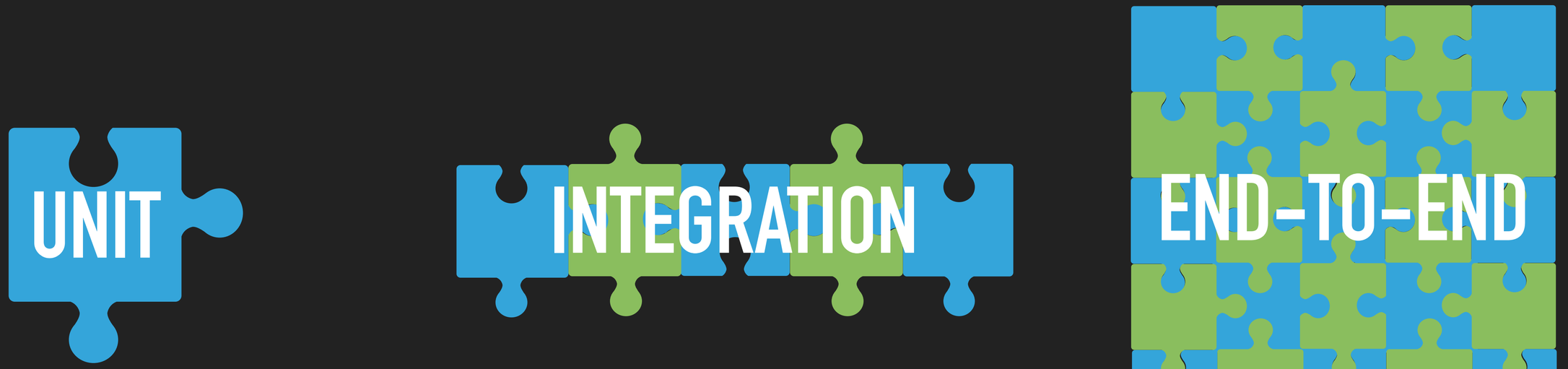
AUTOMATED

SOFTWARE TESTING



Time/Complexity

AUTOMATED
SOFTWARE TESTING



Time/Complexity

Realism

AUTOMATED

SOFTWARE TESTING

UI TESTS

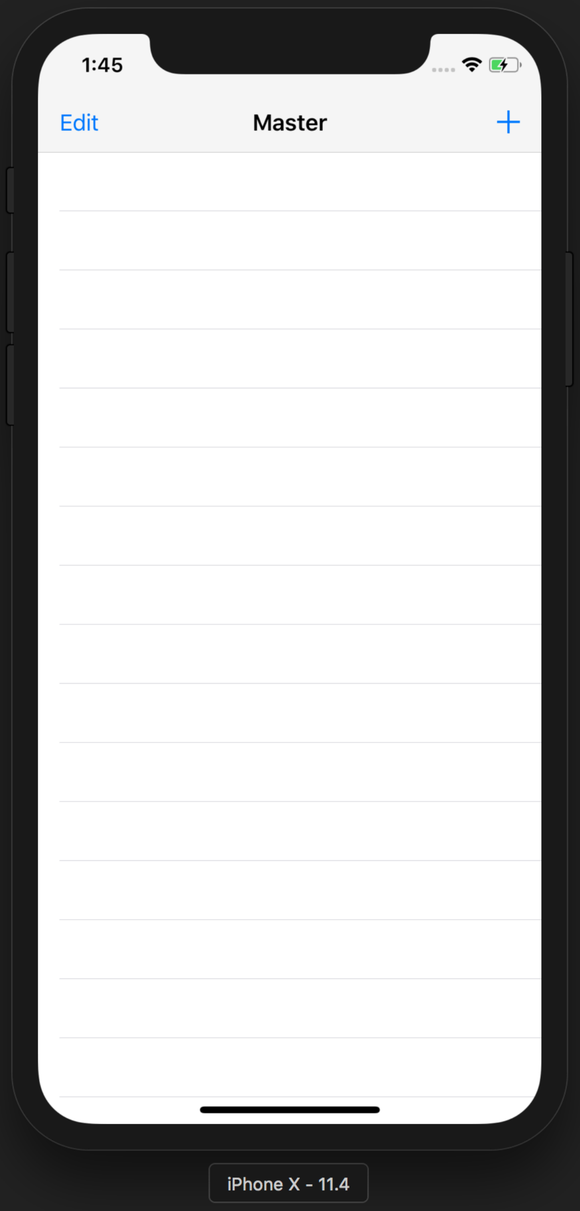
Time/Complexity

Realism

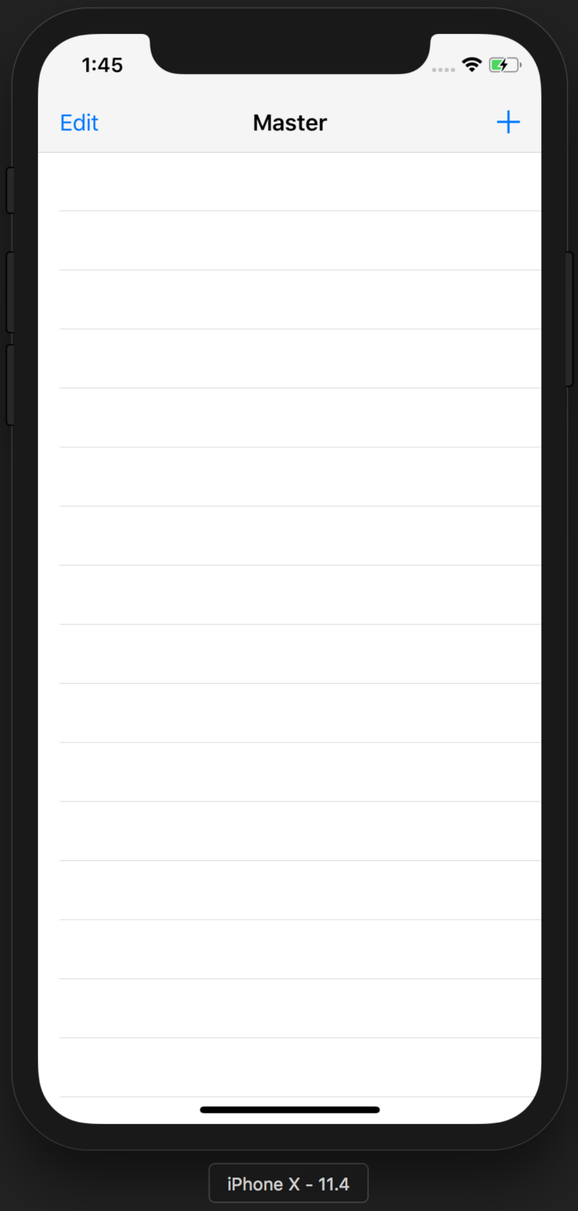
AUTOMATED

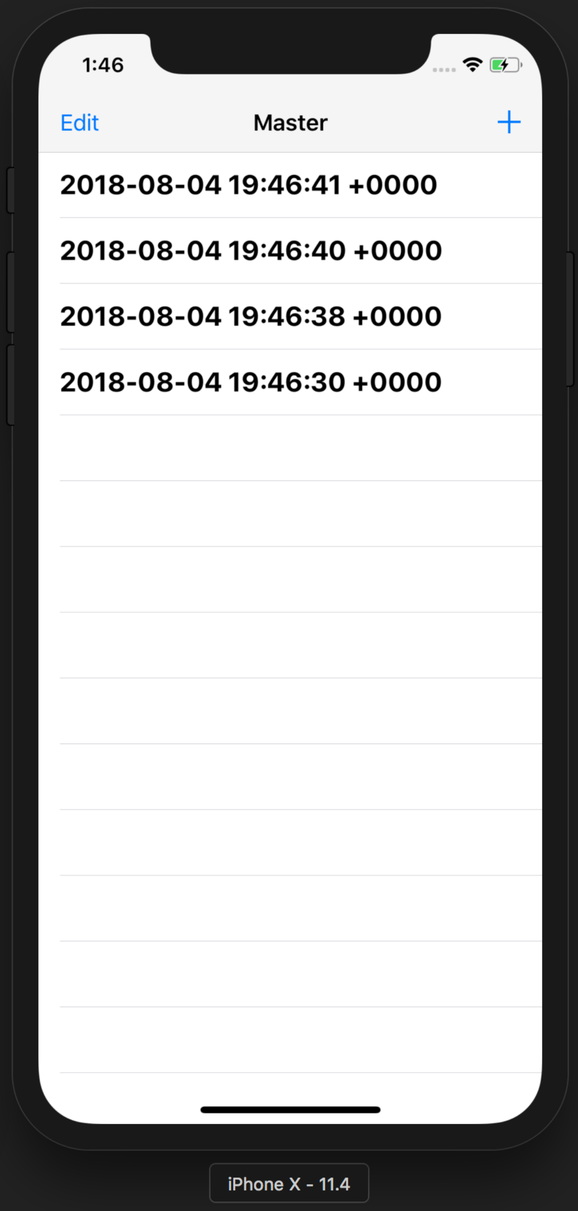
SOFTWARE TESTING

UI TESTS

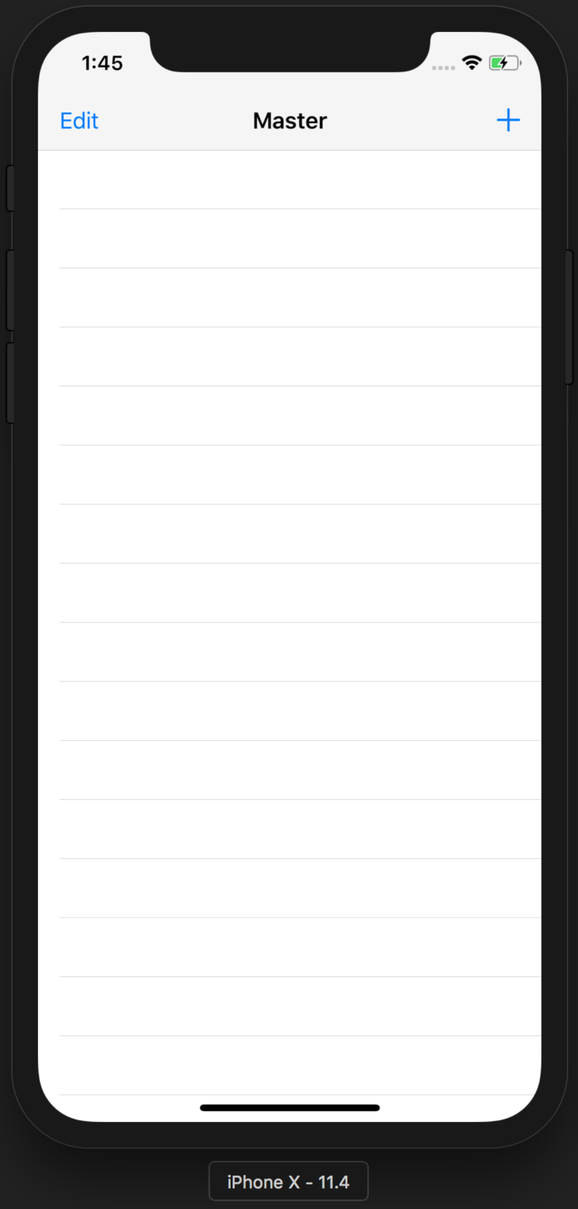


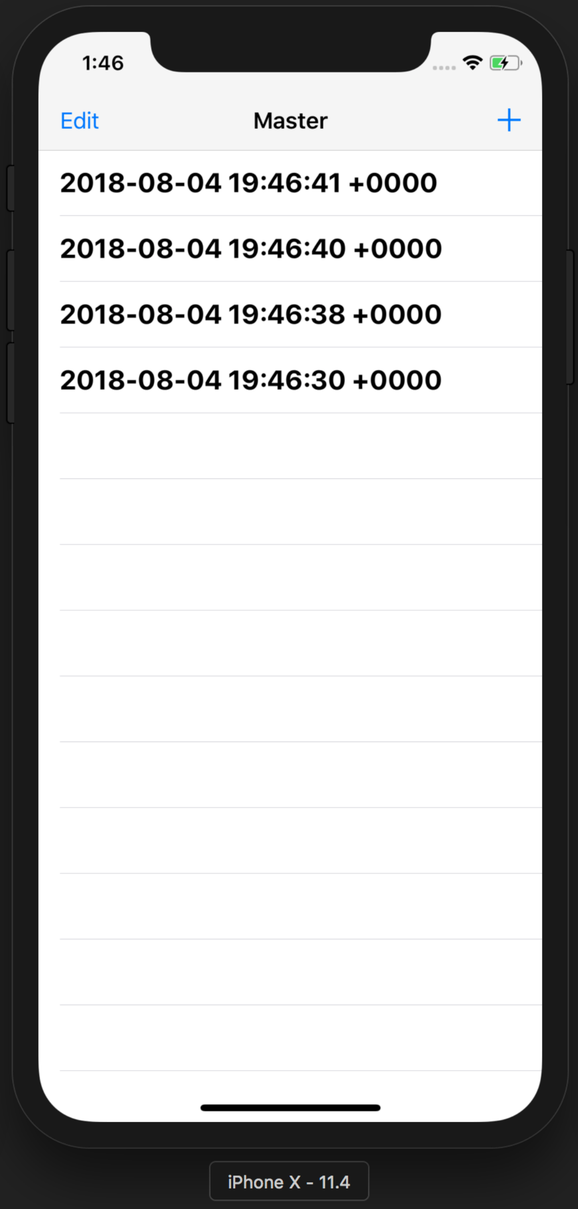
UI TESTS

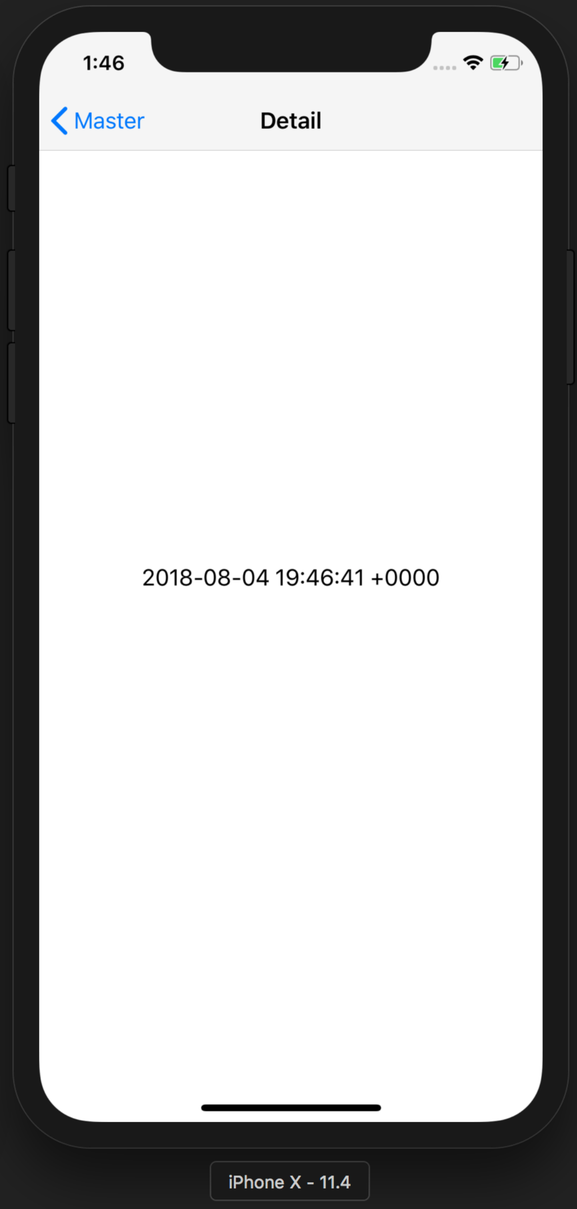

1:45
Edit
Master
iPhone X - 11.4


1:46
Edit
Master
2018-08-04 19:46:41 +0000
2018-08-04 19:46:40 +0000
2018-08-04 19:46:38 +0000
2018-08-04 19:46:30 +0000
iPhone X - 11.4

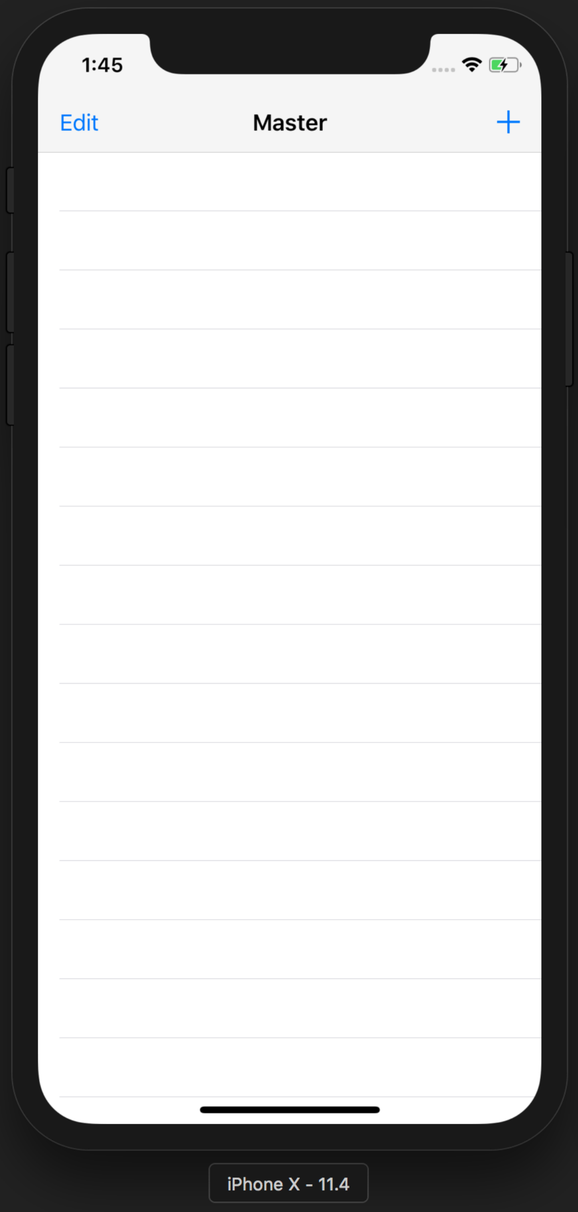
UI TESTS

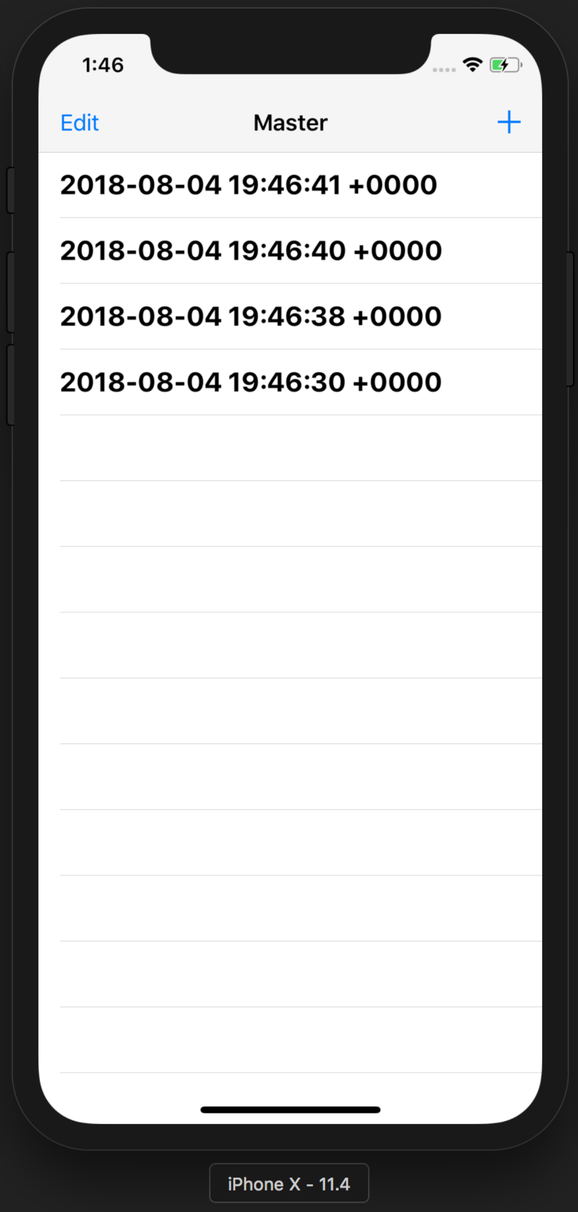

1:45
Edit
Master
iPhone X - 11.4

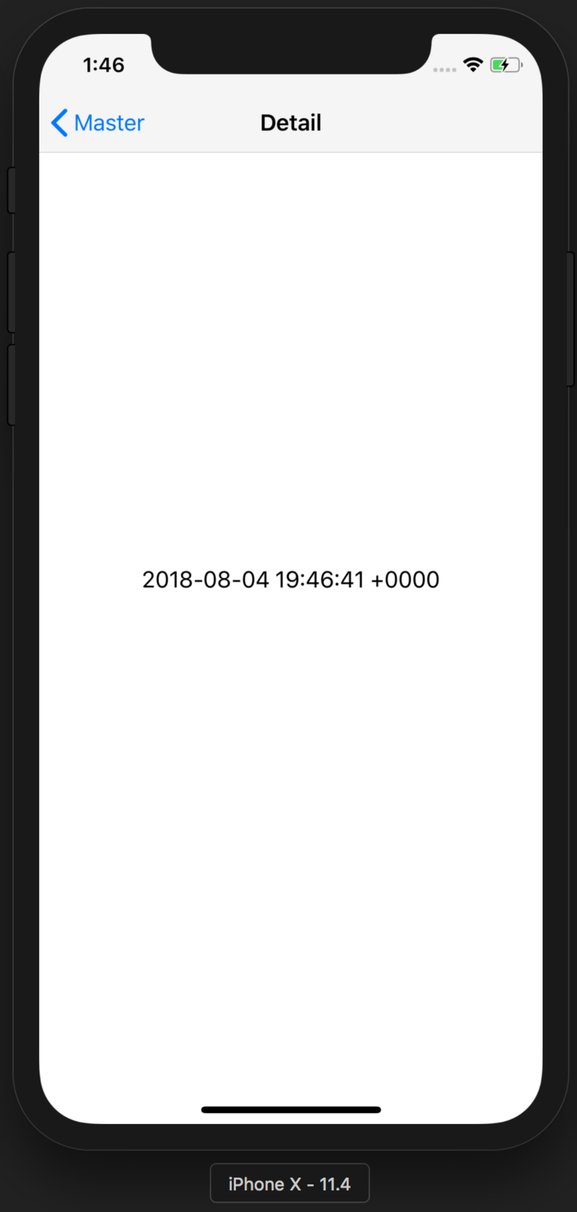

1:46
Edit
Master
2018-08-04 19:46:41 +0000
2018-08-04 19:46:40 +0000
2018-08-04 19:46:38 +0000
2018-08-04 19:46:30 +0000
iPhone X - 11.4

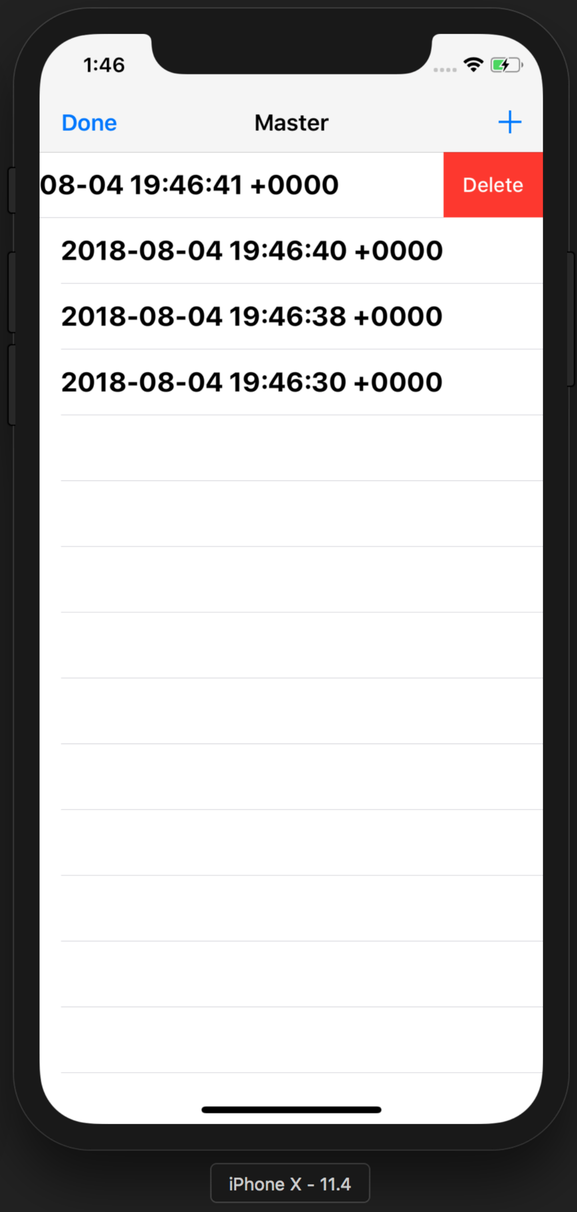

1:46
Master
Detail
2018-08-04 19:46:41 +0000
iPhone X - 11.4

UI TESTS


1:45
Edit
Master
iPhone X - 11.4


1:46
Edit
Master
2018-08-04 19:46:41 +0000
2018-08-04 19:46:40 +0000
2018-08-04 19:46:38 +0000
2018-08-04 19:46:30 +0000
iPhone X - 11.4


1:46
Master
Detail
2018-08-04 19:46:41 +0000
iPhone X - 11.4


1:46
Done
Master
08-04 19:46:41 +0000
Delete
2018-08-04 19:46:40 +0000
2018-08-04 19:46:38 +0000
2018-08-04 19:46:30 +0000
iPhone X - 11.4

UI TESTS

UI TESTS

VERIFY BEHAVIOR

UI TESTS

VERIFY BEHAVIOR

BUILT ON ACCESSIBILITY

UI TESTS

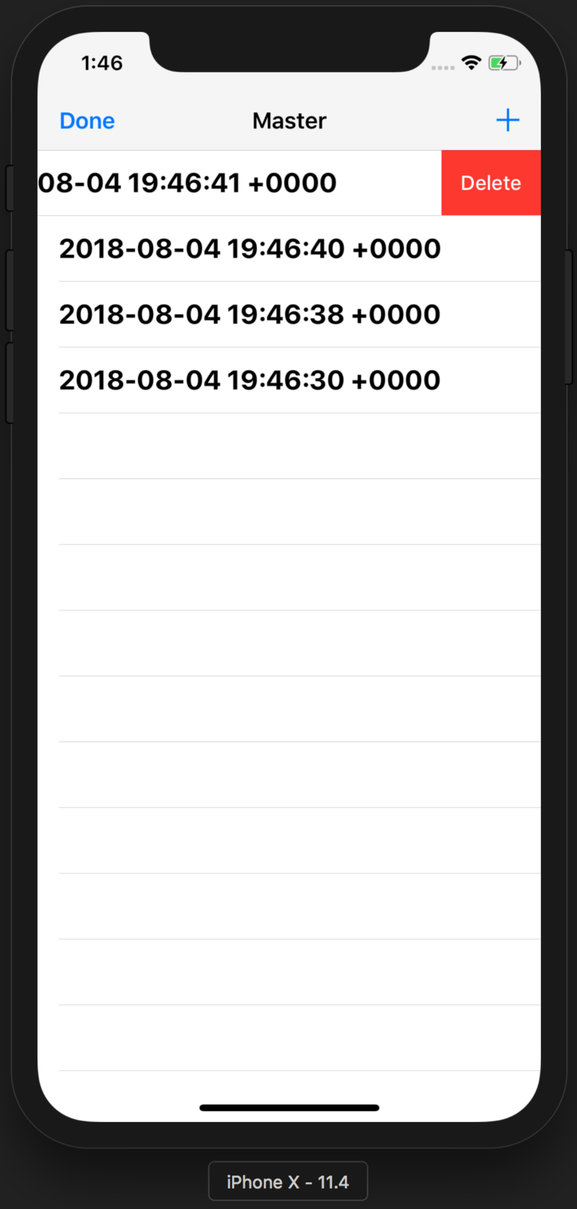
VERIFY BEHAVIOR

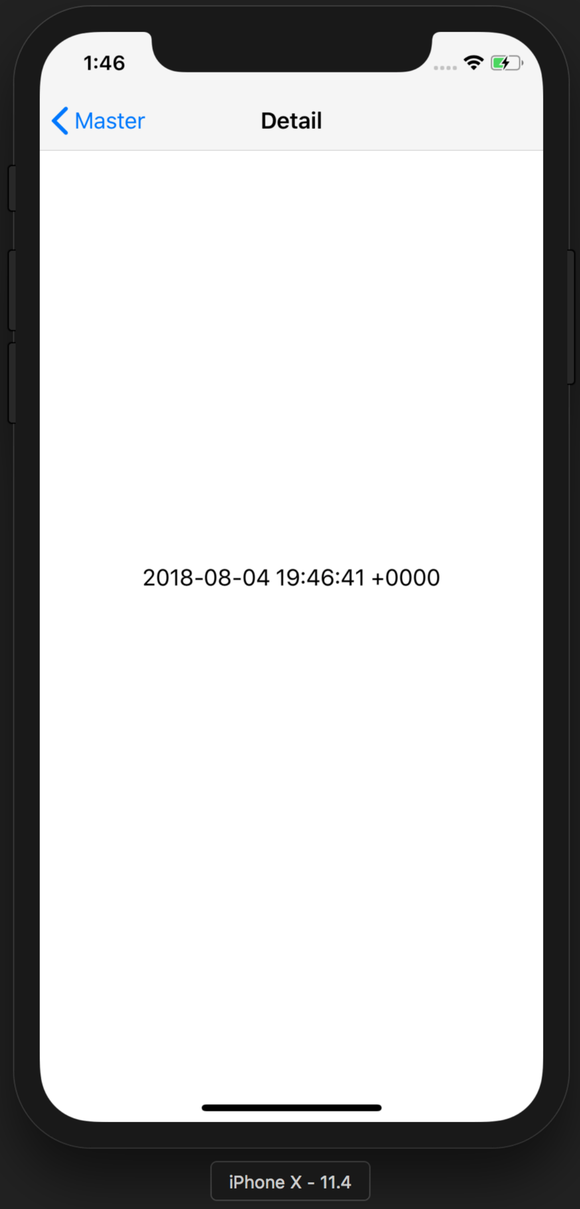
BUILT ON ACCESSIBILITY

RECORD TO LEARN

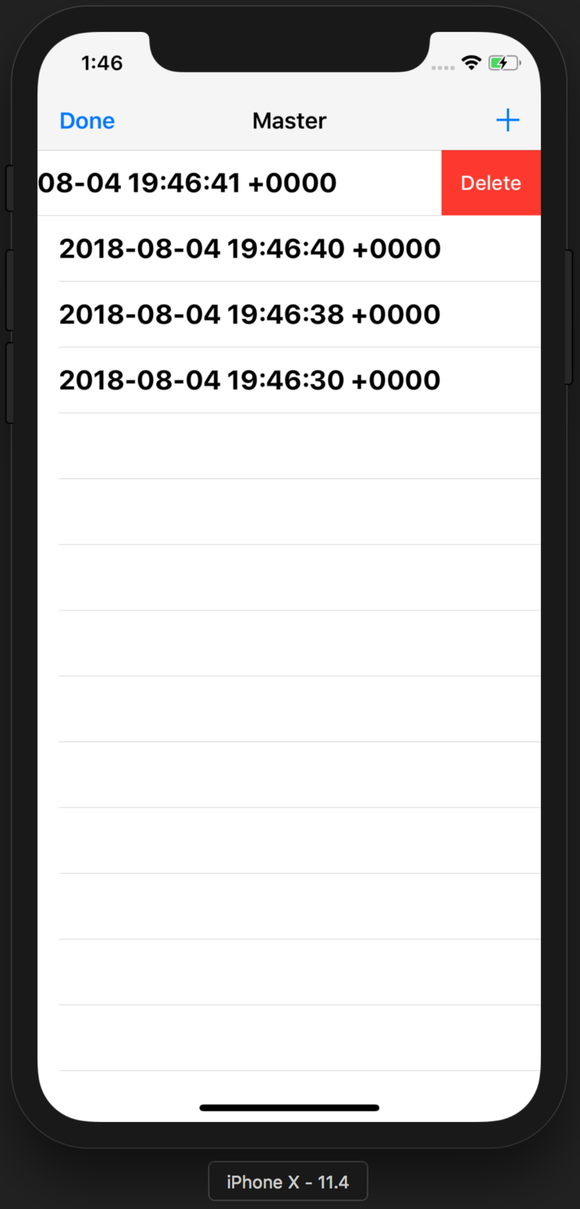
UI TESTS

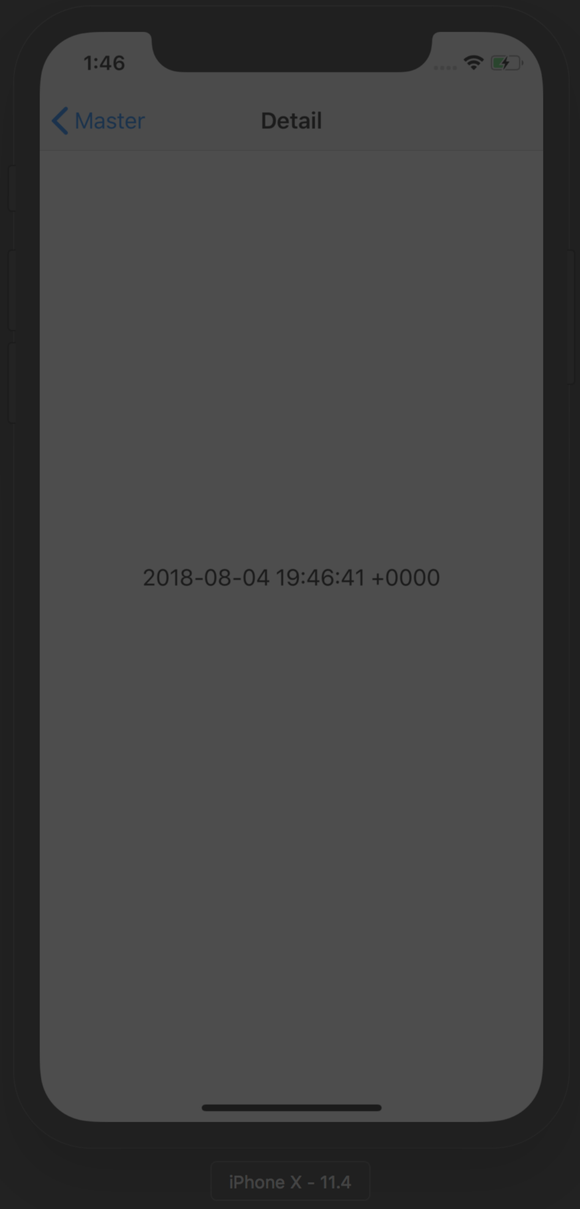
BETTER UI TESTING



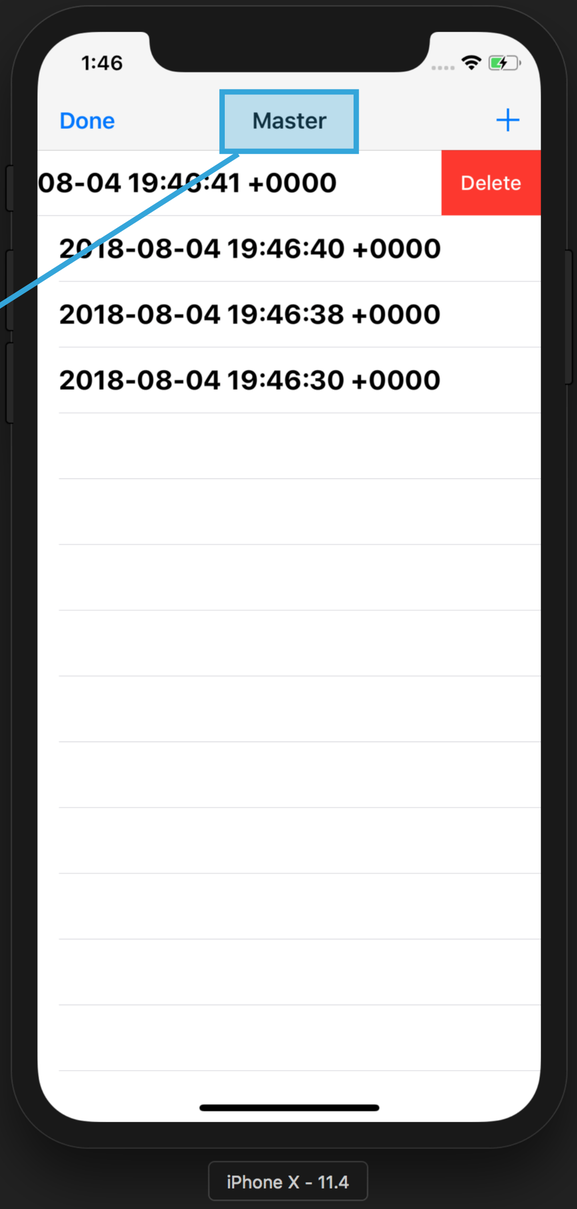


BETTER UI TESTING

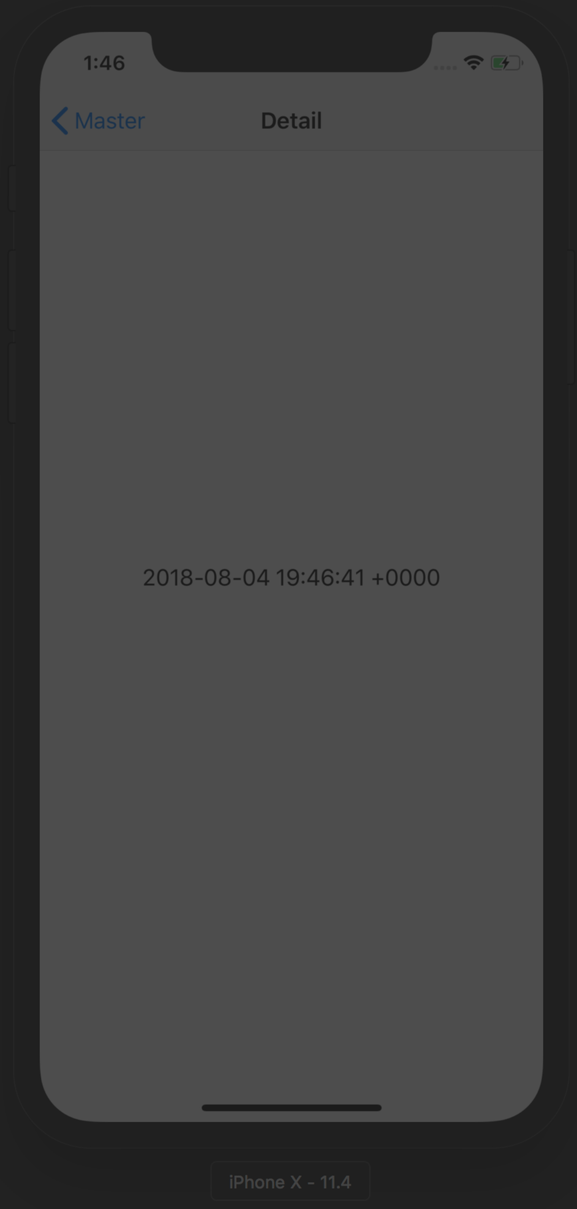




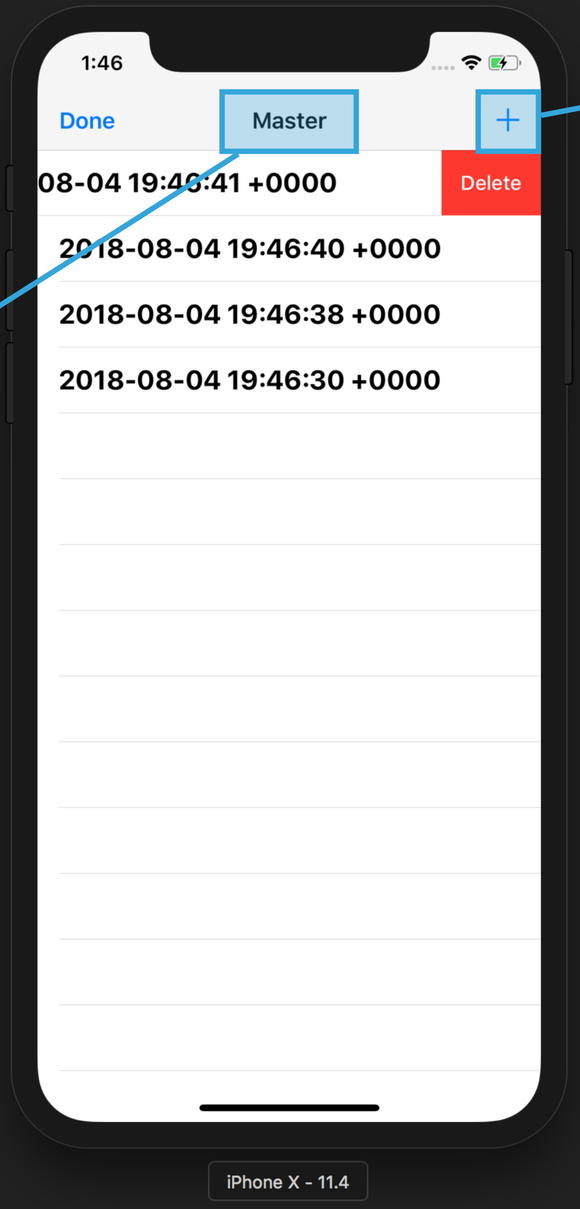
BETTER UI TESTING



Title

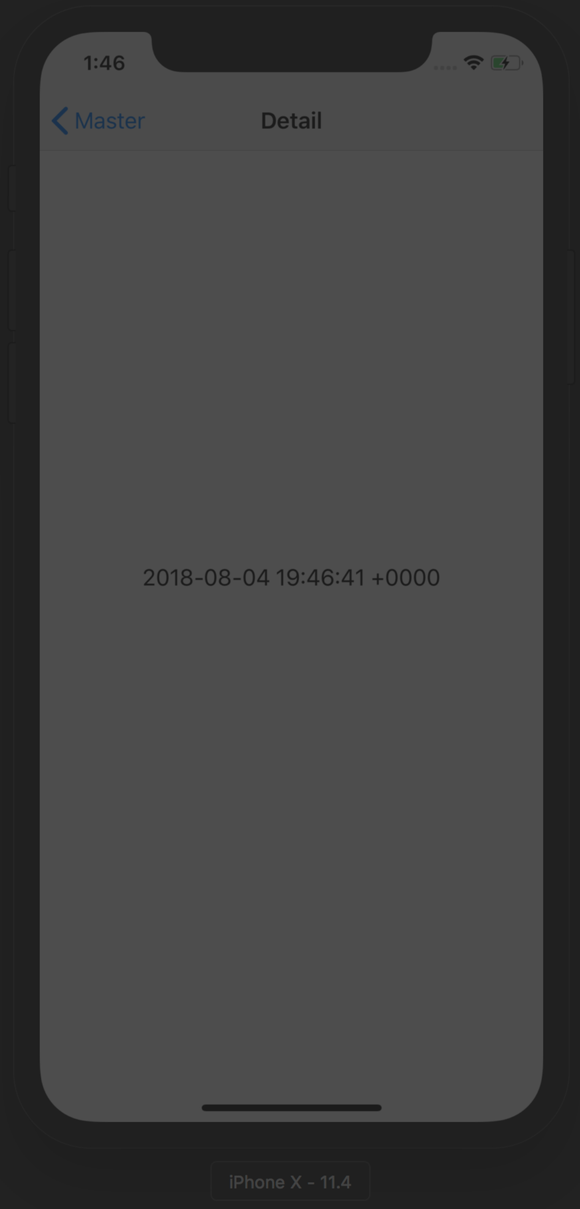


BETTER UI TESTING

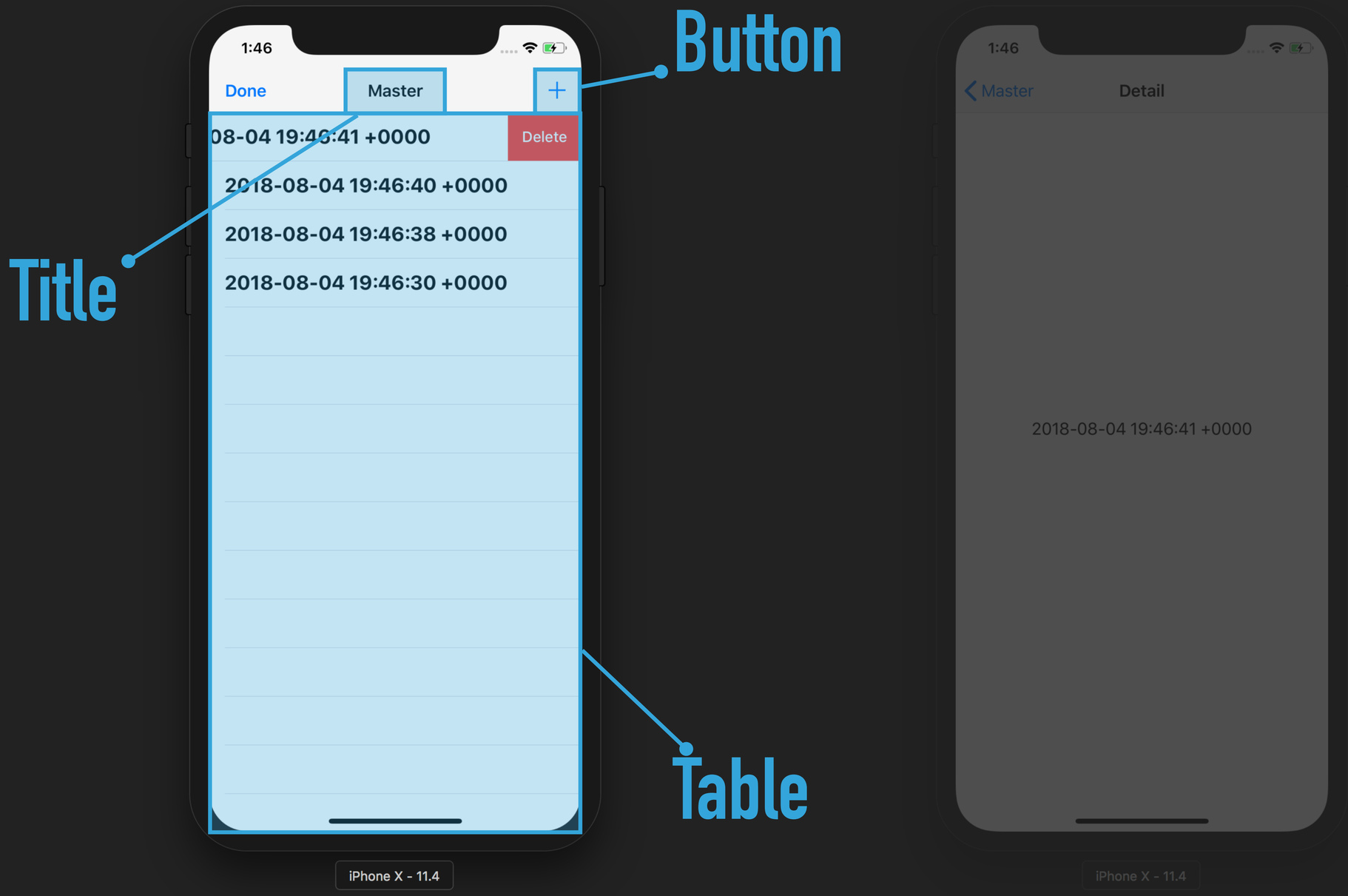


Button

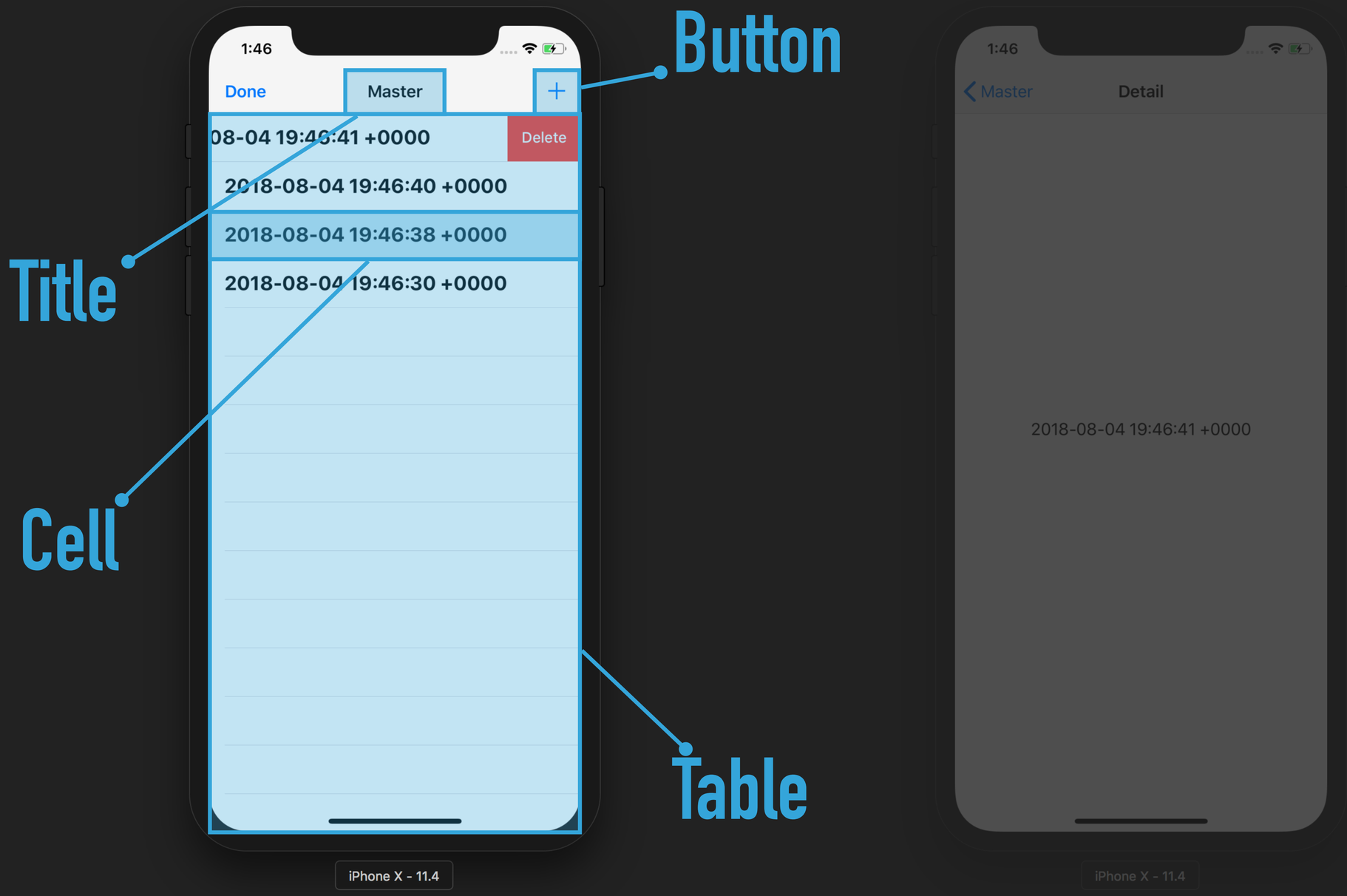
Title



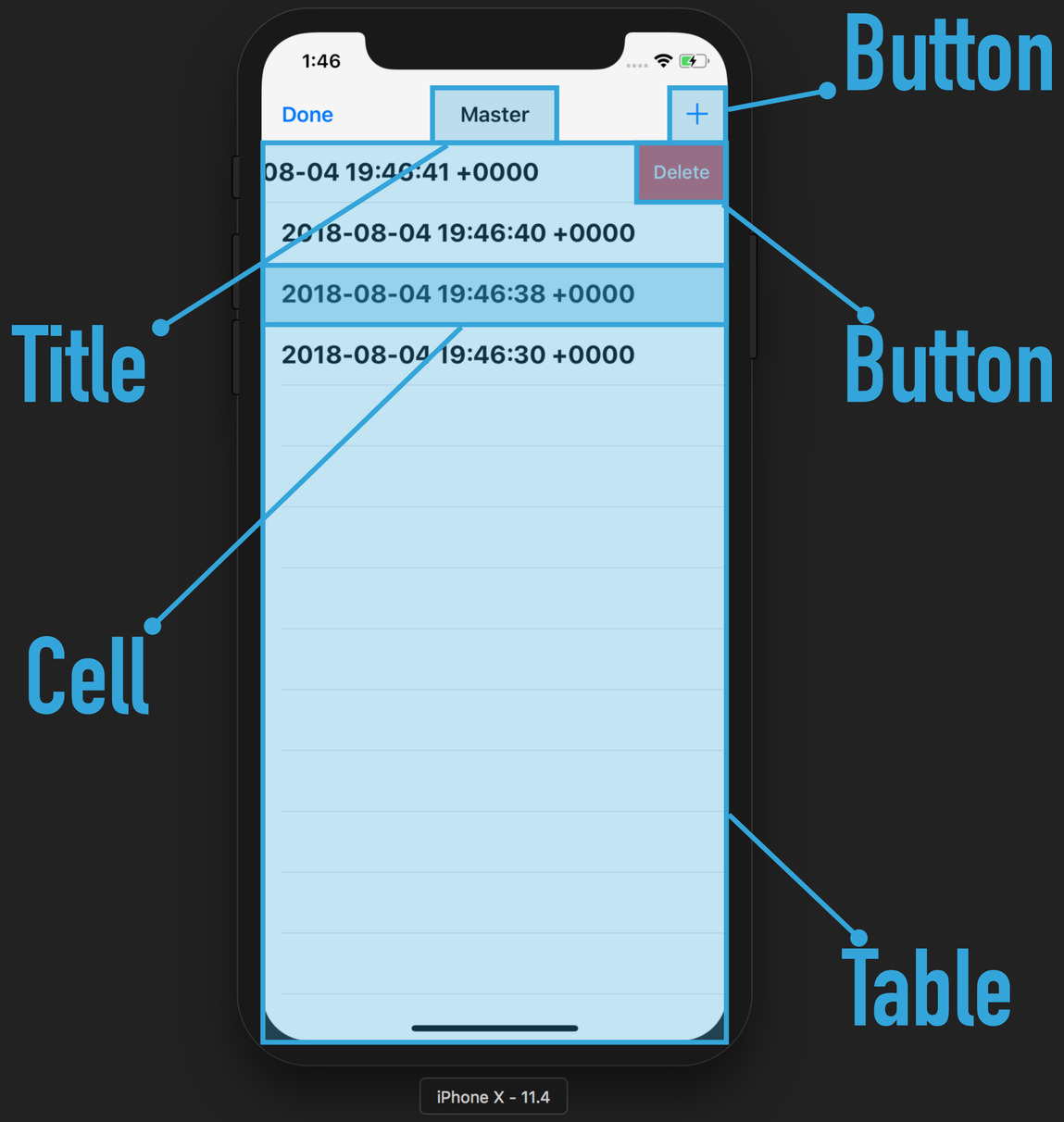
BETTER UI TESTING

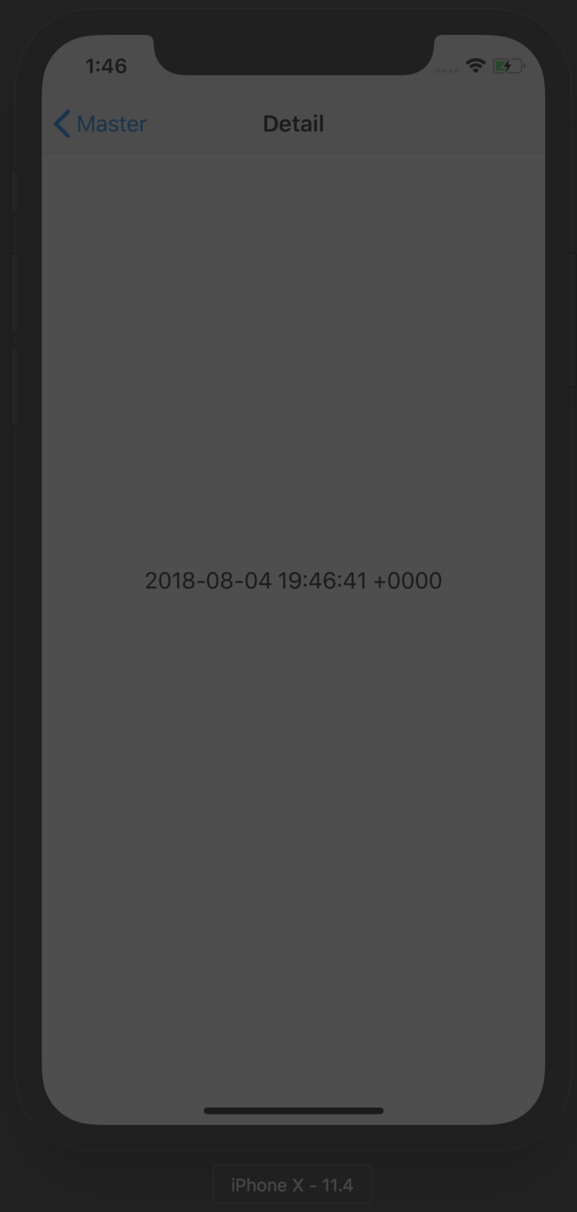


BETTER UI TESTING

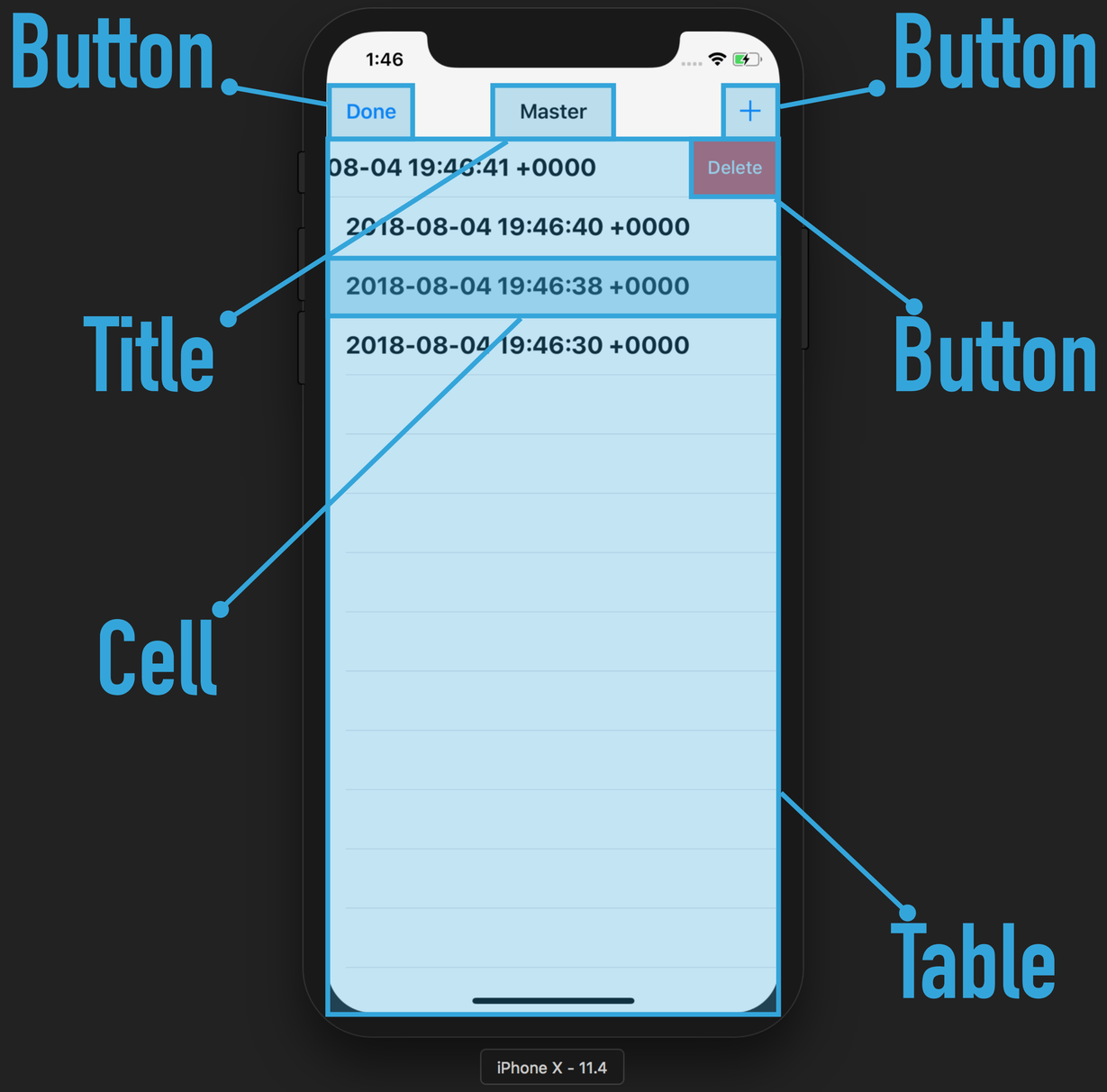


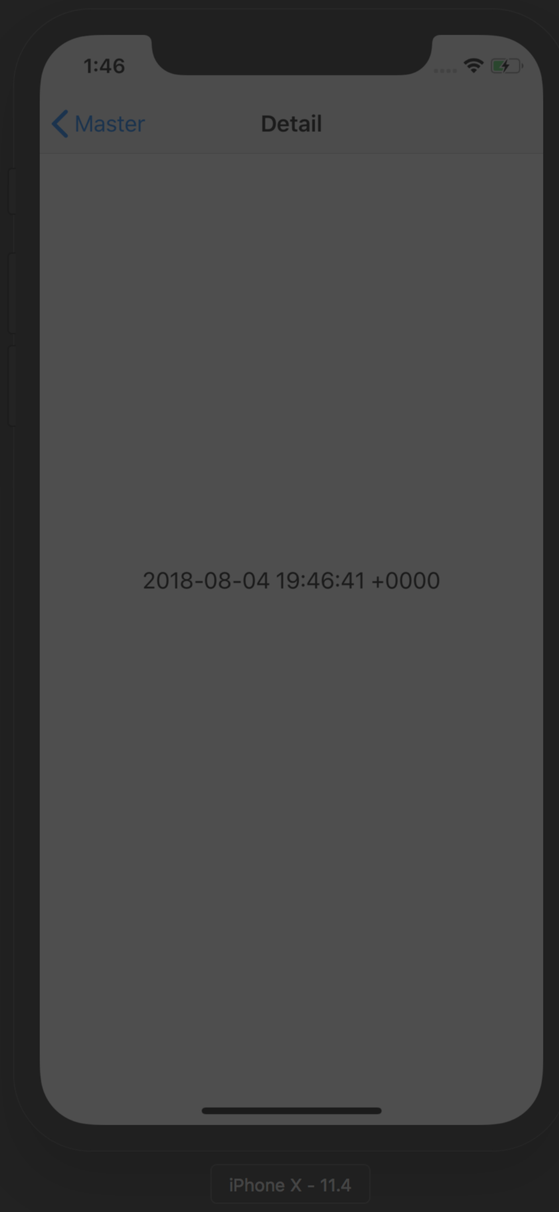
BETTER UI TESTING



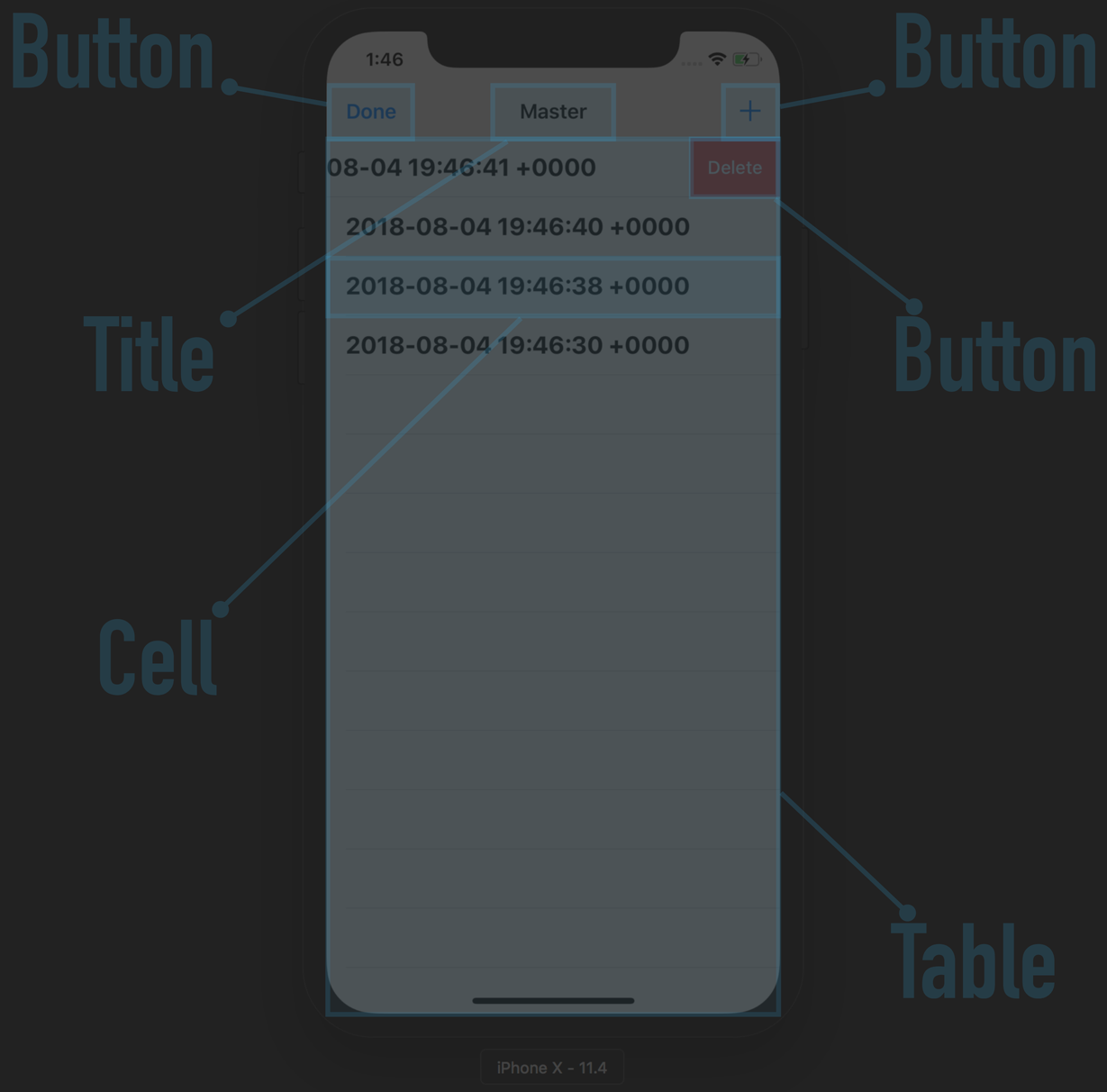


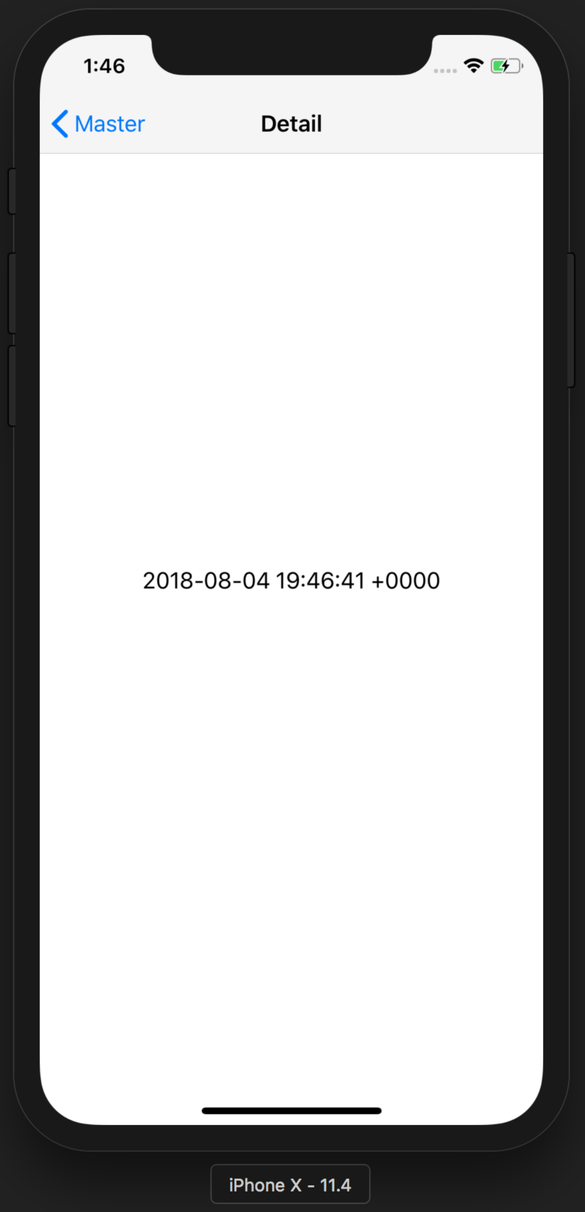
BETTER UI TESTING



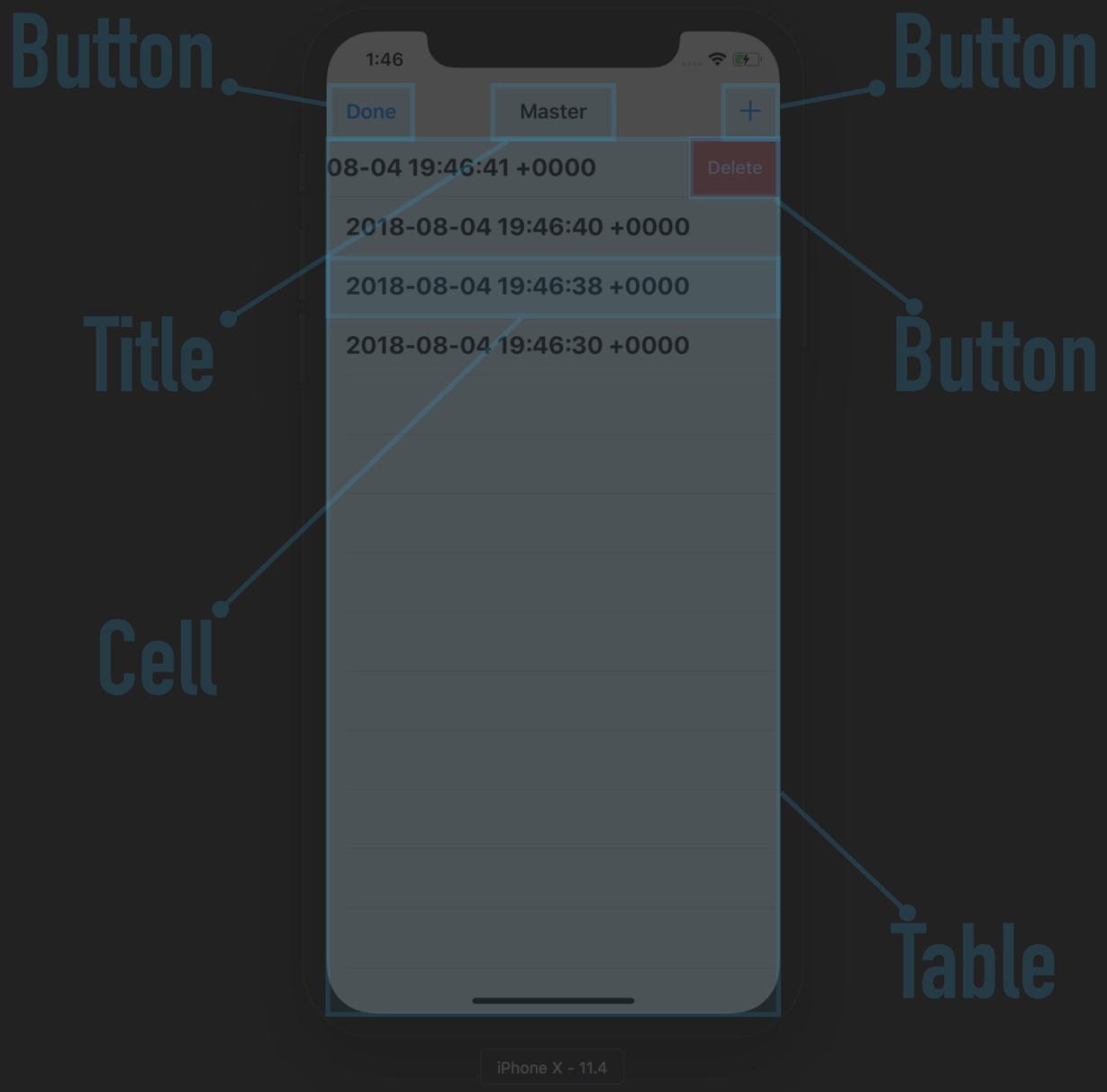


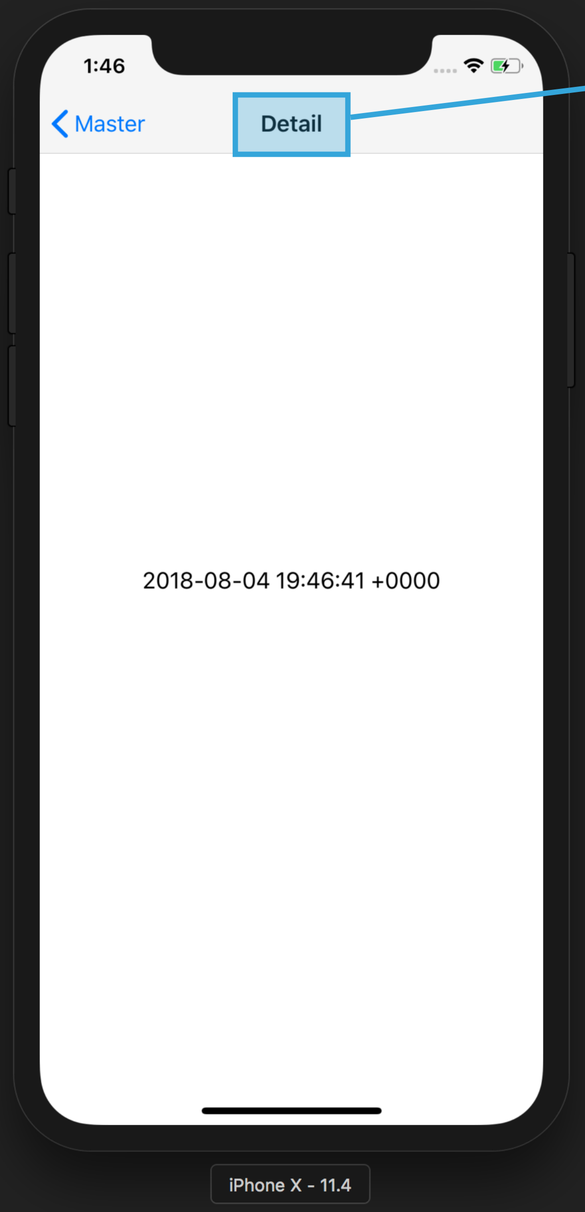
BETTER UI TESTING



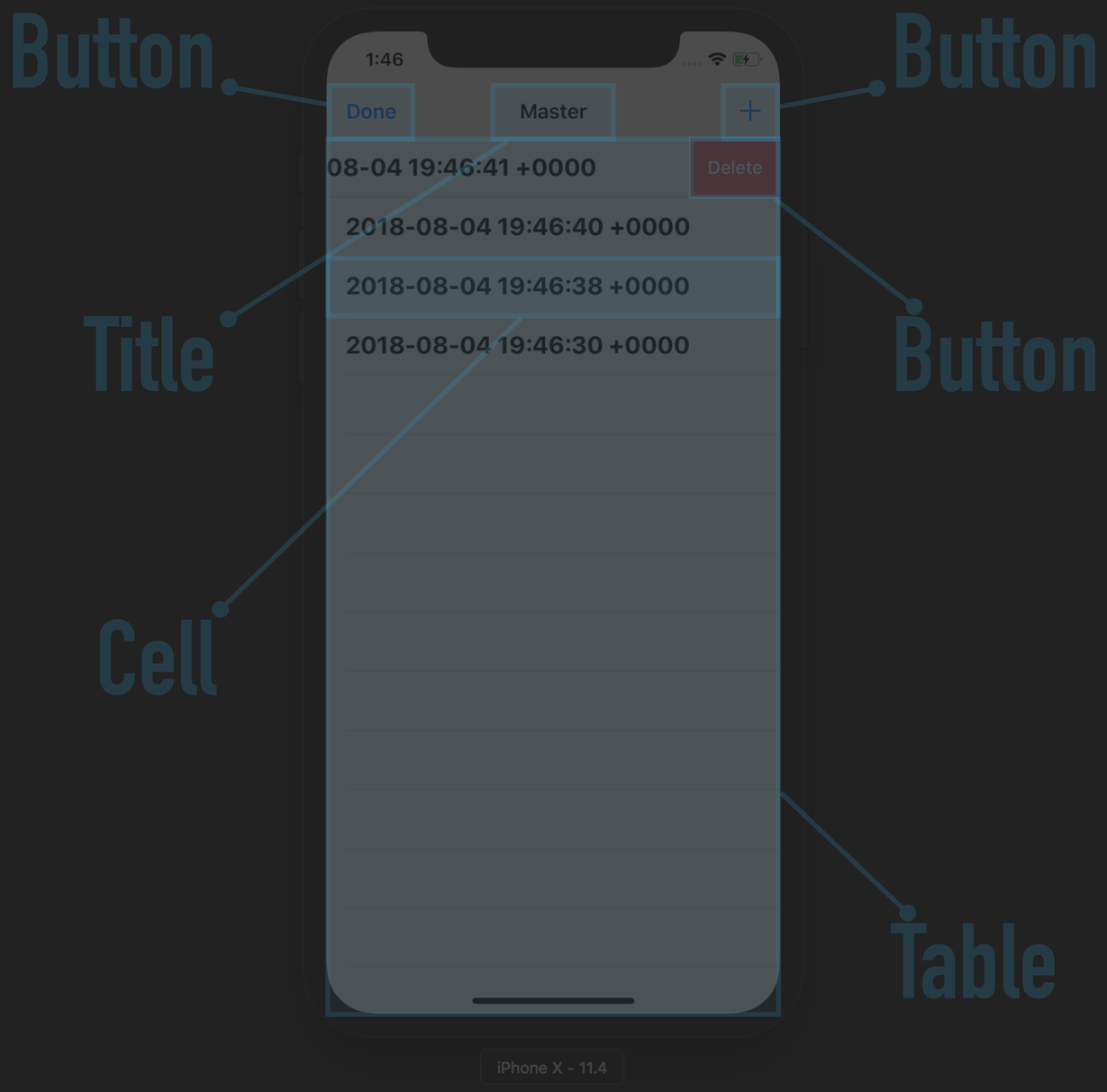


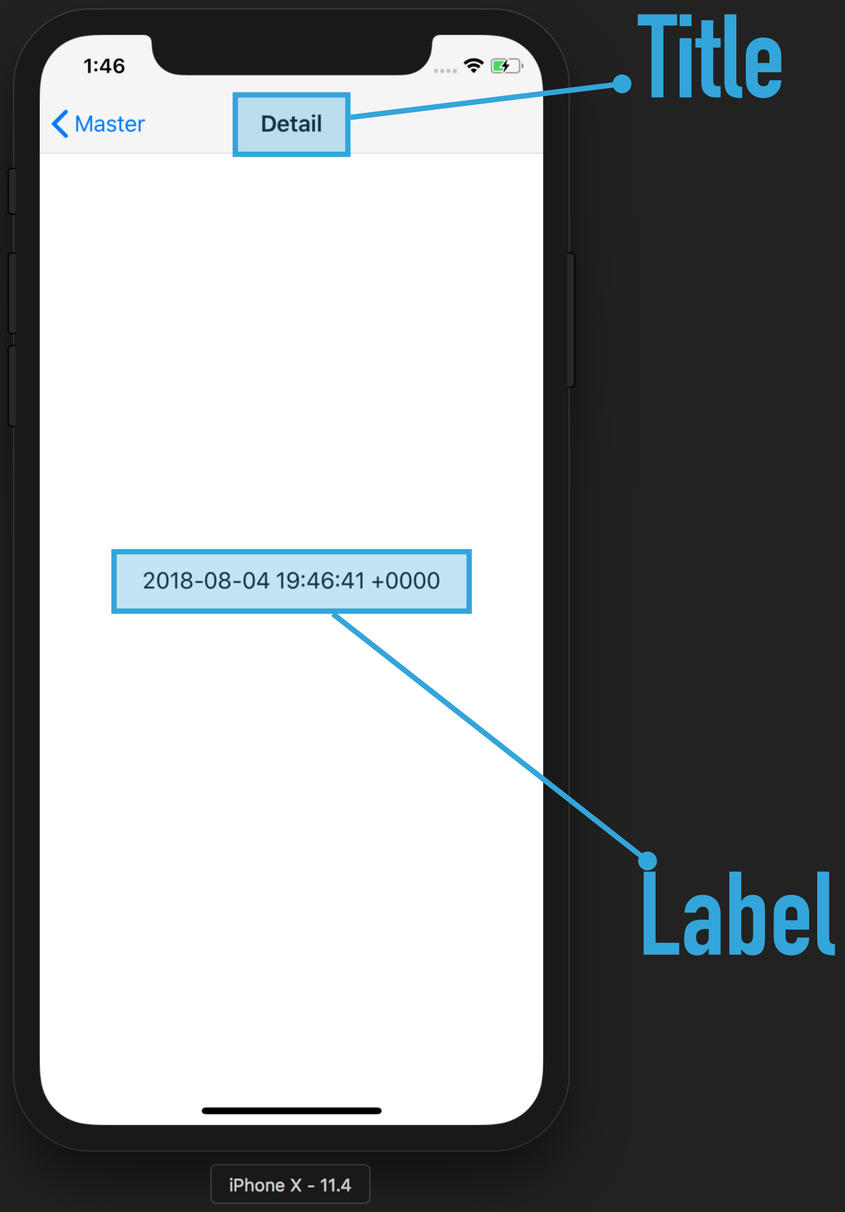
BETTER UI TESTING



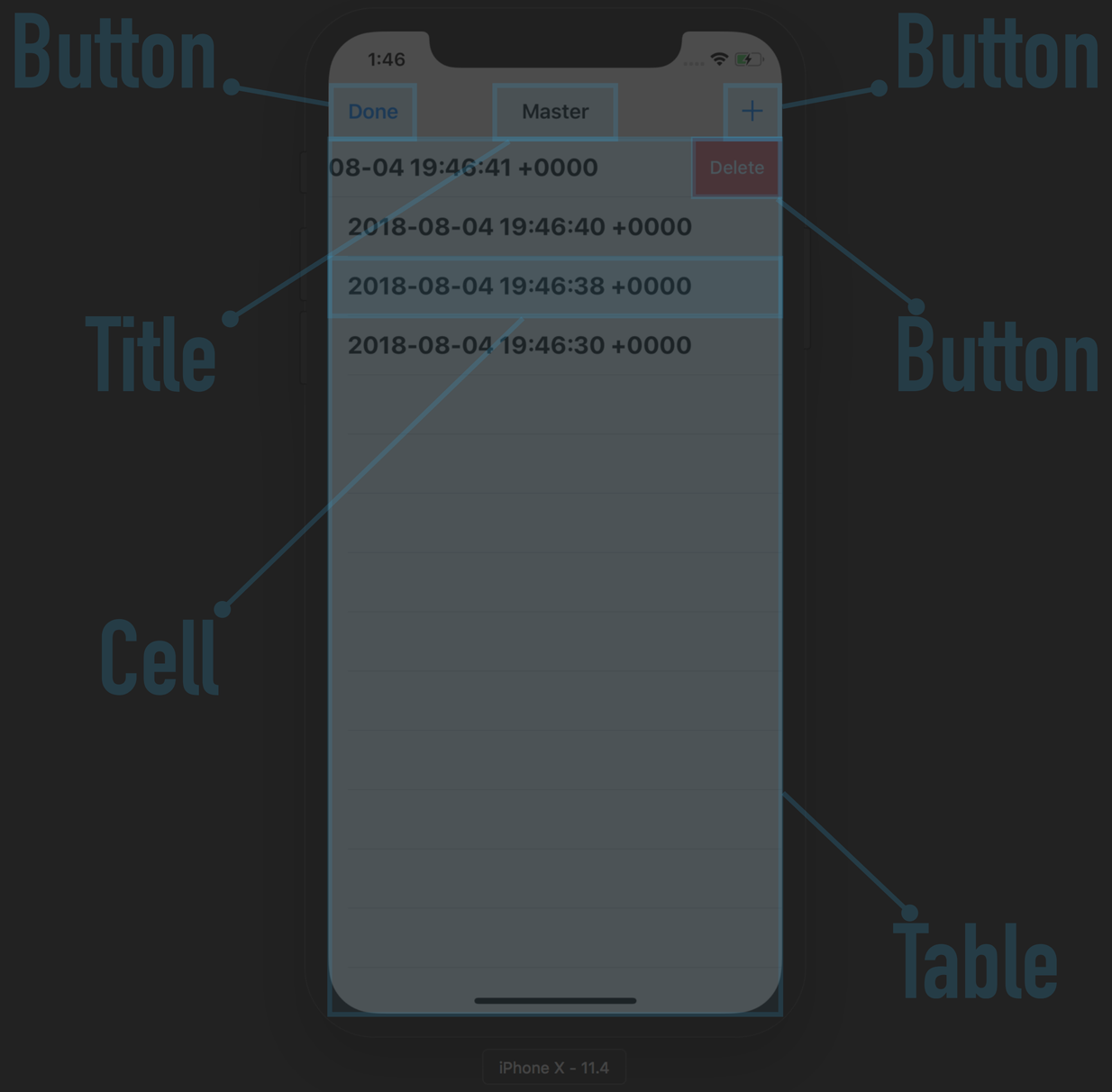


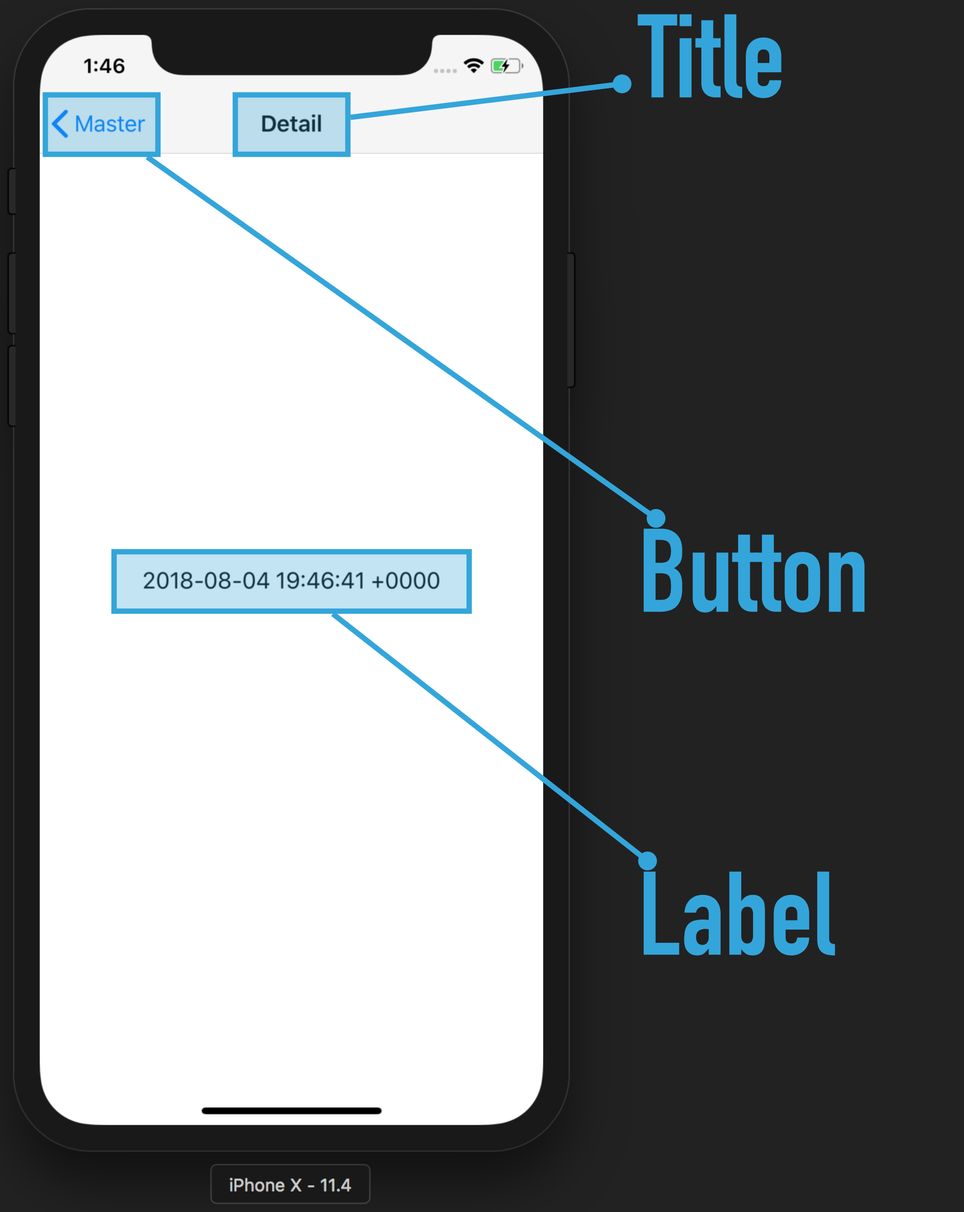
BETTER UI TESTING



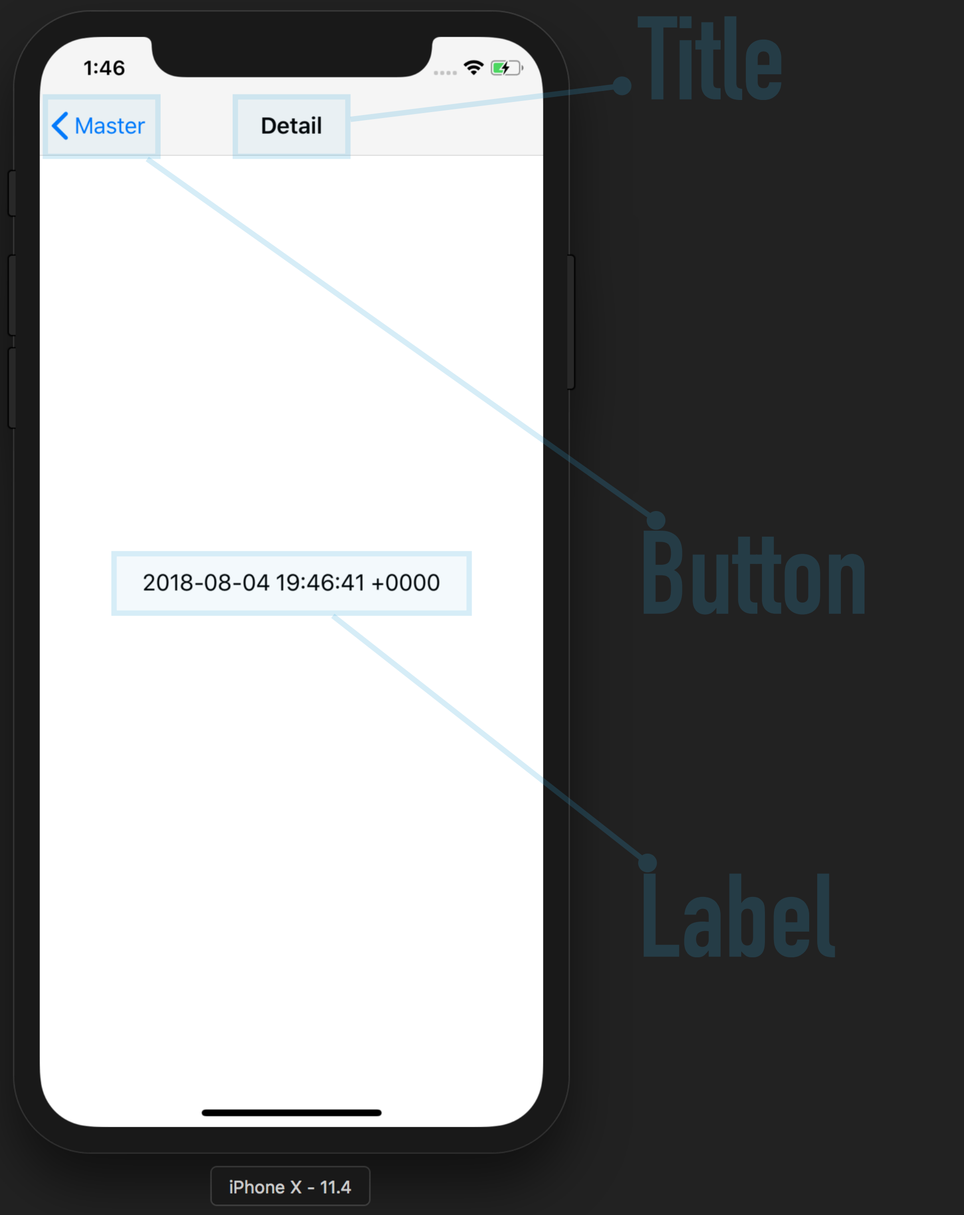


BETTER UI TESTING





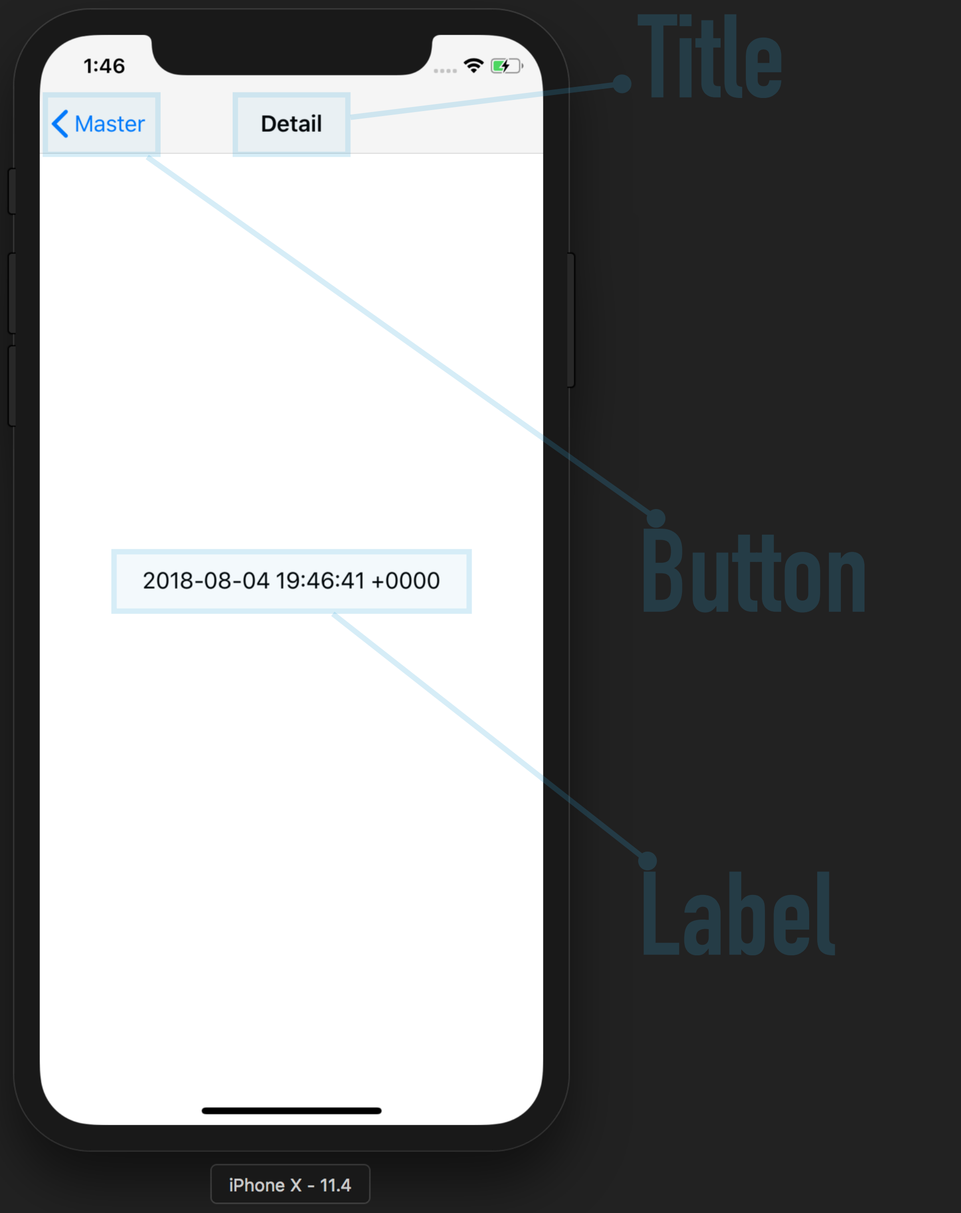
BETTER UI TESTING



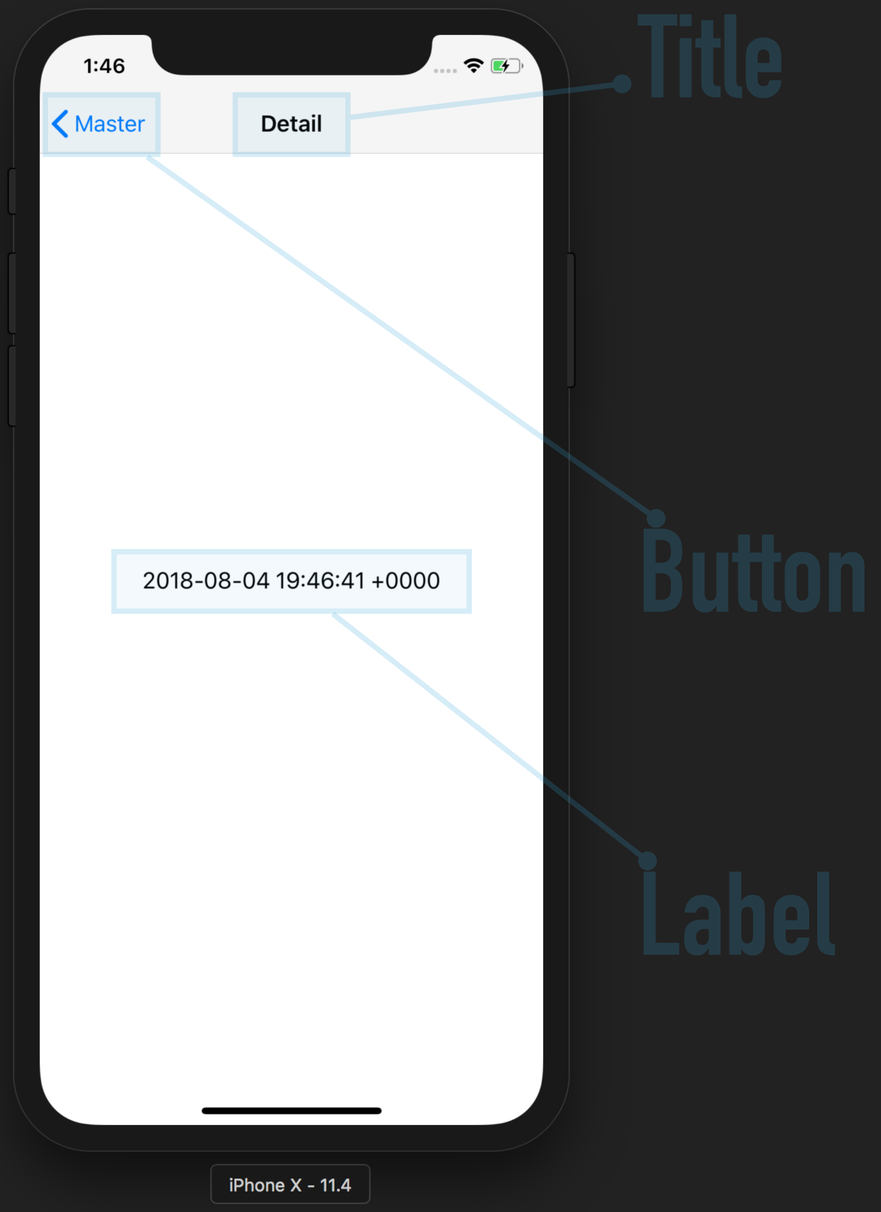
BETTER UI TESTING

BETTER UI TESTING

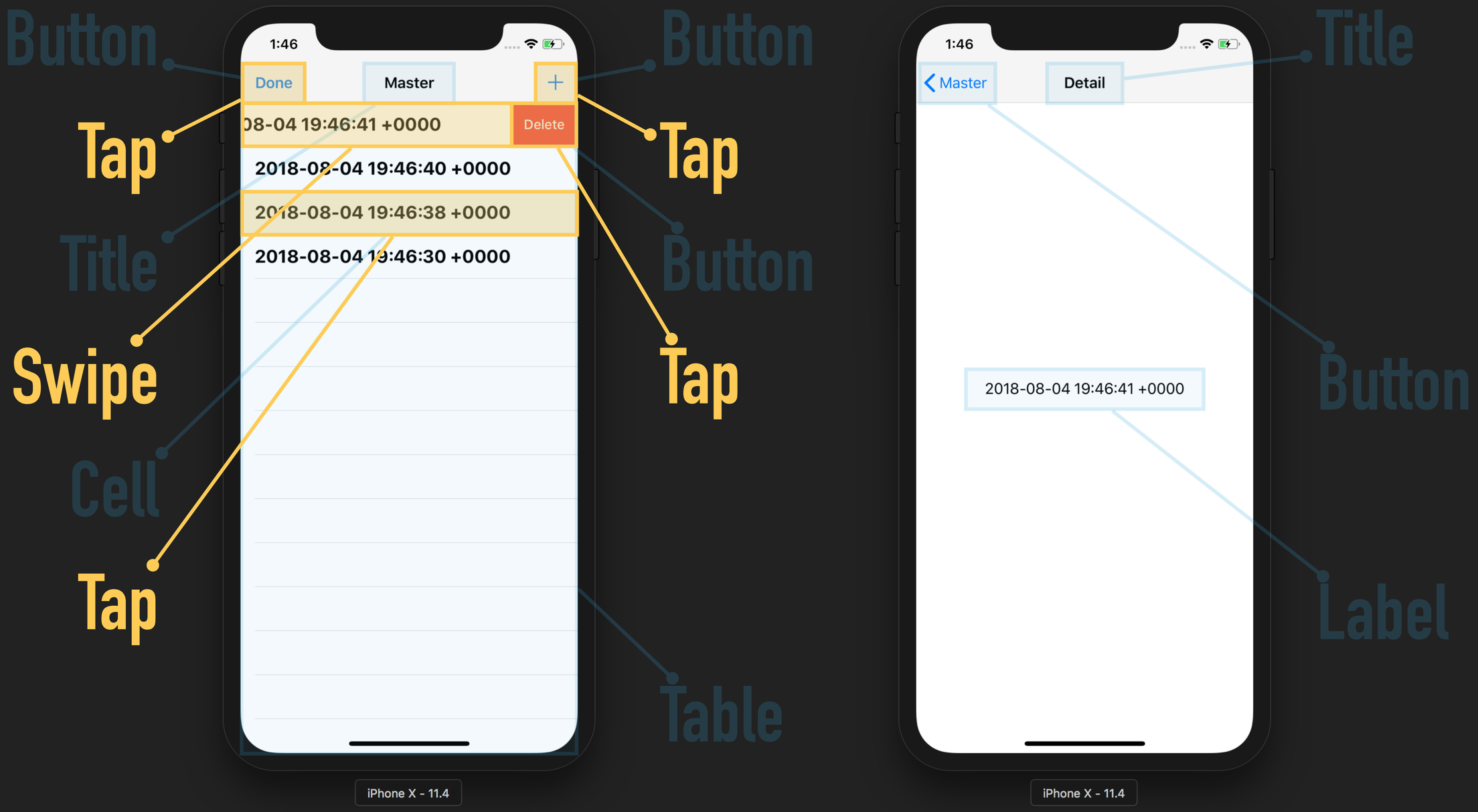
BETTER UI TESTING



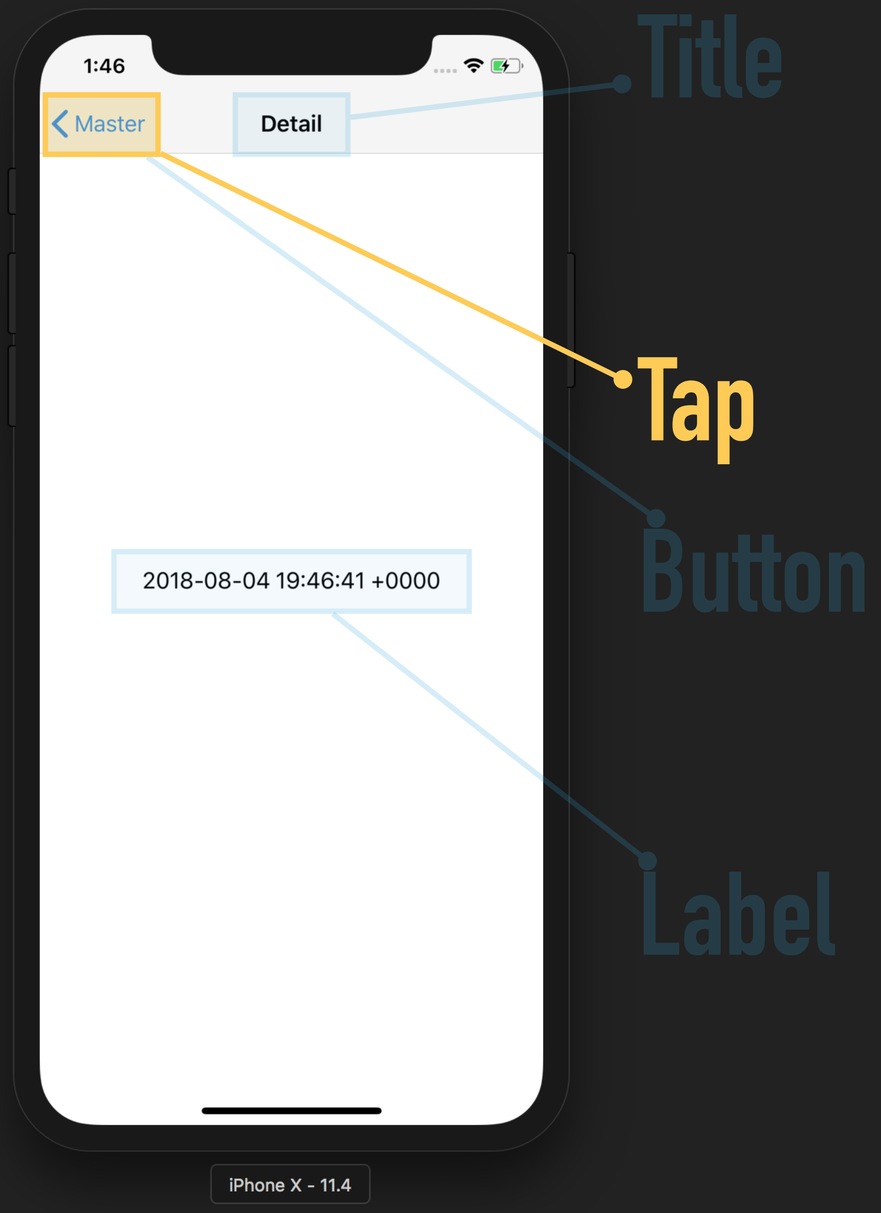
BETTER UI TESTING



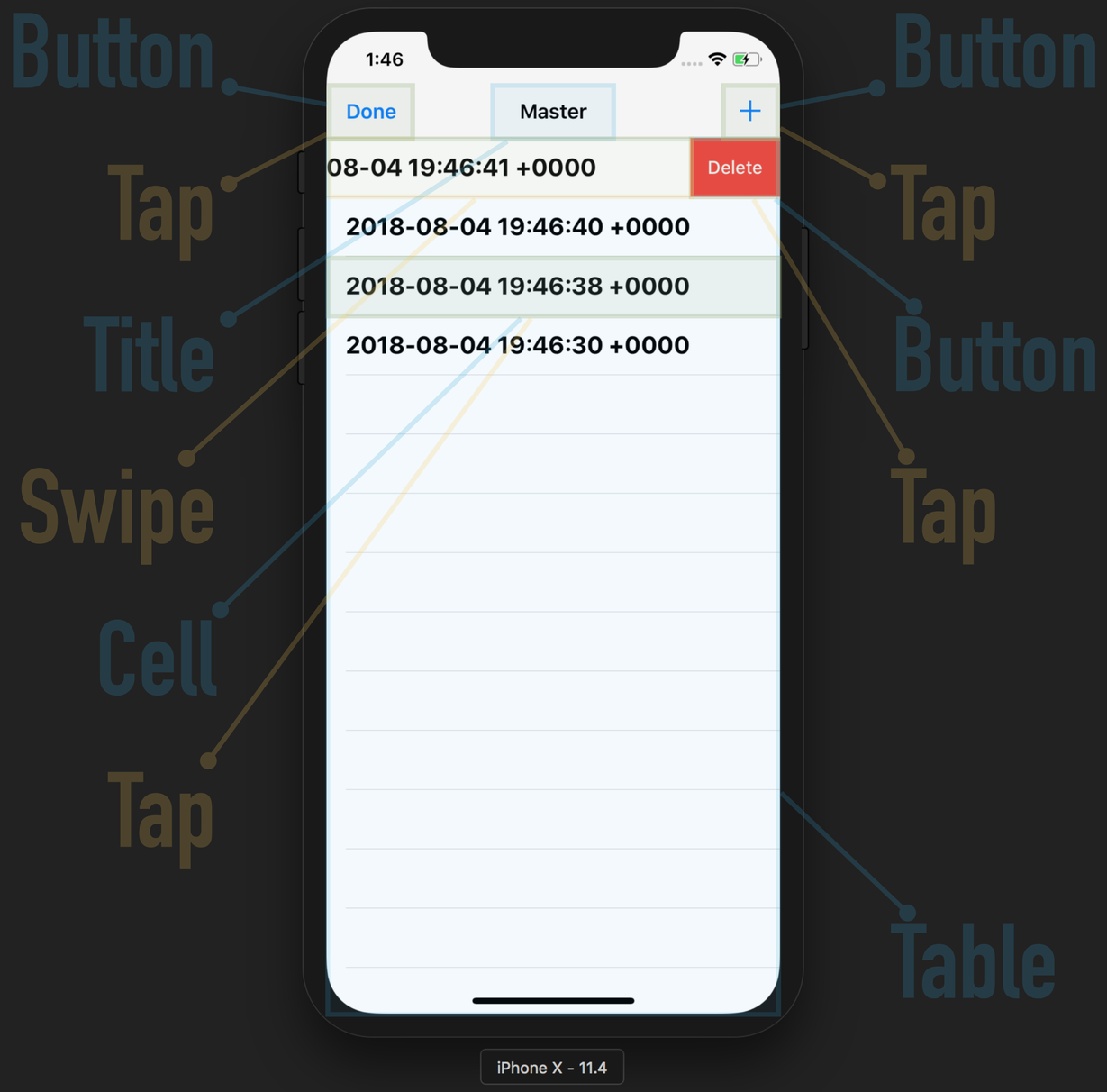
BETTER UI TESTING

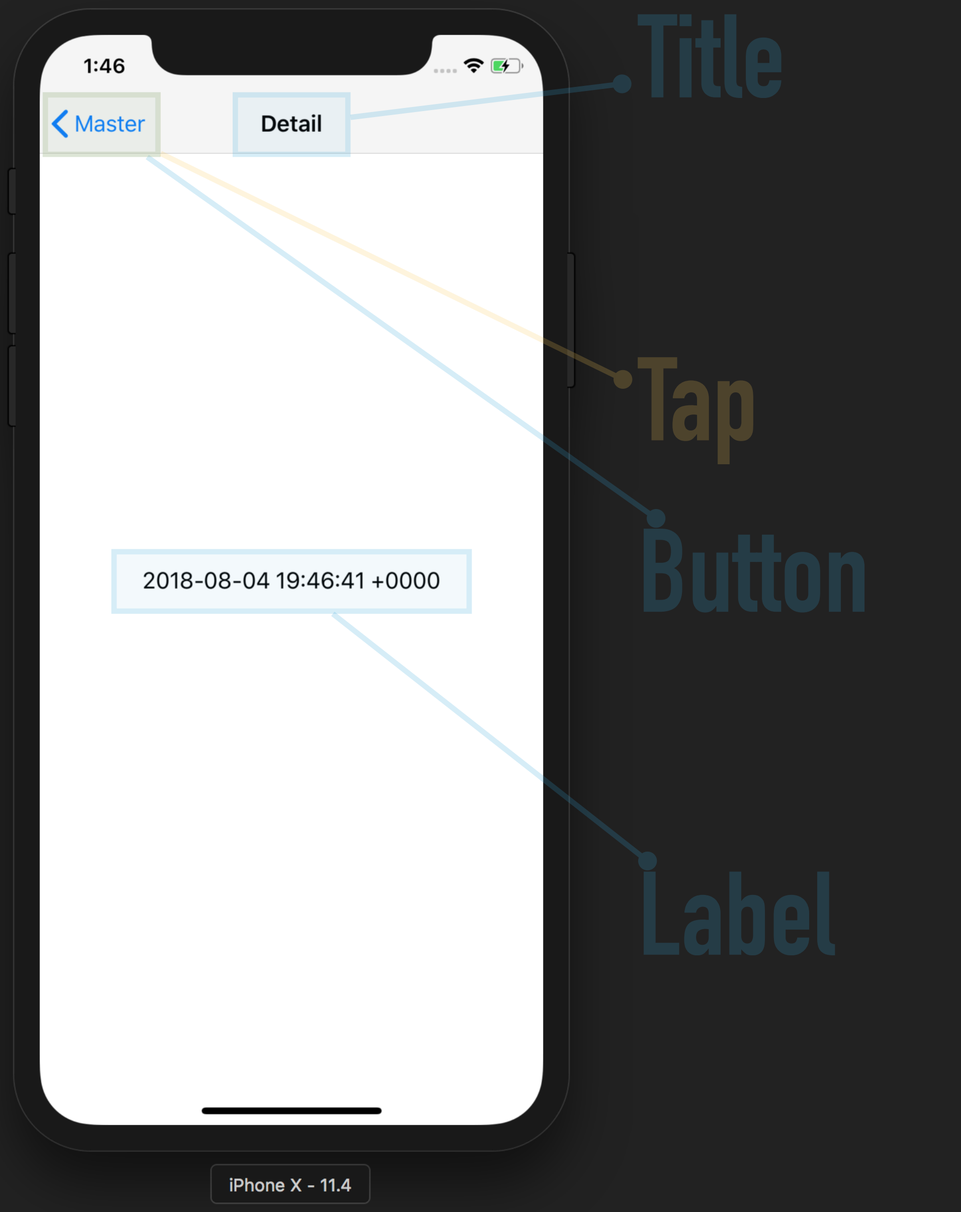


BETTER UI TESTING

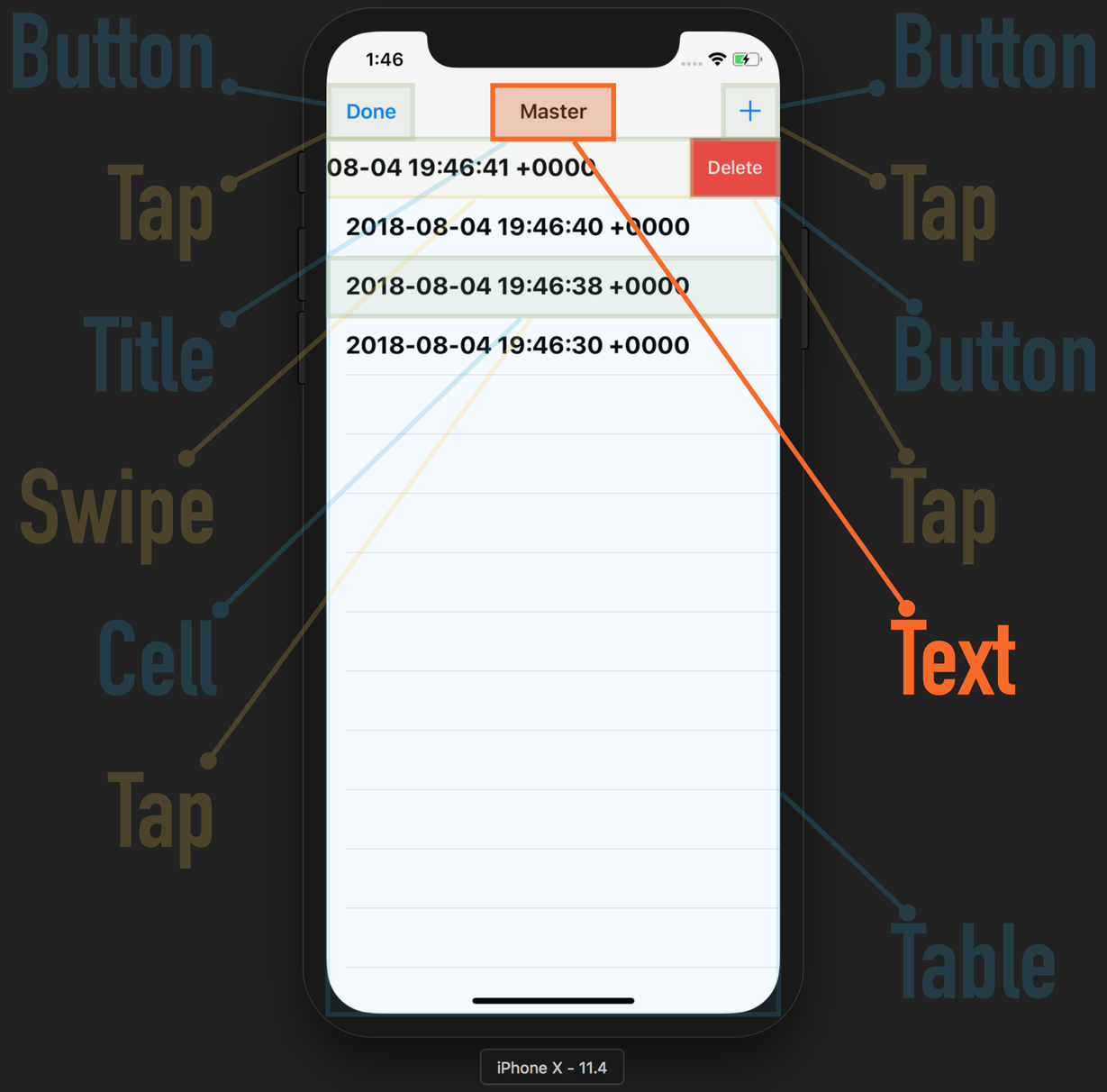


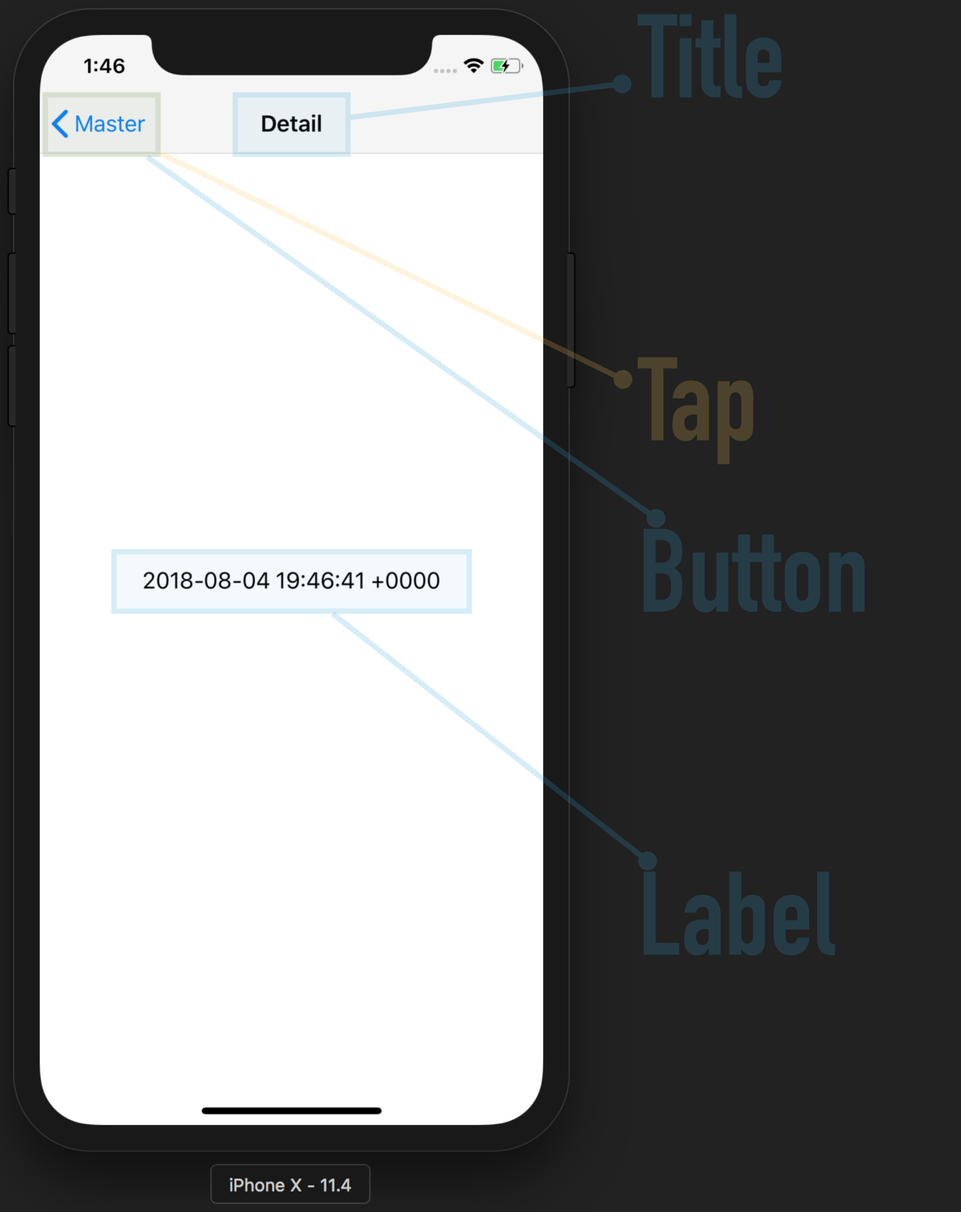
BETTER UI TESTING



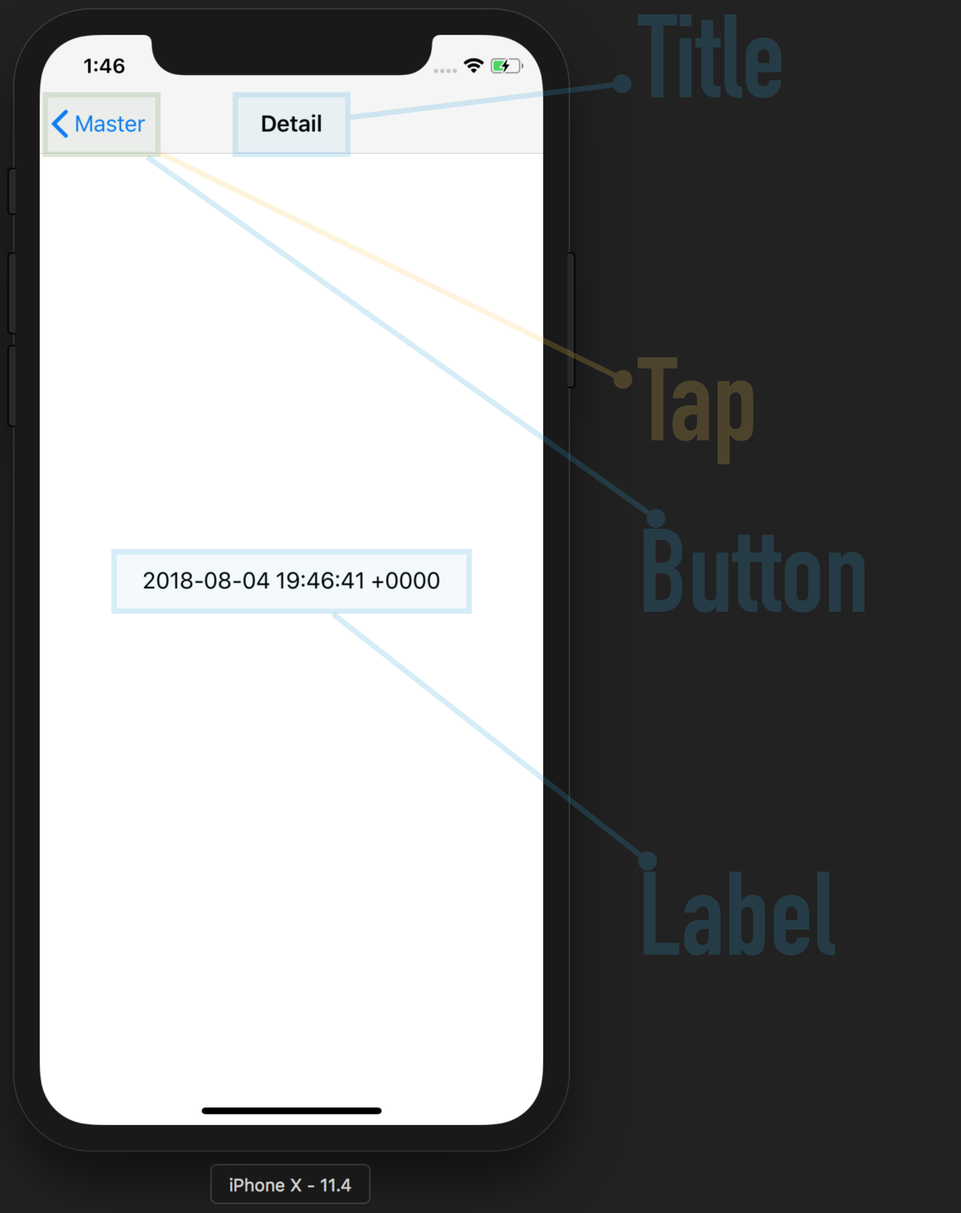


BETTER UI TESTING

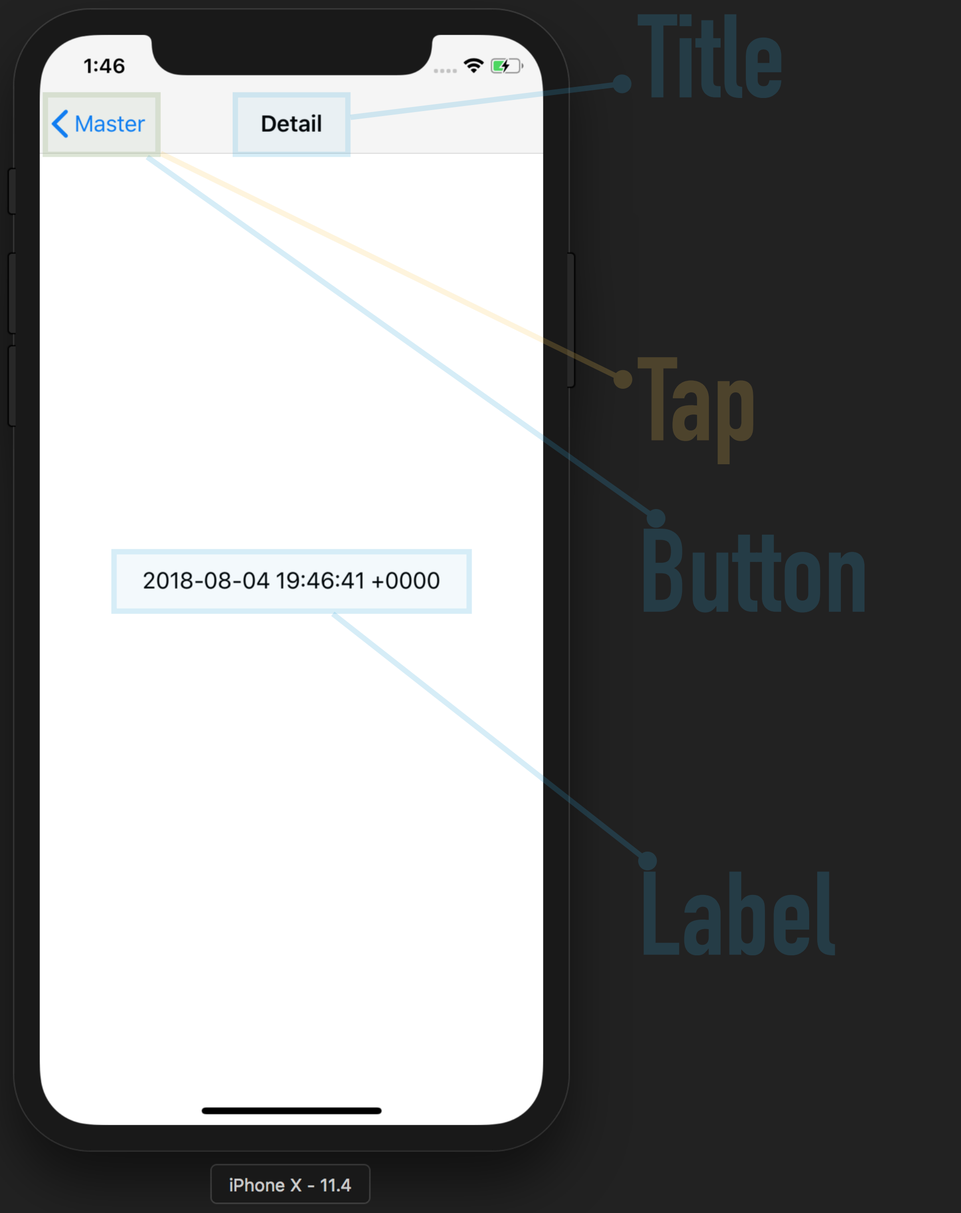




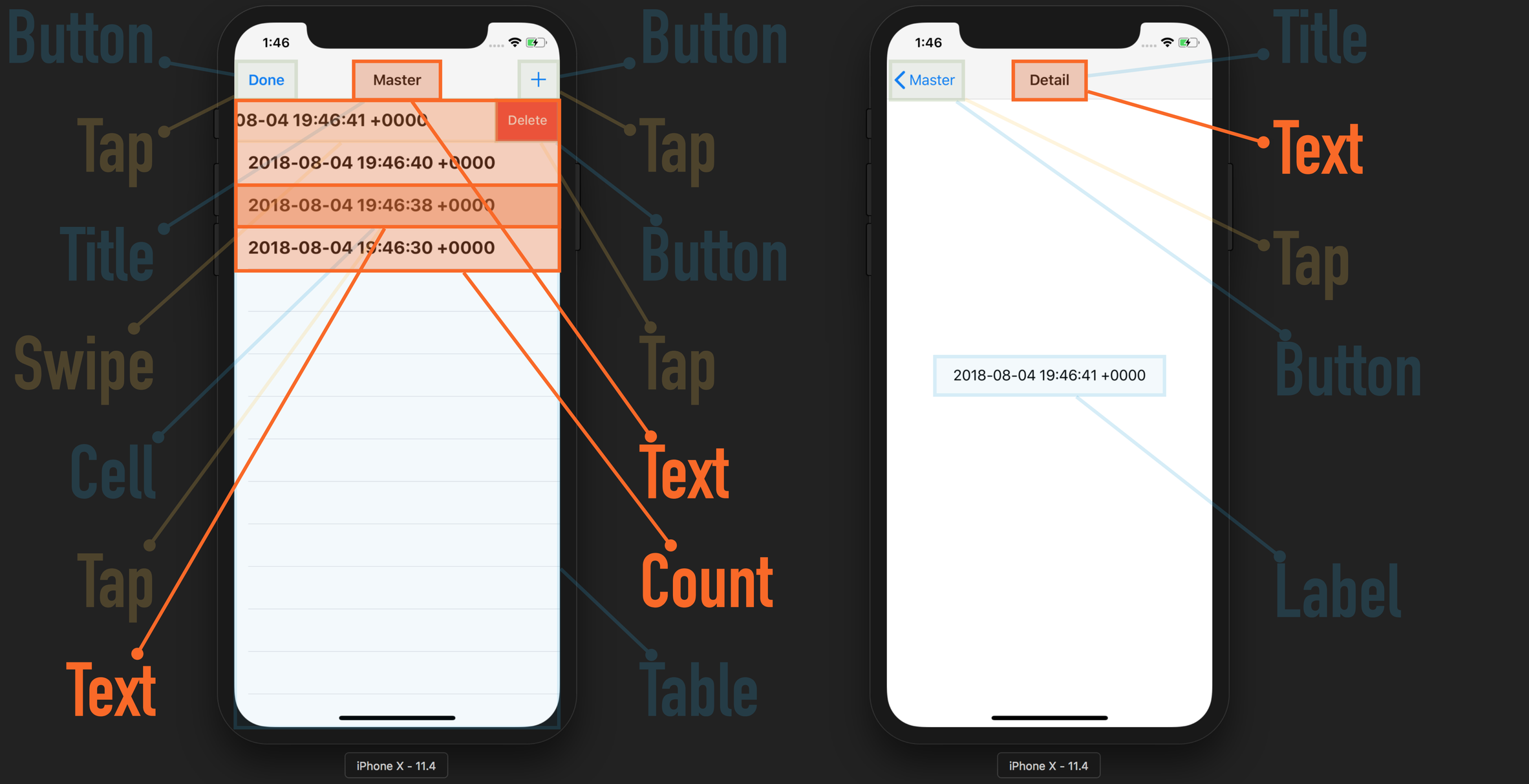
BETTER UI TESTING



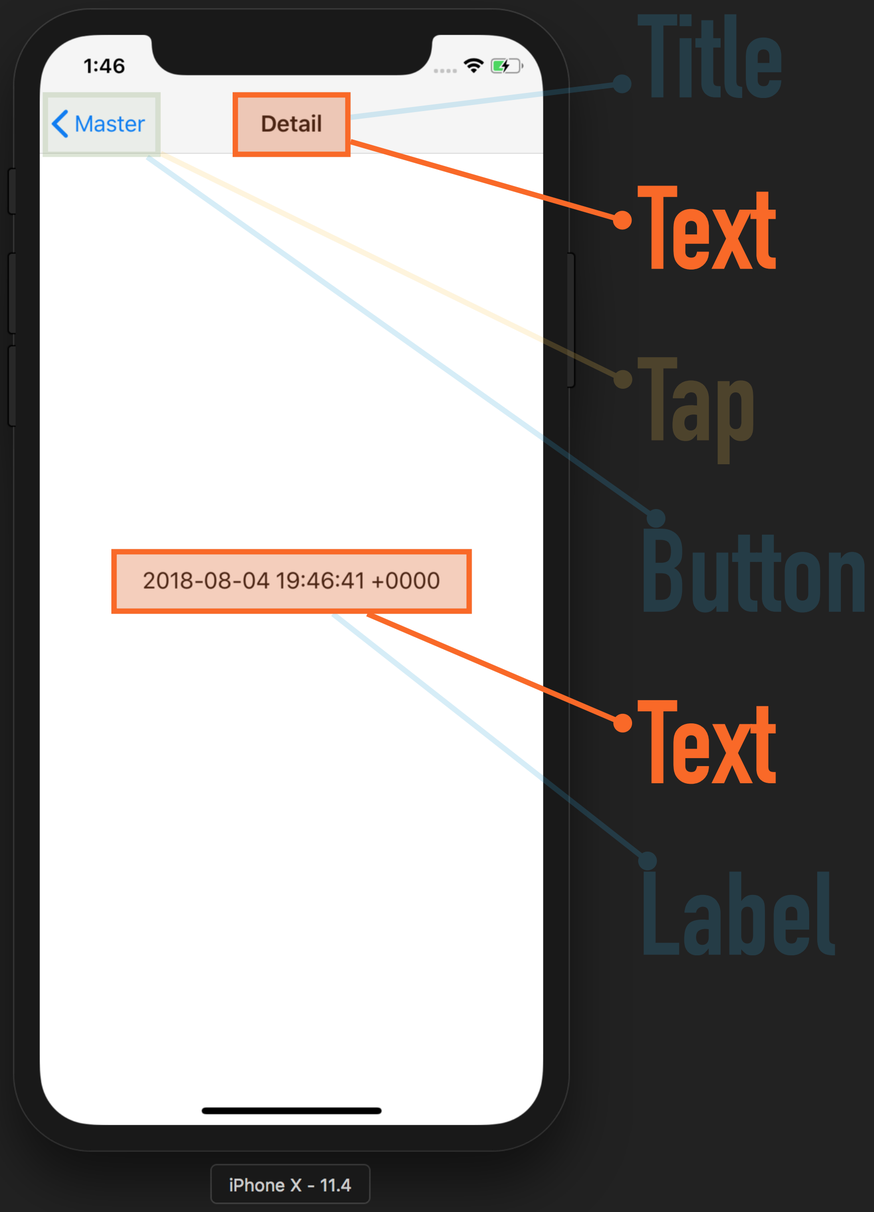
BETTER UI TESTING



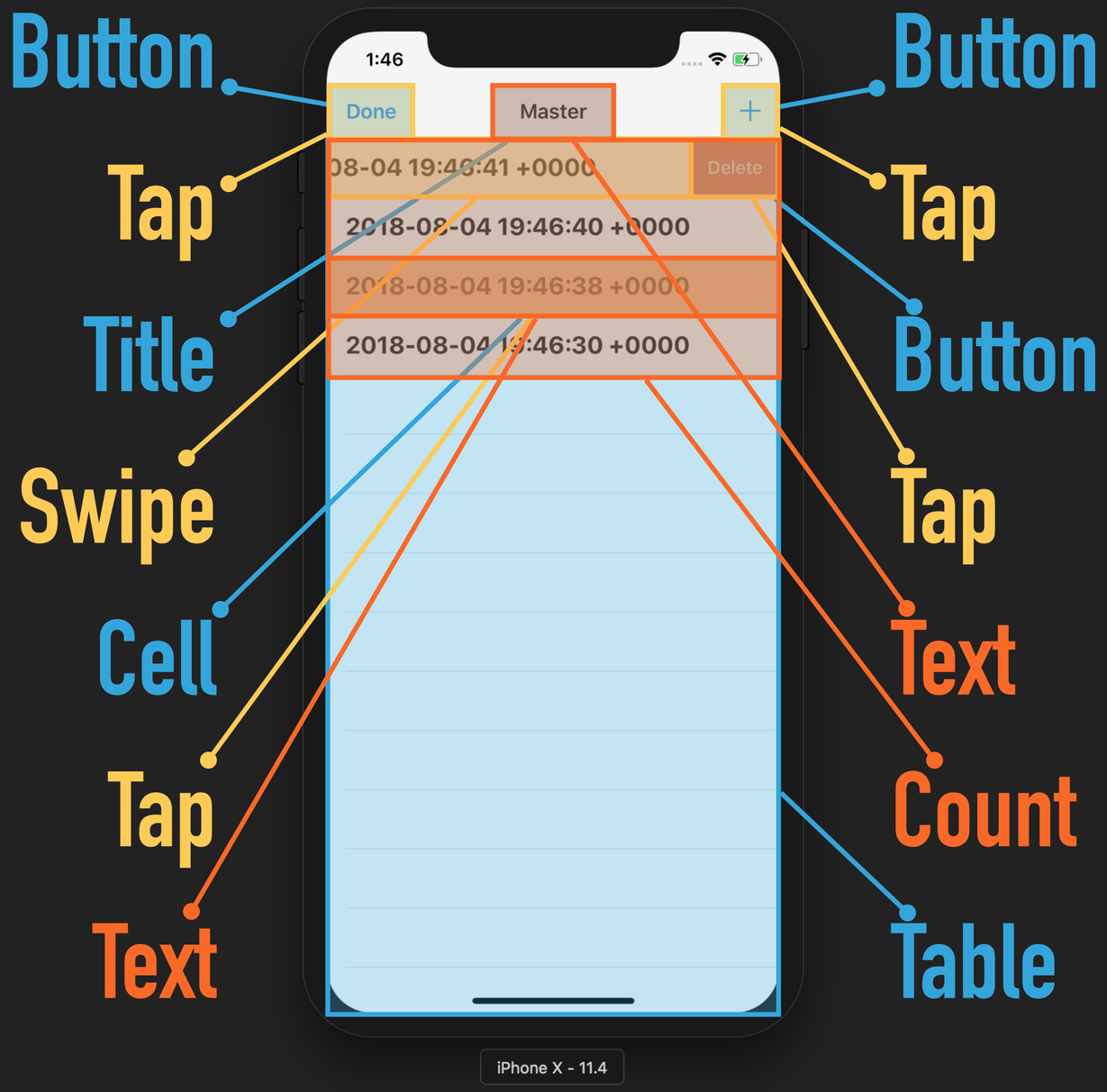
BETTER UI TESTING

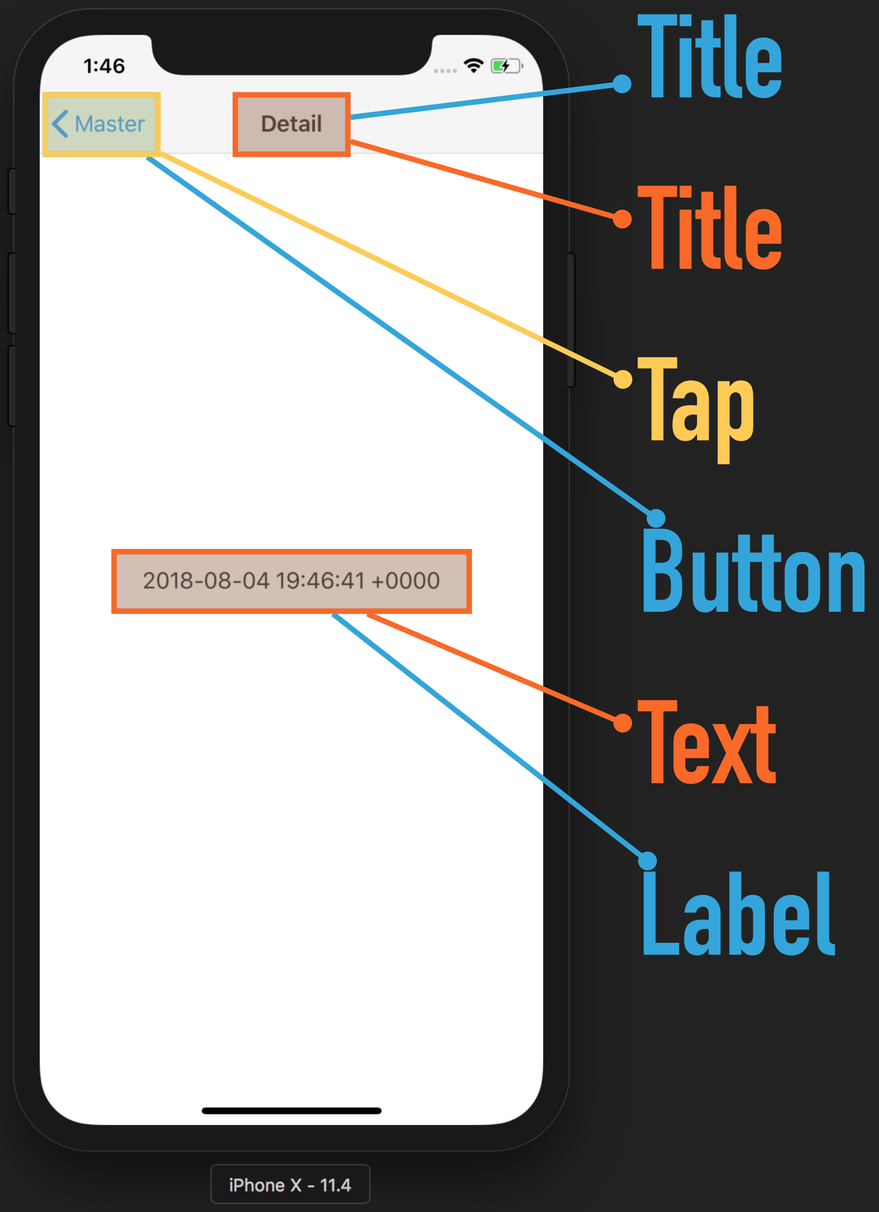


BETTER UI TESTING

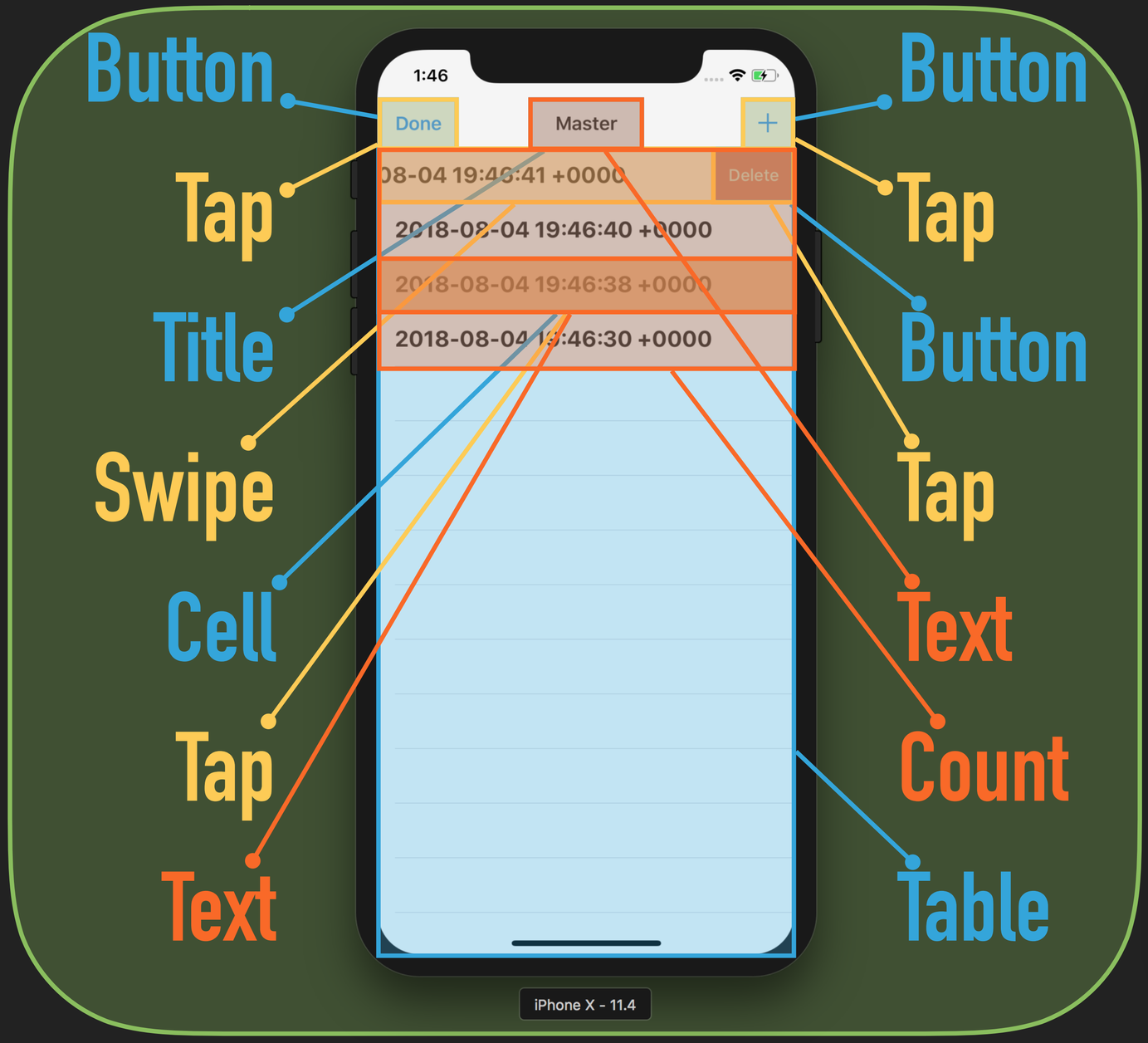


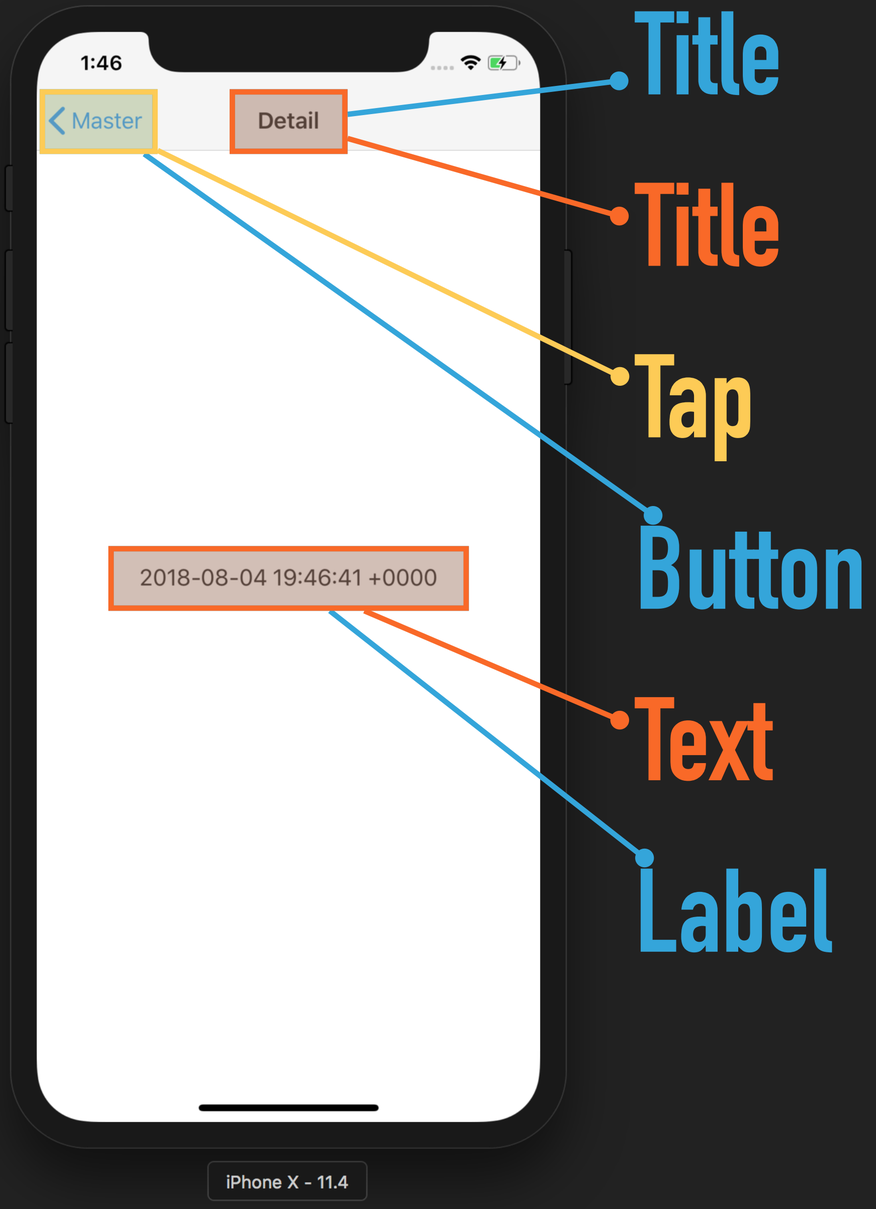
BETTER UI TESTING



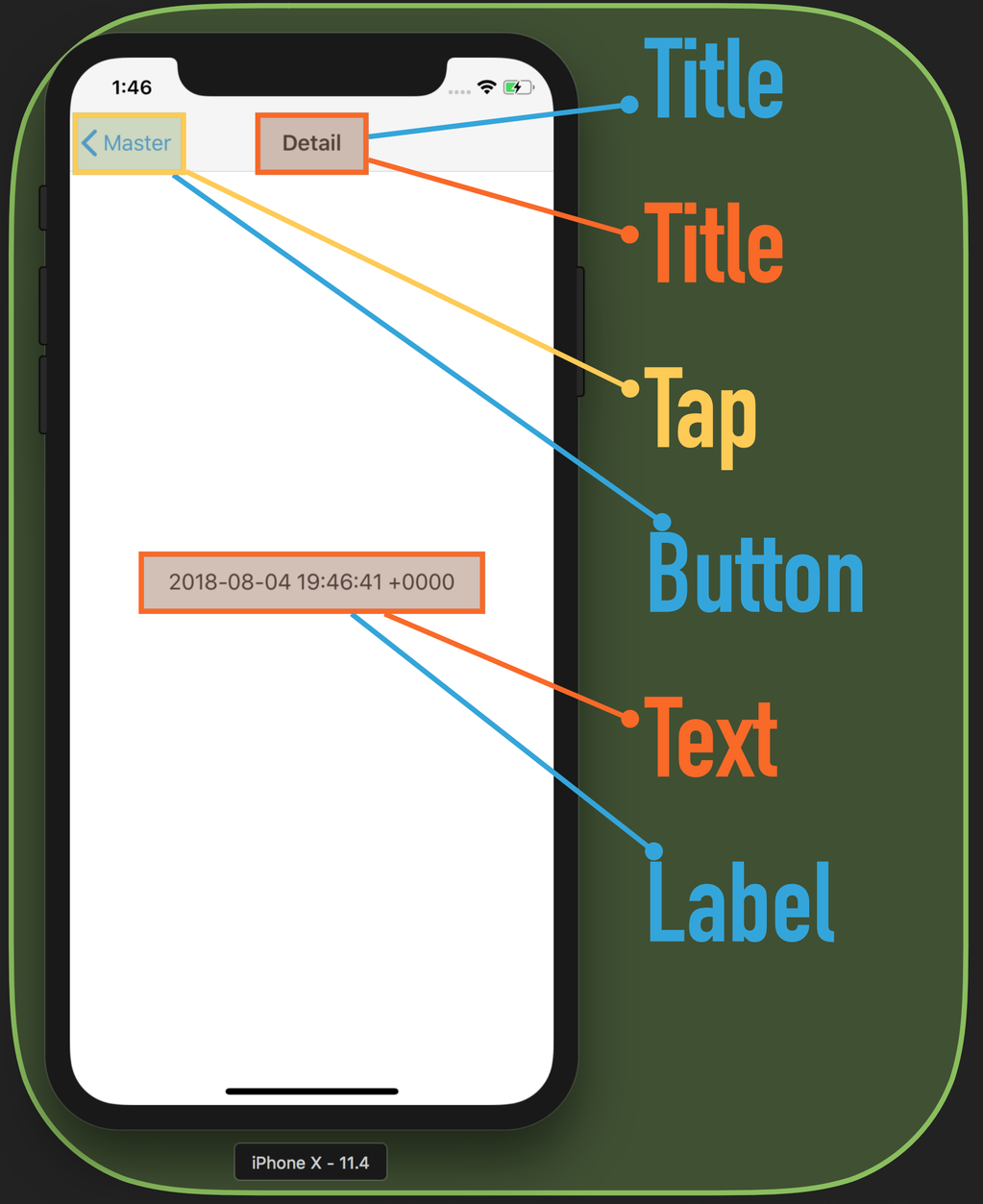


BETTER UI TESTING





BETTER UI TESTING



BETTER UI TESTING

BENEFITS

EASIER TO READ

BENEFITS

EASIER TO READ

Before

```
XCUIApplication().tables.children(matching: .cell).element(boundBy: 0).staticTexts["2018-07-03 20:45:14 +0000"].tap()
```

BENEFITS

EASIER TO READ

Before

```
XCUIApplication().tables.children(matching: .cell).element(boundBy: 0).staticTexts["2018-07-03 20:45:14 +0000"].tap()
```

After

```
.tapOnCell(at: 0)
```

BENEFITS

EASIER TO READ

EASIER TO MAINTAIN

BENEFITS

EASIER TO READ

EASIER TO MAINTAIN

WRITE TESTS FIRST!

BENEFITS

TDD

EASIER TO READ

EASIER TO MAINTAIN

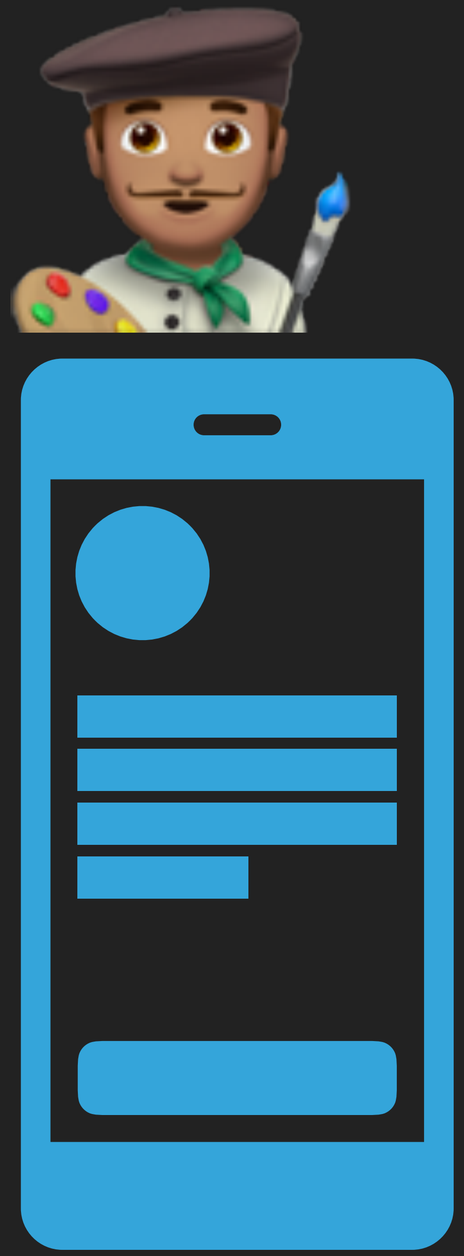
WRITE TESTS FIRST!

BENEFITS

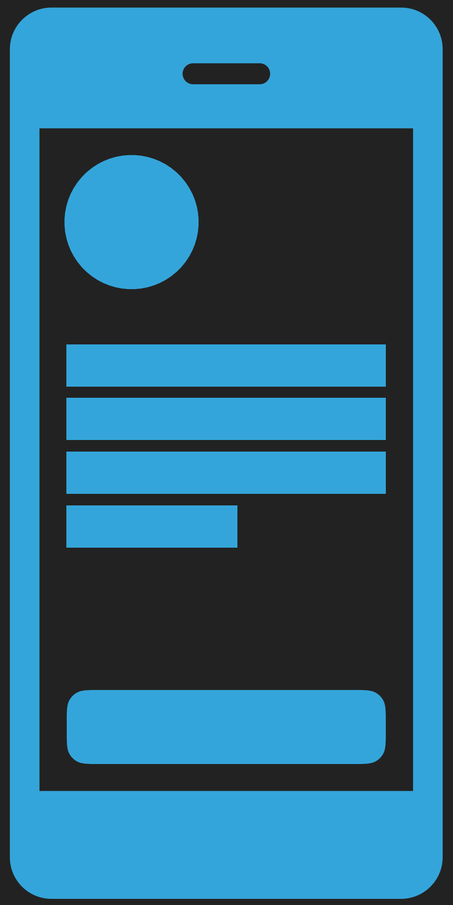
TDD

TEST-DRIVEN DEVELOPMENT

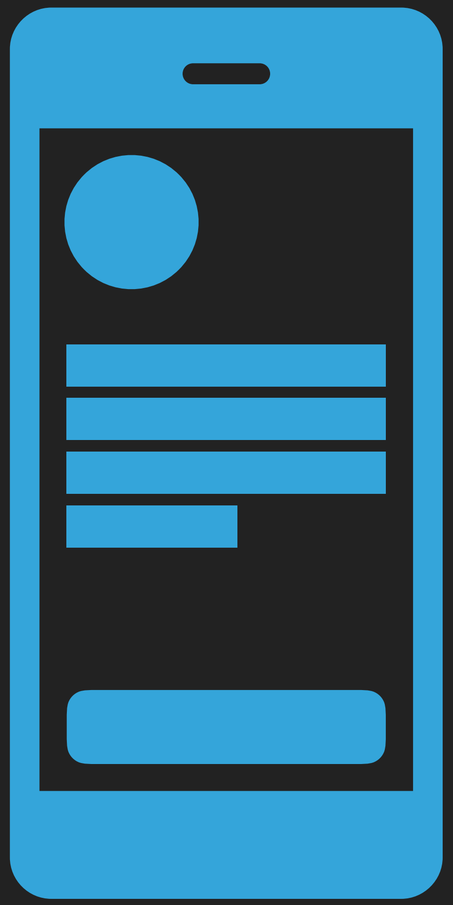
TDD

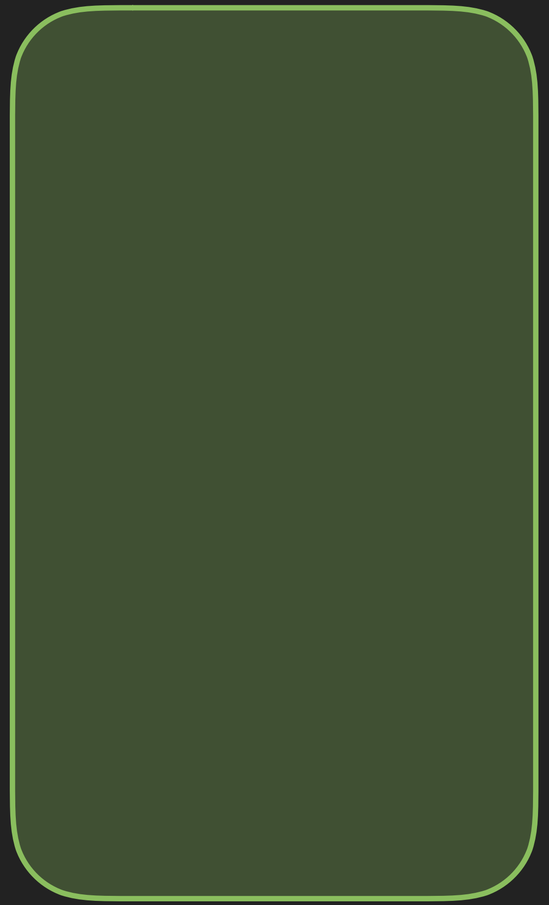


TDD

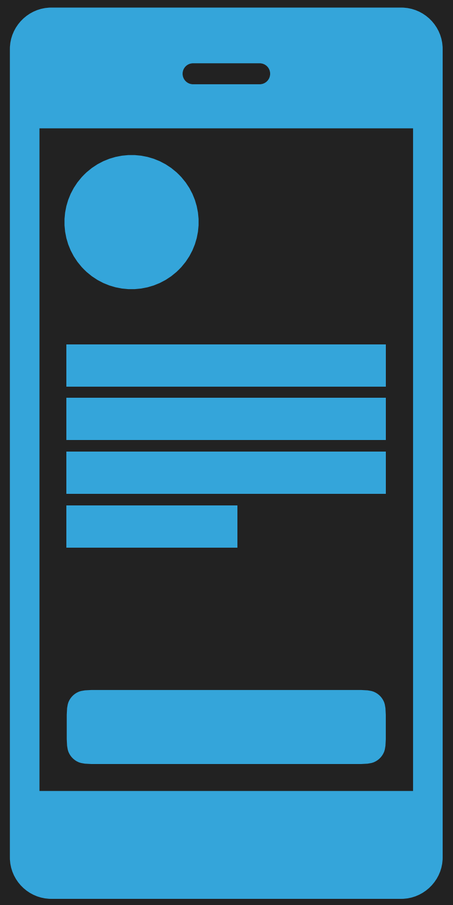


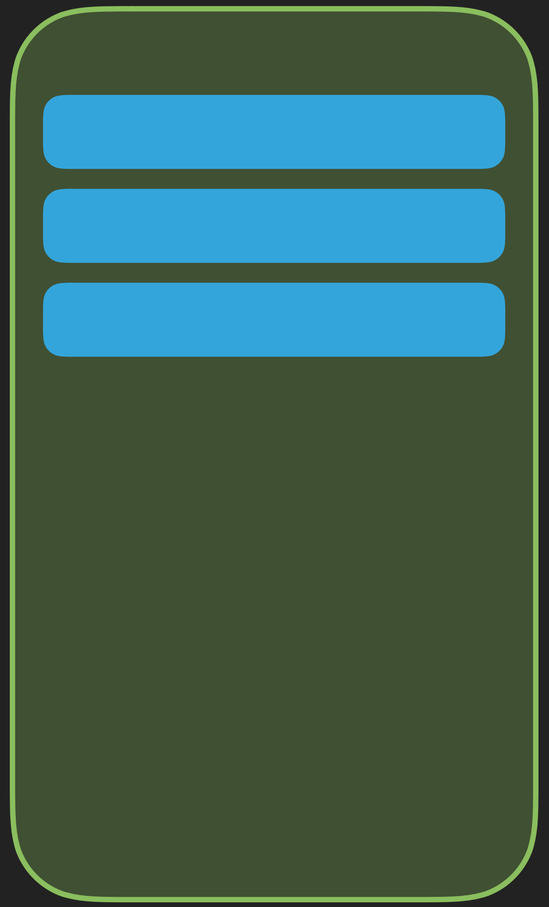
TDD



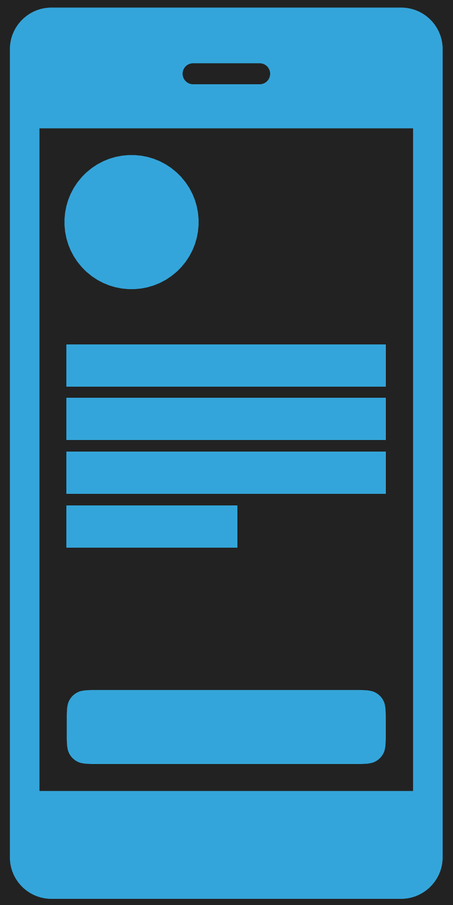


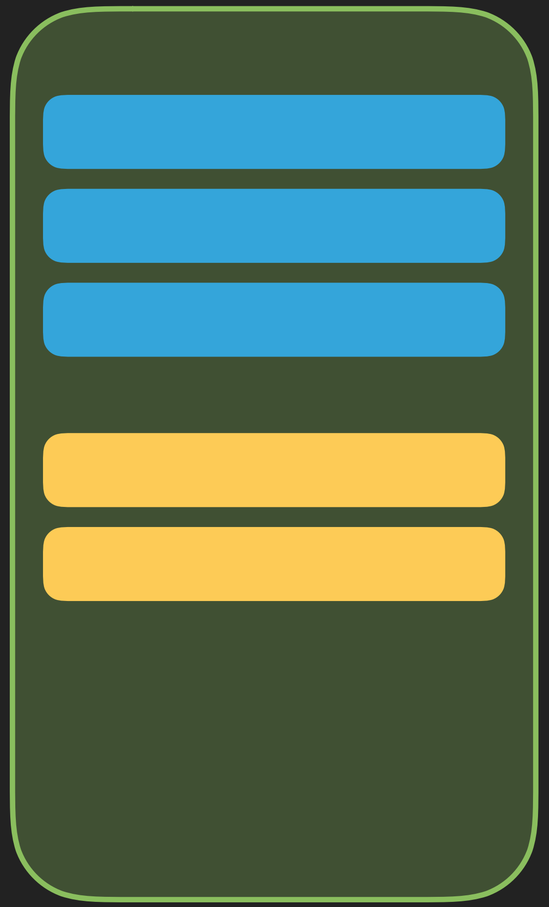
TDD



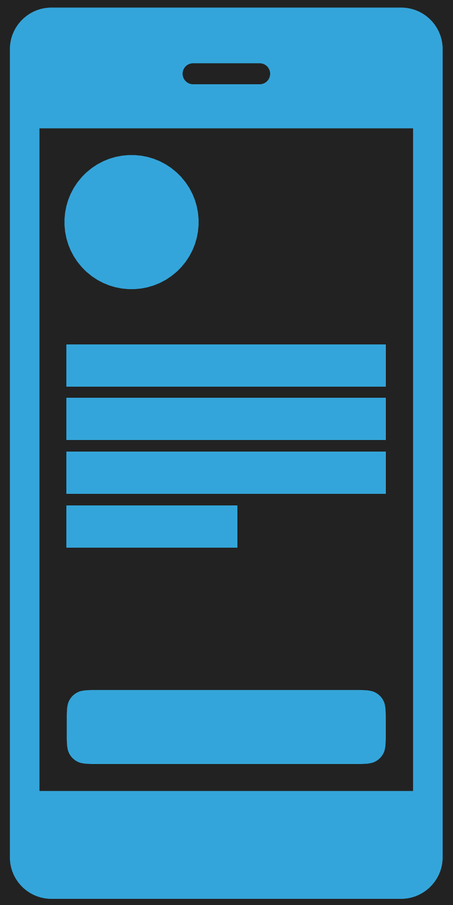


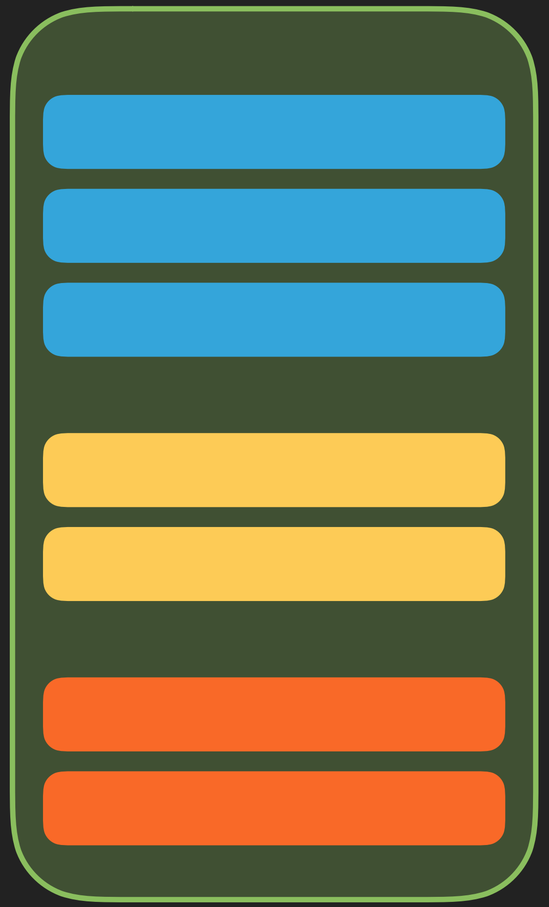
TDD



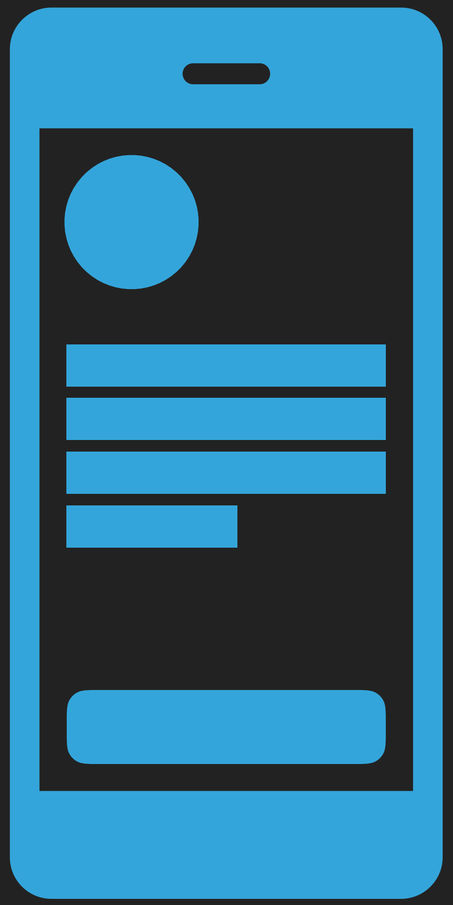


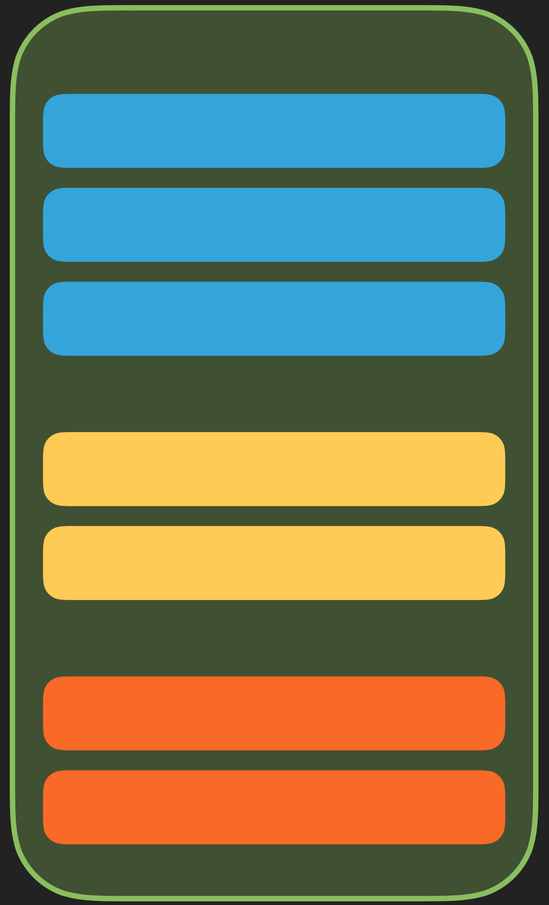
TDD



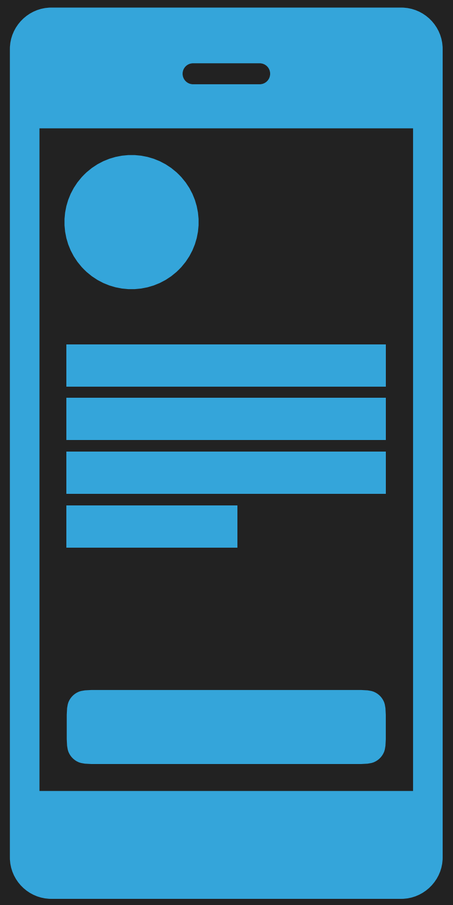


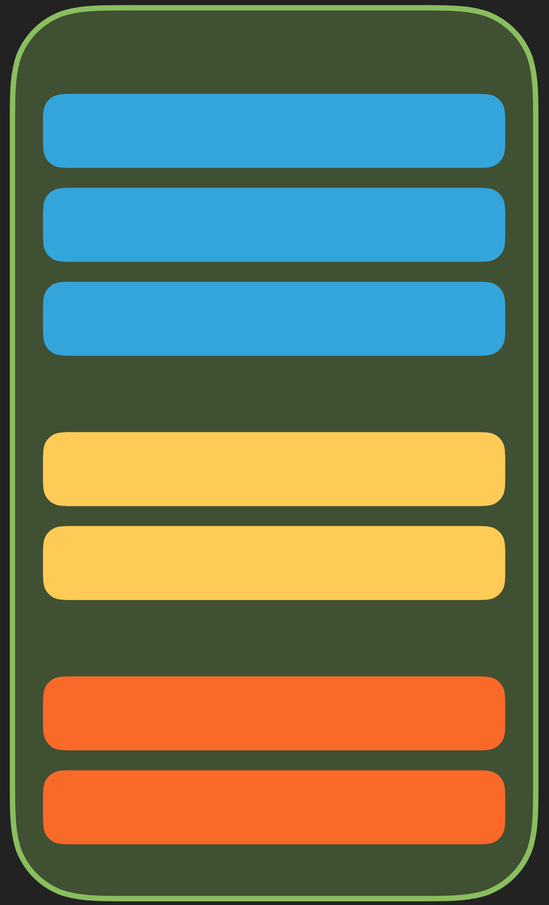
TDD

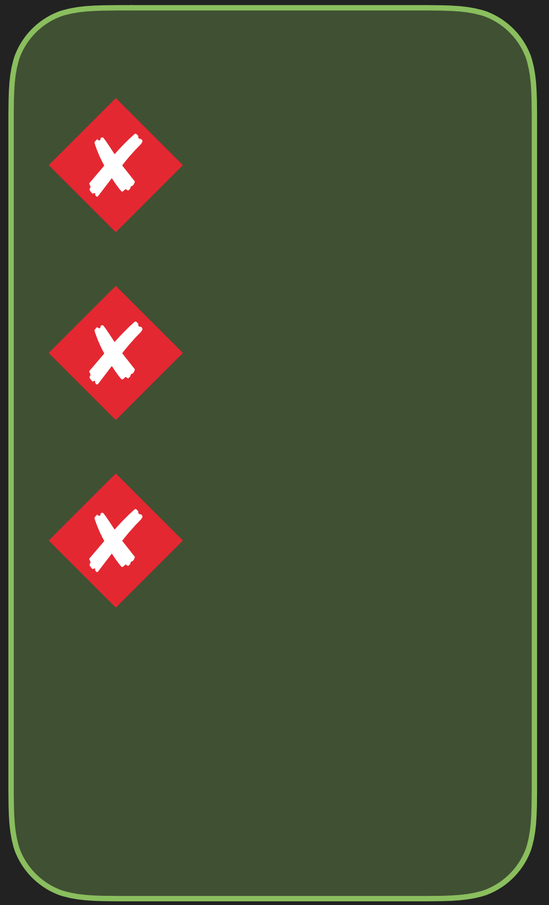




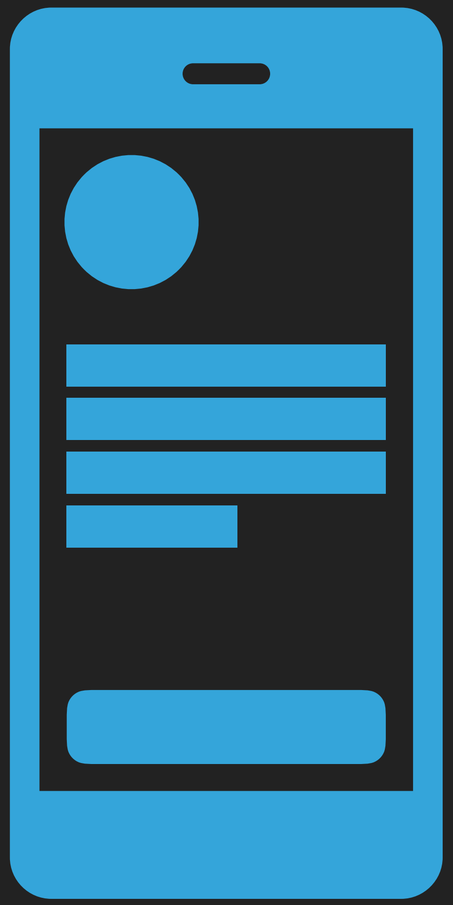
TDD

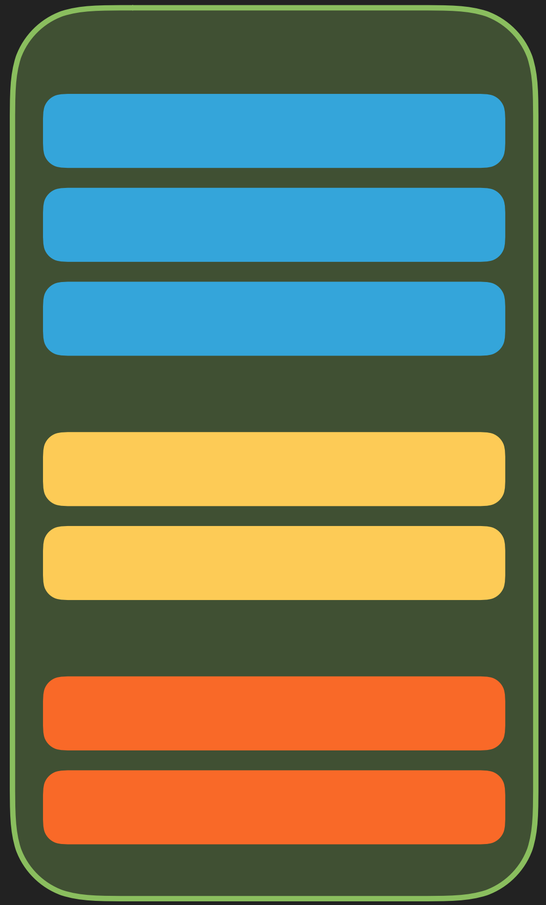


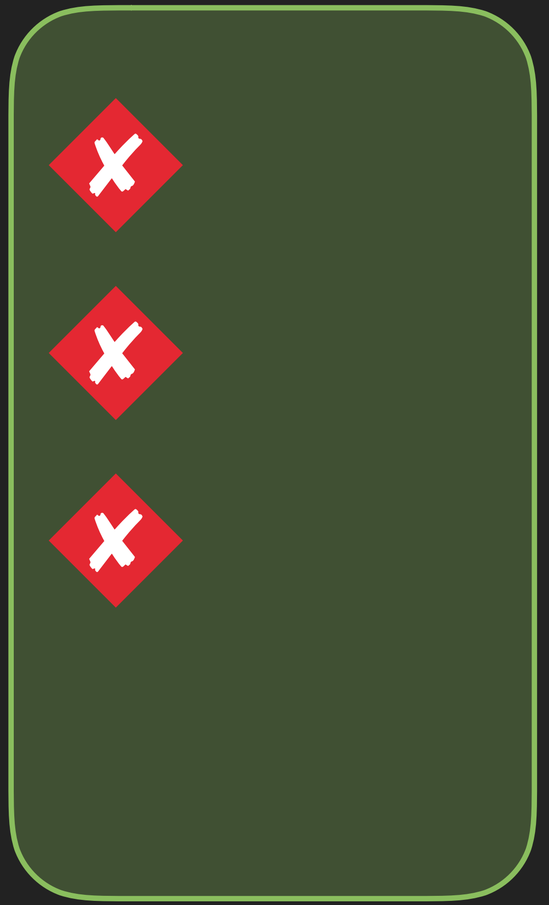


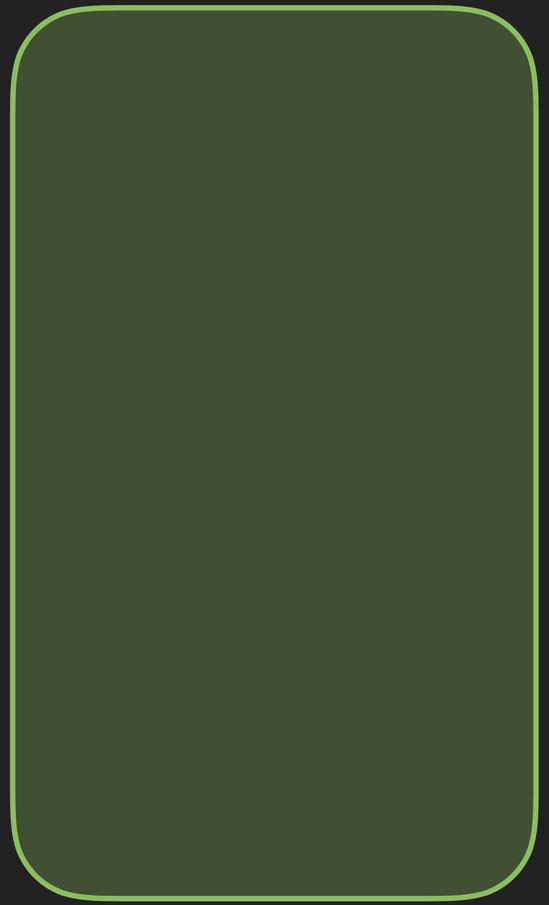


TDD

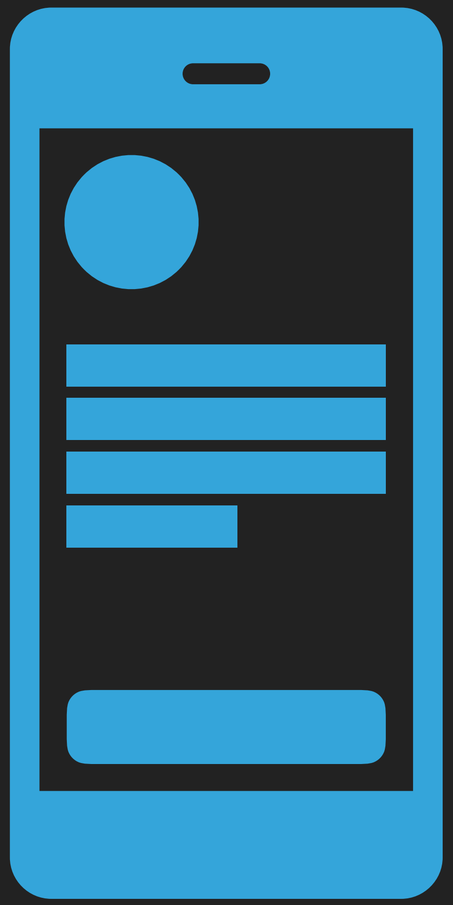


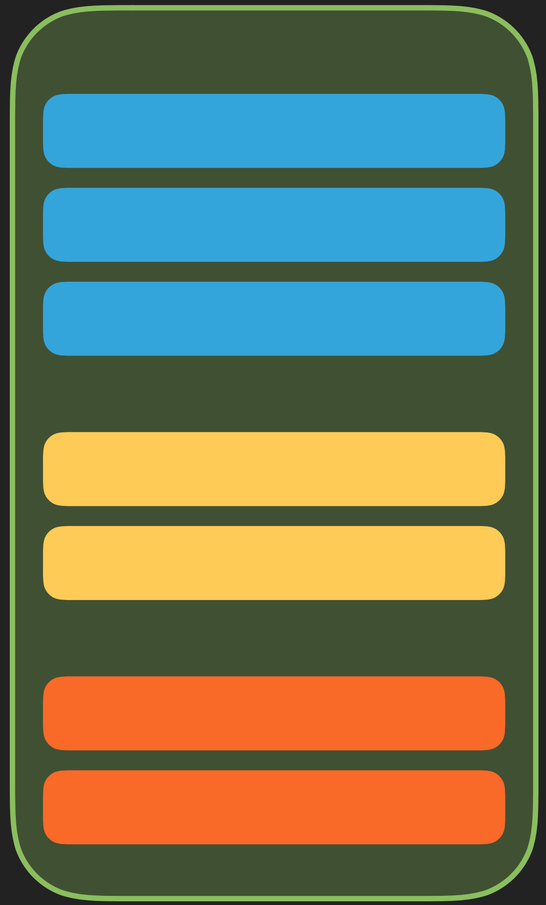


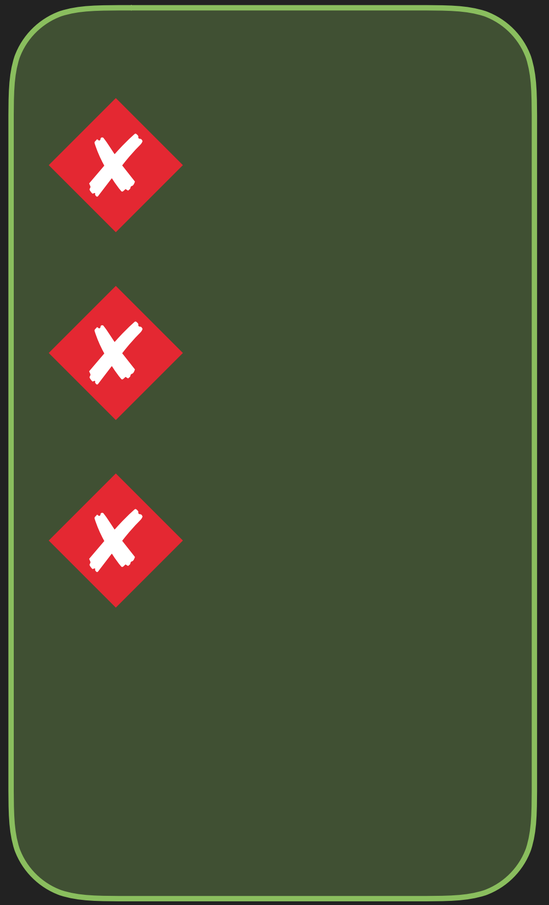


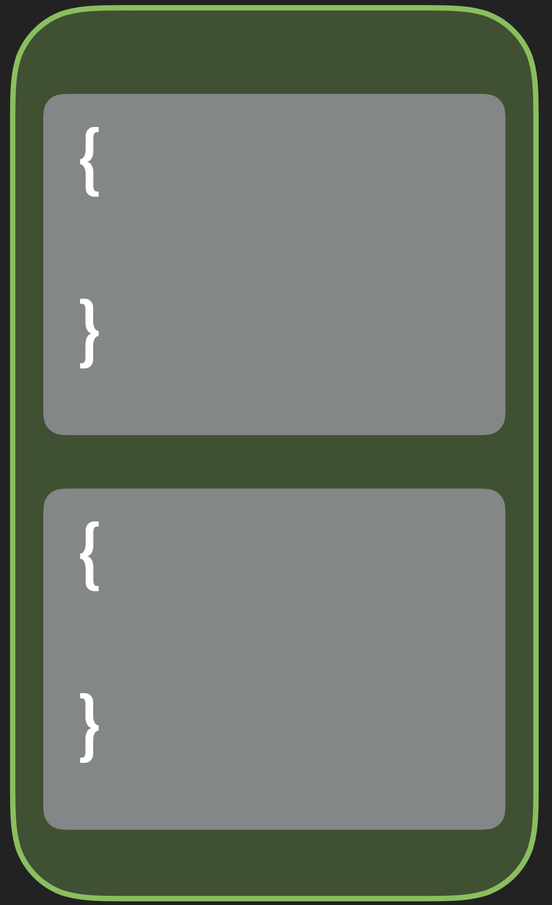


TDD

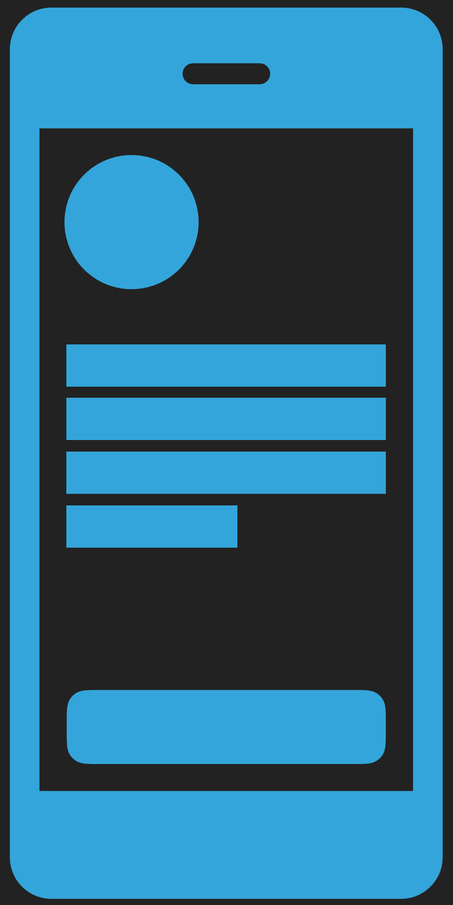


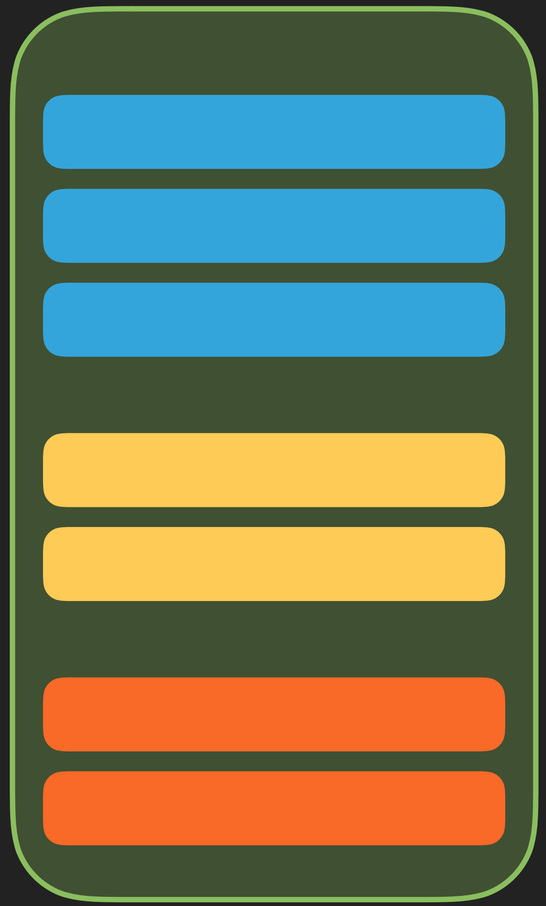


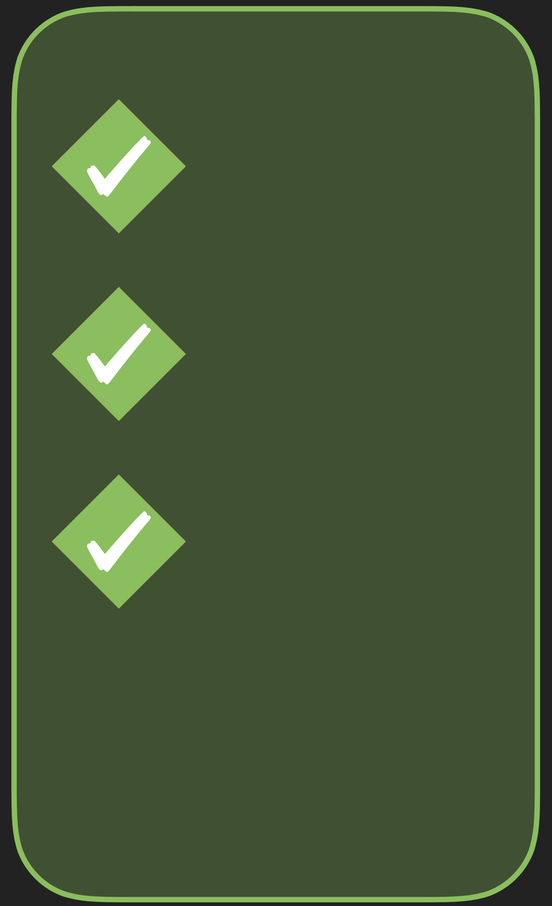


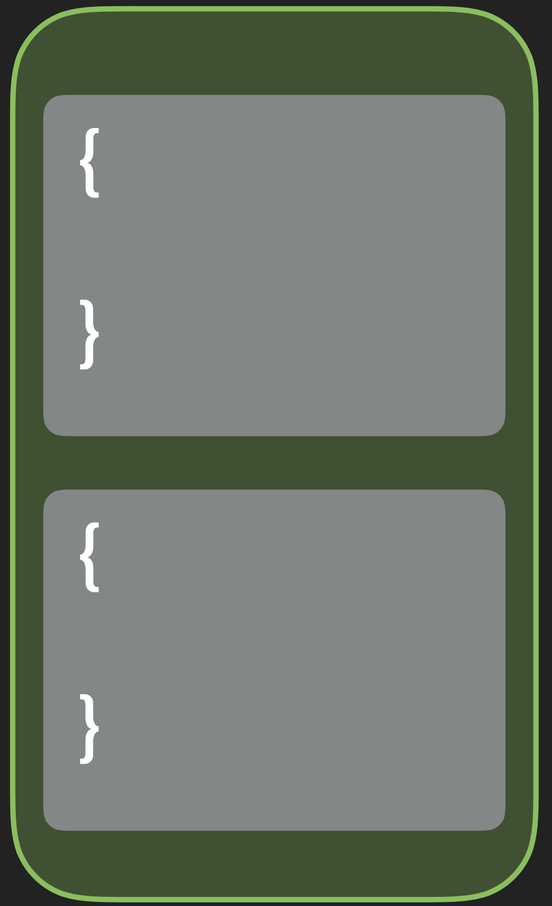


TDD



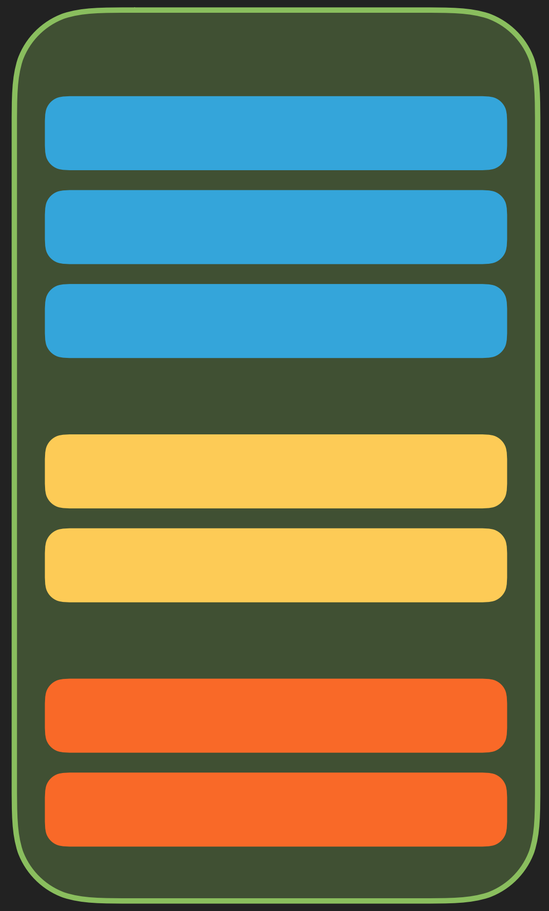






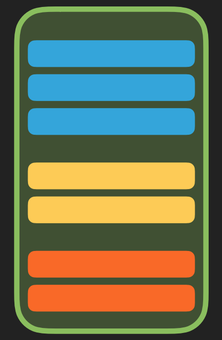
TDD

CODE



CODE

```
import XCTest

struct DetailPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements


    // MARK: - Actions


    // MARK: - Verifications



}
```



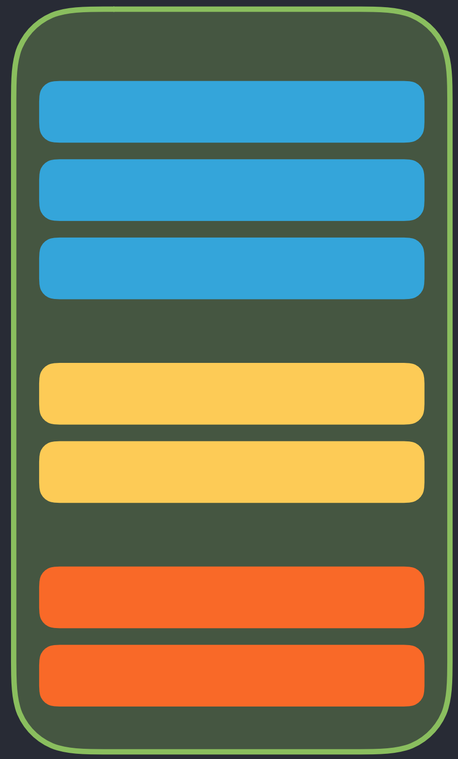
CODE

```swift
import XCTest

struct DetailPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements


    // MARK: - Actions


    // MARK: - Verifications



}
```

```swift
import XCTest

protocol TestPage {
    var testCase: XCTestCase { get }
}


extension TestPage {

    var app: XCUIApplication {
        return XCUIApplication()
    }

}
```

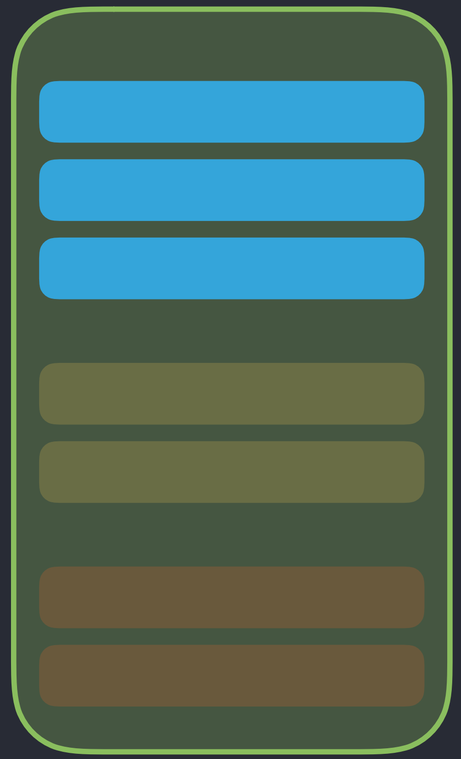
CODE

```
import XCTest

struct DetailPage: TestPage {
    let testCase: XCTestCase

    // MARK: - Elements

    // MARK: - Actions

    // MARK: - Verifications


}
```



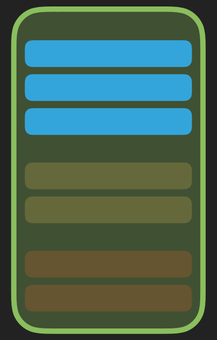
CODE

```
import XCTest

struct DetailPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements


    // MARK: - Actions


    // MARK: - Verifications



}
```



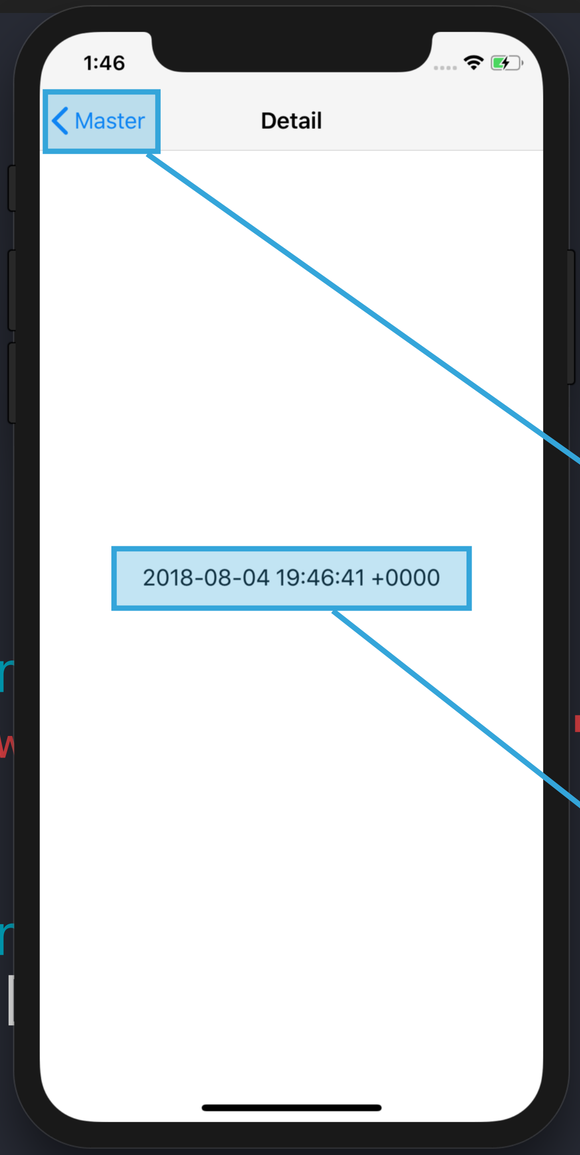
CODE

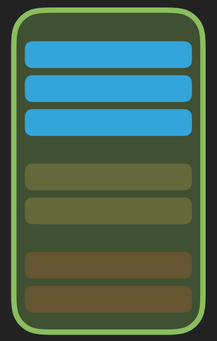
```
import XCTest

struct DetailPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate var detailText: XCUIElement {
        return app.staticTexts[“DetailViewController.label"]
    }

    fileprivate var backButton: XCUIElement {
        return app.navigationBars.buttons["Master"]
    }


    ...
}
```



CODE

```
import XCTest

struct DetailPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate var detailText: XCUIElemer
        return app.staticTexts[“DetailView                "]
    }

    fileprivate var backButton: XCUIElemer
        return app.navigationBars.buttons[
    }

    ...
}
```



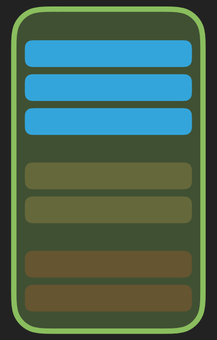


Button

Label

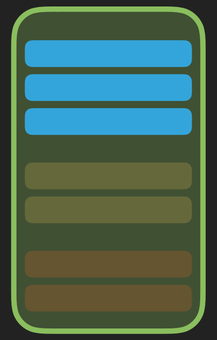
CODE

```
import XCTest

struct DetailPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate var detailText: XCUIElement {
        return app.staticTexts[“DetailViewController.label"]
    }

    fileprivate var backButton: XCUIElement {
        return app.navigationBars.buttons["Master"]
    }


    ...
}
```



CODE

```
import XCTest

struct DetailPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate var detailText: XCUIElement {
        return app.staticTexts[“DetailViewController.label"]
    }

    fileprivate var backButton: XCUIElement {
        return app.navigationBars.buttons["Master"]
    }


    ...
}
```

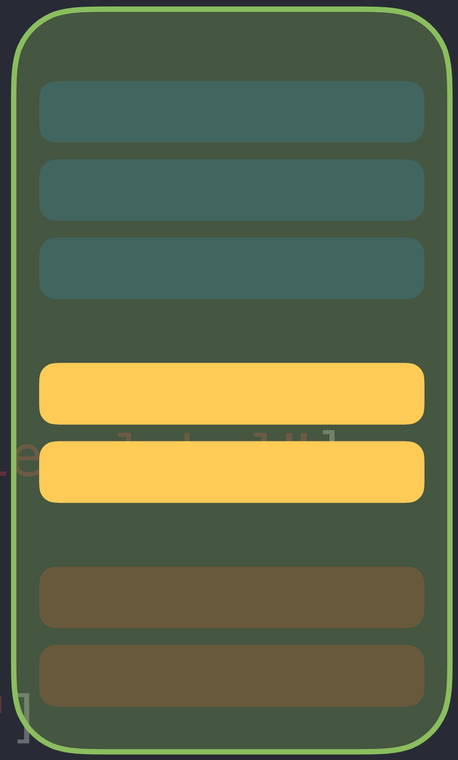


CODE

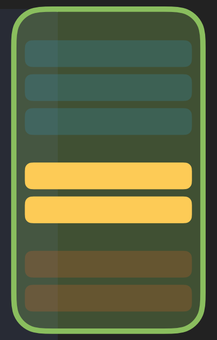
```
import XCTest

struct DetailPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate var detailText: XCUIElement {
        return app.staticTexts[“DetailViewController.label"]
    }

    fileprivate var backButton: XCUIElement {
        return app.navigationBars.buttons["Master"]
    }

    ...
}
```

CODE

```
import XCTest

struct DetailPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate var detailText: XCUIElement {
        return app.staticTexts[“DetailViewControlle
    }

    fileprivate var backButton: XCUIElement {
        return app.navigationBars.buttons["Master"]
    }

    ...
}
```



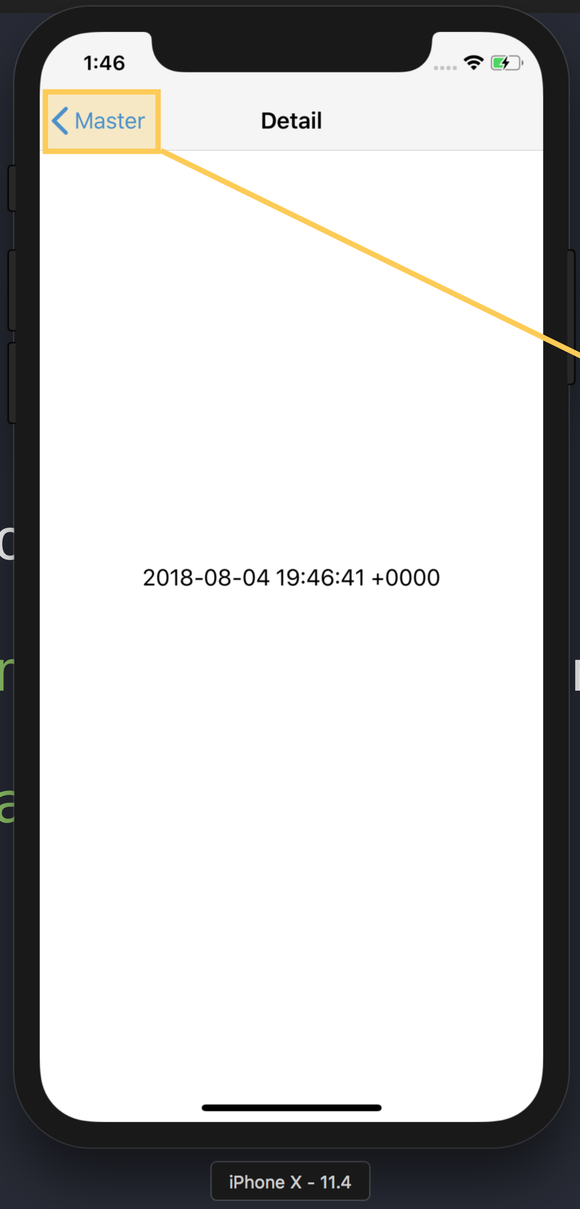
CODE



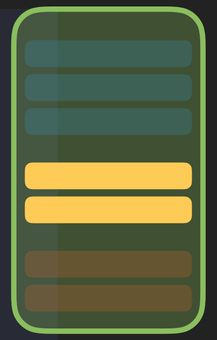
```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }


    ...
}
```

CODE

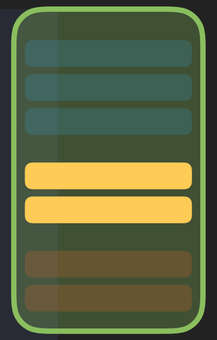
```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButto                    #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButtor                ne: line)
        backButton.tap()
        return MasterPage(testCase: testCa
    }

    ...
}
```



Tap

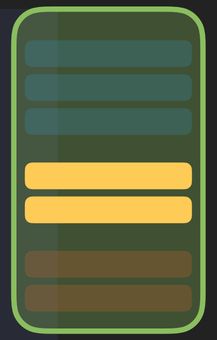


CODE



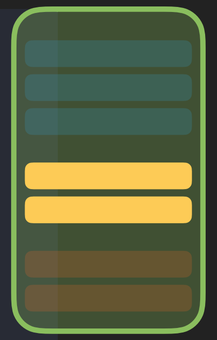
```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }


    ...
}
```

CODE



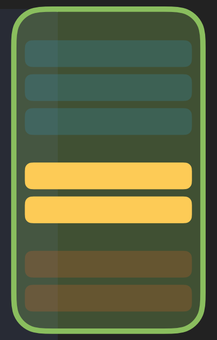
```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }


    ...
}
```

CODE



```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }

    ...
}
```

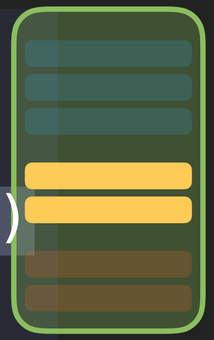
CODE



```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }


    ...
}
```

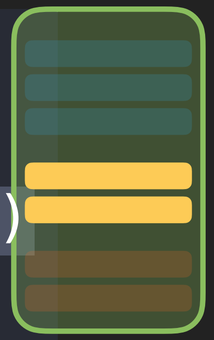
CODE

```
    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }


   ...

    func expect(exists element: XCUIElement, file: String =
#file, line: UInt = #line) {
        if !element.exists {
            recordFailure(withDescription: "Expected \(element)
to exist.", inFile: file, atLine: Int(line), expected: true)
        }
    }

   ...
```



CODE

```
    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }


   ...

    func expect(exists element: XCUIElement, file: String =
#file, line: UInt = #line) {
        if !element.exists {
            recordFailure(withDescription: "Expected \(element)
to exist.", inFile: file, atLine: Int(line), expected: true)
        }
    }

   ...
```

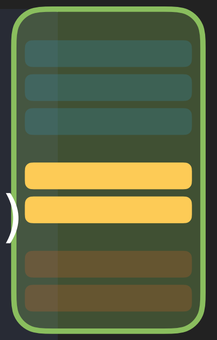


CODE

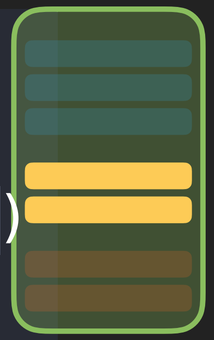
```
    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }



   ...

    func expect(exists element: XCUIElement, file: String =
#file, line: UInt = #line) {
        if !element.exists {
            recordFailure(withDescription: "Expected \(element)
to exist.", inFile: file, atLine: Int(line), expected: true)
        }
    }

   ...
```

CODE

```
    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }



   ...

    func expect(exists element: XCUIElement, file: String =
#file, line: UInt = #line) {
        if !element.exists {
            recordFailure(withDescription: "Expected \(element)
to exist.", inFile: file, atLine: Int(line), expected: true)
        }
    }

   ...
```

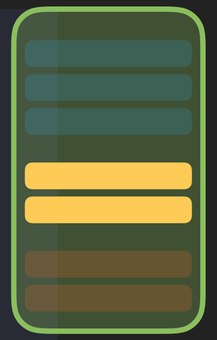


CODE



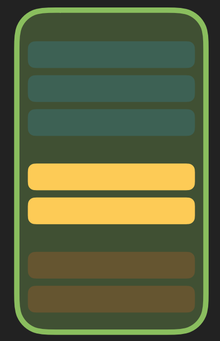
```
    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }



   ...

    func expect(exists element: XCUIElement, file: String =
#file, line: UInt = #line) {
        if !element.exists {
            recordFailure(withDescription: "Expected \(element)
to exist.", inFile: file, atLine: Int(line), expected: true)
        }
    }

   ...
```

CODE



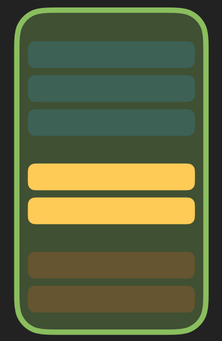
```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }


    ...
}
```

CODE



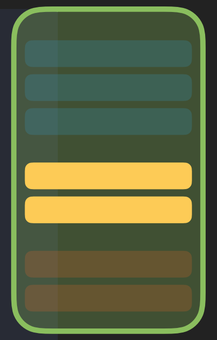
```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: StaticString =
#file, line: UInt = #line) -> MasterPage {
        XCTAssertTrue(backButton.exists, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }

    ...
}
```

CODE



```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: StaticString =
#file, line: UInt = #line) -> MasterPage {
        XCTAssertTrue(backButton.exists, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }

    ...
}
```

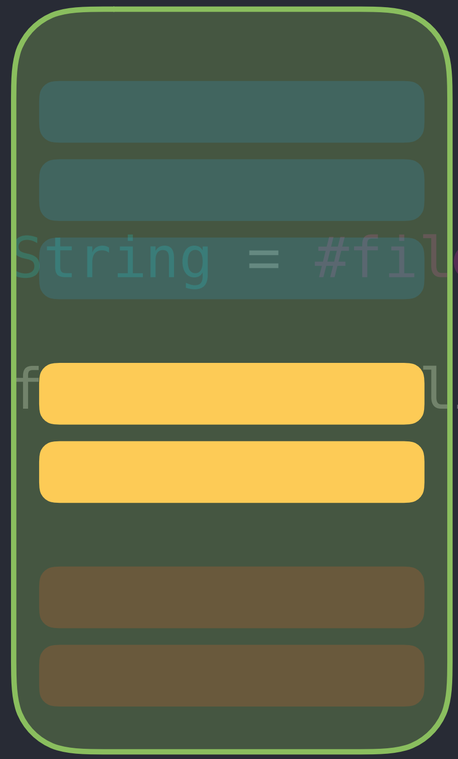
CODE



```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }


    ...
}
```

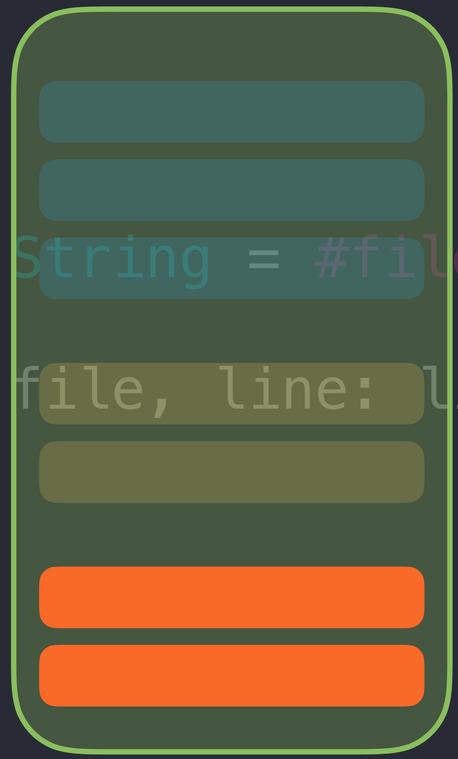
CODE

```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }

    ...
}
```

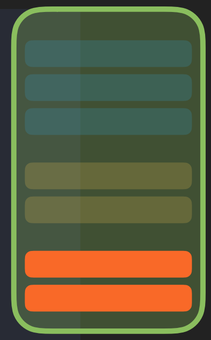


CODE

```
import XCTest

struct DetailPage: TestPage {
    ...

    // MARK: - Actions

    @discardableResult func tapOnBackButton(file: String = #file,
line: UInt = #line) -> MasterPage {
        testCase.expect(exists: backButton, file: file, line: line)
        backButton.tap()
        return MasterPage(testCase: testCase)
    }

    ...
}
```

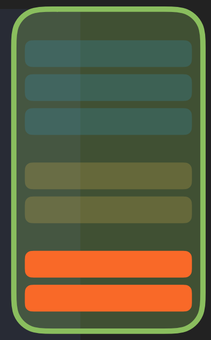


CODE



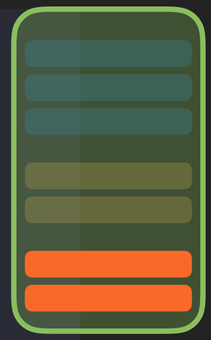
```
    ...

    // MARK: - Verifications

    @discardableResult func verifyDetailPageIsShowing(file: String
= #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        return self
    }

    @discardableResult func verifyLabelText(is text: String, file:
String = #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        testCase.expect(detailText.label, equals: text, file: file,
line: line)
        return self
    }
  ...
}
```

CODE



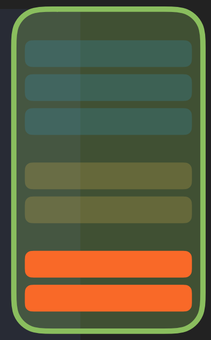
```
    ...

    // MARK: - Verifications

    @discardableResult func verifyDetailPageIsShowing(file: String
= #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        return self
    }

    @discardableResult func verifyLabelText(is text: String, file:
String = #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        testCase.expect(detailText.label, equals: text, file: file,
line: line)
        return self
    }
  ...
}
```

CODE



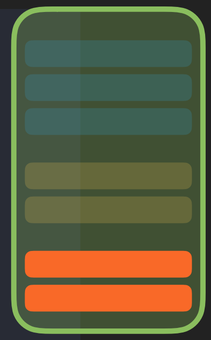
```
    ...

    // MARK: - Verifications

    @discardableResult func verifyDetailPageIsShowing(file: String
= #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        return self
    }

    @discardableResult func verifyLabelText(is text: String, file:
String = #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        testCase.expect(detailText.label, equals: text, file: file,
line: line)
        return self
    }
  ...
}
```

CODE



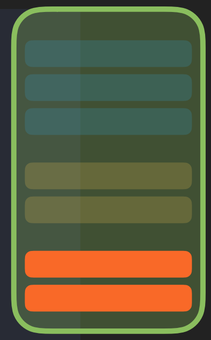
```
    // MARK: - Elements

    fileprivate var detailText: XCUIElement {
        return app.staticTexts[“DetailViewController.label"]
    }




   ...

    // MARK: - Verifications

    @discardableResult func verifyDetailPageIsShowing(file: String
= #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        return self
    }
```

CODE



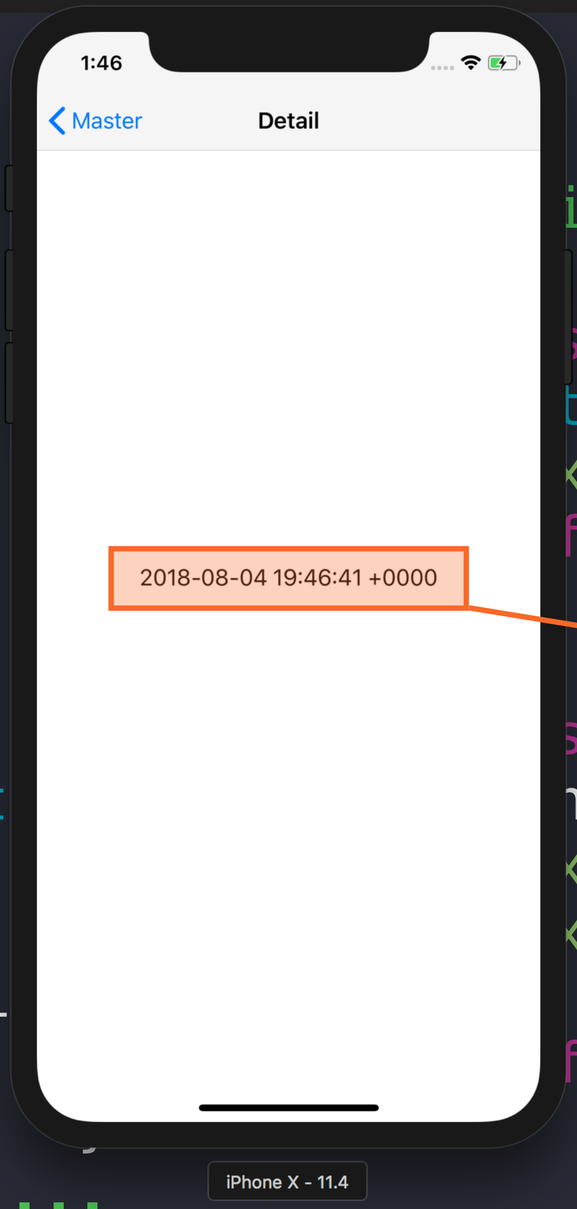
```
    ...

    // MARK: - Verifications

    @discardableResult func verifyDetailPageIsShowing(file: String
= #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        return self
    }

    @discardableResult func verifyLabelText(is text: String, file:
String = #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        testCase.expect(detailText.label, equals: text, file: file,
line: line)
        return self
    }
  ...
}
```

CODE



```
    ...

    // MARK: - Verifications

    @discardableResult func verifyDetailPageIsShowing(file: String
= #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        return self
    }

    @discardableResult func verifyLabelText(is text: String, file:
String = #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        testCase.expect(detailText.label, equals: text, file: file,
line: line)
        return self
    }
  ...
}
```

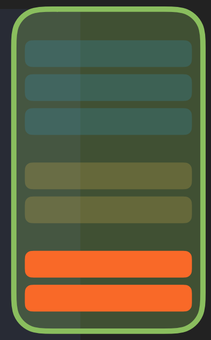
CODE



Text

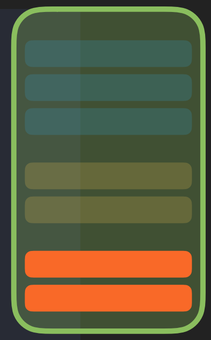
```
ifications

sult func verifyDetailPageIsShowing(file: String
t = #line) -> DetailPage {
xpect(exists: detailText, file: file, line: line)
f

sult func verifyLabelText(is text: String, file:
ne: UInt = #line) -> DetailPage {
xpect(exists: detailText, file: file, line: line)
xpect(detailText.label, equals: text, file: file,
f
```

CODE



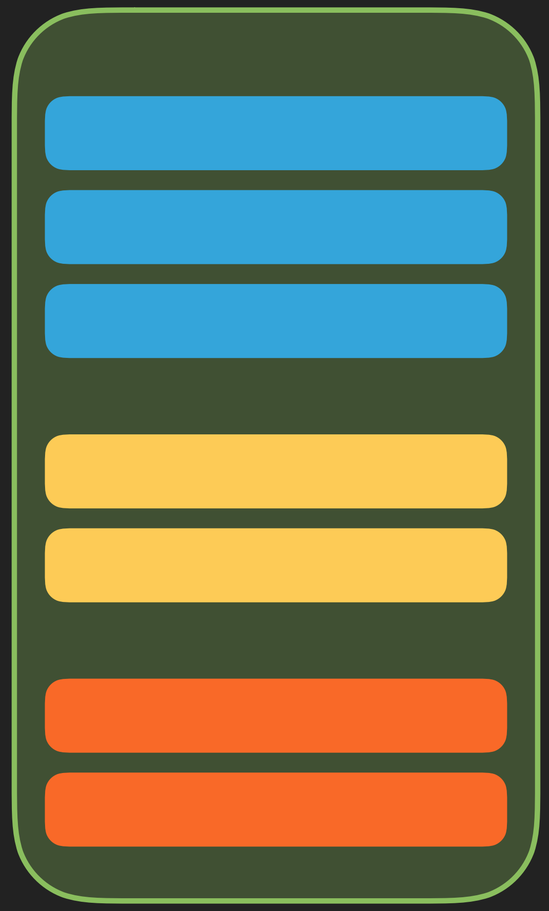
```
    ...

    // MARK: - Verifications

    @discardableResult func verifyDetailPageIsShowing(file: String
= #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        return self
    }

    @discardableResult func verifyLabelText(is text: String, file:
String = #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        testCase.expect(detailText.label, equals: text, file: file,
line: line)
        return self
    }
  ...
}
```

CODE

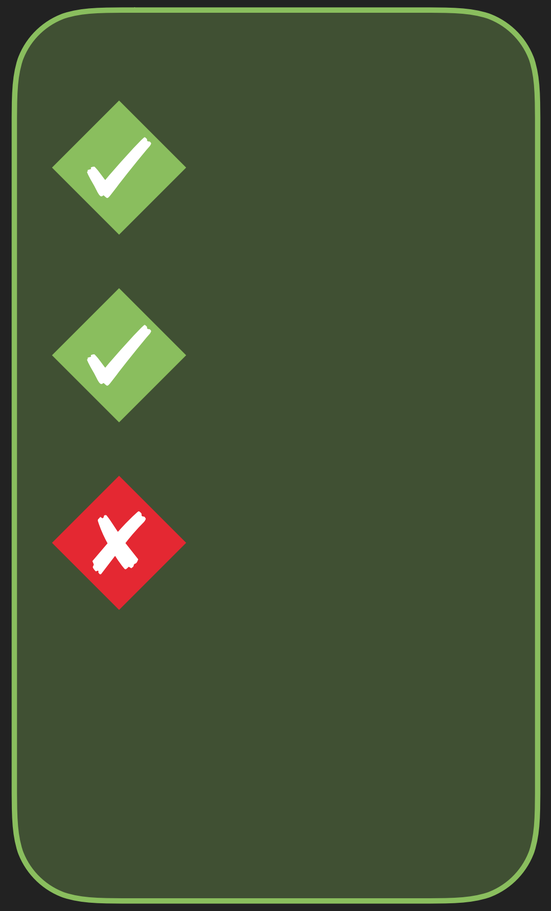


```
    ...

    // MARK: - Verifications

    @discardableResult func verifyDetailPageIsShowing(file: String
= #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        return self
    }

    @discardableResult func verifyLabelText(is text: String, file:
String = #file, line: UInt = #line) -> DetailPage {
        testCase.expect(exists: detailText, file: file, line: line)
        testCase.expect(detailText.label, equals: text, file: file,
line: line)
        return self
    }
  ...
}
```

CODE



CODE

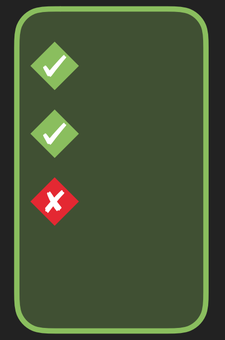


CODE

```
import XCTest

class BasicEntryUITests: XCTestCase {



}
```

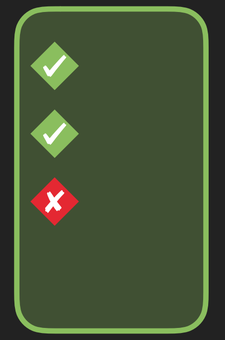
CODE

```
import XCTest

class BasicEntryUITests: XCTestCase {

    override func setUp() {
        super.setUp()

        continueAfterFailure = false
        let application = XCUIApplication()
        application.launchArguments.append("--uitesting")
        application.launch()
    }

}
```



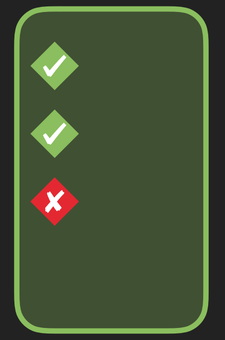
CODE

```
import XCTest

class BasicEntryUITests: XCTestCase {

    override func setUp() {
        super.setUp()

        continueAfterFailure = false
        let application = XCUIApplication()
        application.launchArguments.append("--uitesting")
        application.launch()
    }

}
```



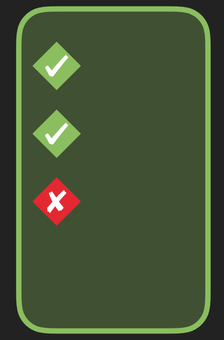
CODE

```
import XCTest

class BasicEntryUITests: XCTestCase {

    override func setUp() {
        super.setUp()

        continueAfterFailure = false
        let application = XCUIApplication()
        application.launchArguments.append("--uitesting")
        application.launch()
    }

}
```



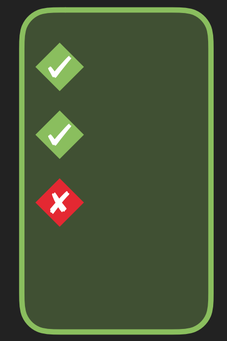
CODE

```
        continueAfterFailure = false
        let application = XCUIApplication()
        application.launchArguments.append("--uitesting")
        application.launch()
    }

}


class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        if CommandLine.arguments.contains("--uitesting") {
            // Set up state for UI testing
        }
        ...
    }
}
```



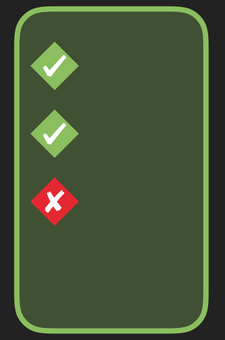
CODE

```
        continueAfterFailure = false
        let application = XCUIApplication()
        application.launchArguments.append("--uitesting")
        application.launch()
    }

}


class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        if CommandLine.arguments.contains("--uitesting") {
            // Set up state for UI testing
        }
        ...
    }
}
```



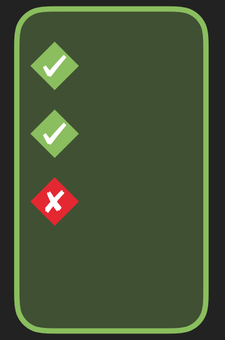
CODE

```
import XCTest

class BasicEntryUITests: XCTestCase {

    override func setUp() {
        super.setUp()

        continueAfterFailure = false
        let application = XCUIApplication()
        application.launchArguments.append("--uitesting")
        application.launch()
    }

}
```



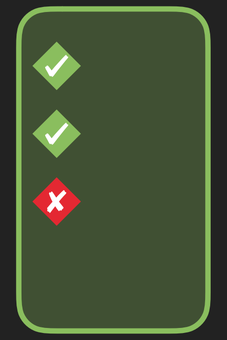
CODE

```
...
func testAddingEntry() {




}
```



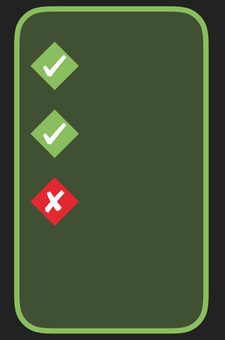
CODE

```
...
func testAddingEntry() {
    MasterPage(testCase: self)




}
```



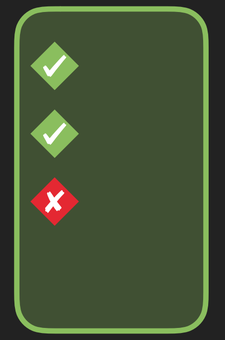
CODE

```
...
func testAddingEntry() {
    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()









}
```



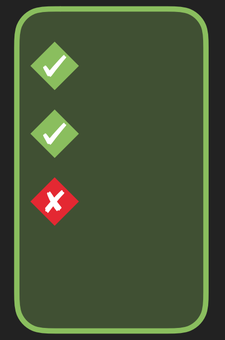
CODE

```
...
func testAddingEntry() {
    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)




}
```



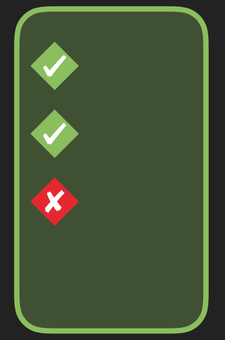
CODE

```
...
func testAddingEntry() {
    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()




}
```



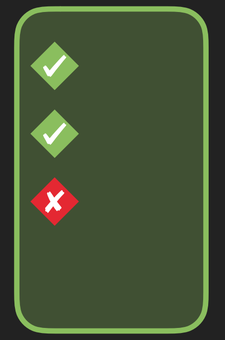
CODE

```
...
func testAddingEntry() {
    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)




}
```



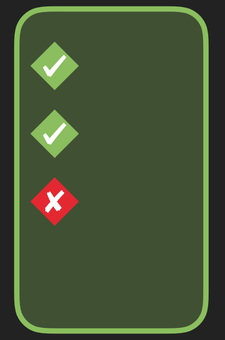
CODE

```
...
func testAddingEntry() {
    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: )




}
```



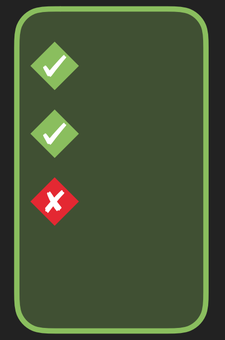
CODE

```
...
func testAddingEntry() {
    let nowLabel = String(describing: Date())

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: nowLabel)









}
```



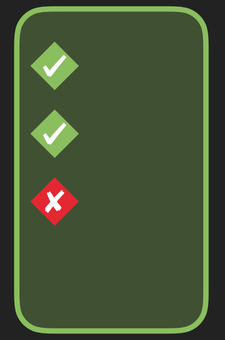
CODE

```
...
func testAddingEntry() {
    let nowLabel = String(describing: Date())

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: nowLabel)
        .tapOnCell(at: 0)




}
```



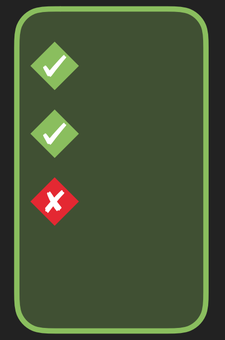
CODE

```
...
func testAddingEntry() {
    let nowLabel = String(describing: Date())

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: nowLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()



}
```



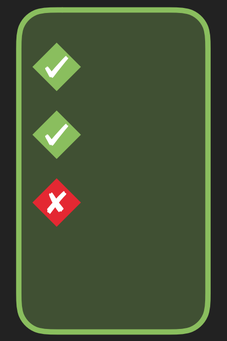
CODE

```
...
func testAddingEntry() {
    let nowLabel = String(describing: Date())

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: nowLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()
        .verifyLabelText(is: nowLabel)



}
```



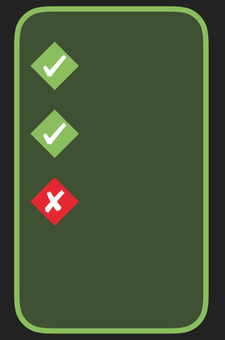
CODE

```
...
func testAddingEntry() {
    let nowLabel = String(describing: Date())

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: nowLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()
        .verifyLabelText(is: nowLabel)
        .tapOnBackButton()



}
```



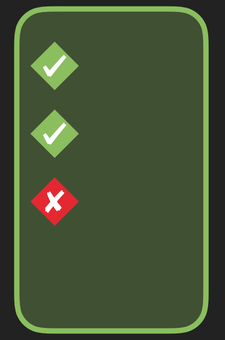
CODE

```
...
func testAddingEntry() {
    let nowLabel = String(describing: Date())

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: nowLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()
        .verifyLabelText(is: nowLabel)
        .tapOnBackButton()

        // Master page
        .verifyMasterPageIsShowing()
}
```



CODE

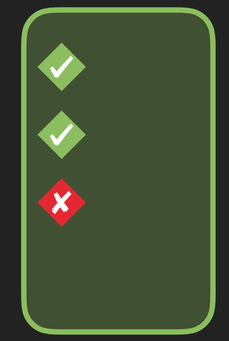
```
...
func testAddingEntry() {
    let nowLabel = String(describing: Date())

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: nowLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()
        .verifyLabelText(is: nowLabel)
        .tapOnBackButton()

        // Master page
        .verifyMasterPageIsShowing()
}
```



CODE

```
...
func testAddingEntry() {
    let nowLabel = String(describing: Date())

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: nowLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()
        .verifyLabelText(is: nowLabel)
        .tapOnBackButton()

        // Master page
        .verifyMasterPageIsShowing()
}
```

✘ Expected ‘Optional(…



CODE

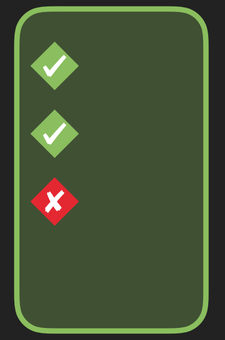
```
...
func testAddingEntry() {
    let zeroLabel = "0"

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: zeroLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()
        .verifyLabelText(is: zeroLabel)
        .tapOnBackButton()

        // Master page
        .verifyMasterPageIsShowing()
}
```

CODE

```
...
func testAddingEntry() {
    let zeroLabel = "0"

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: zeroLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()
        .verifyLabelText(is: zeroLabel)
        .tapOnBackButton()

        // Master page
        .verifyMasterPageIsShowing()
}
```

CODE

```
...
func testAddingAndDeletingEntry() {
    let zeroLabel = "0"

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: zeroLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()
        .verifyLabelText(is: zeroLabel)
        .tapOnBackButton()

        // Master page
        .verifyMasterPageIsShowing()
}
```

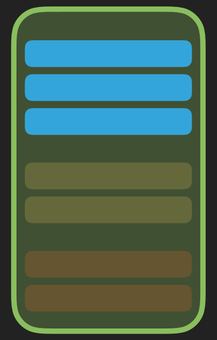


CODE

```
...
func testAddingAndDeletingEntry() {
    let zeroLabel = "0"

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: zeroLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()
        .verifyLabelText(is: zeroLabel)
        .tapOnBackButton()

        // Master page
        .verifyMasterPageIsShowing()
}
```

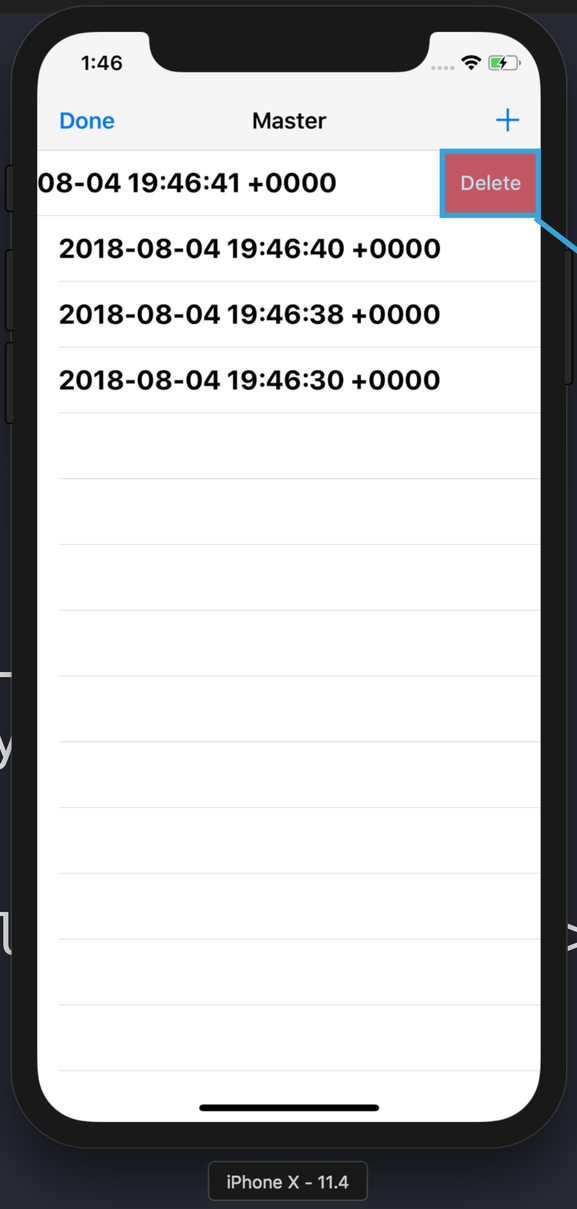
CODE

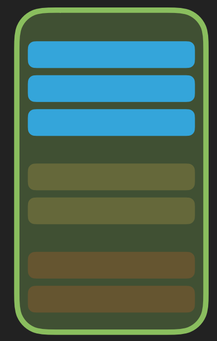
```
import XCTest

struct MasterPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate func cell(at index: Int) -> XCUIElement {
        return table.cells.element(boundBy: index)
    }

    fileprivate func deleteButton(for cell: XCUIElement) ->
    XCUIElement {
        return cell.buttons["Delete"]
    }
    ...
}
```



CODE

```
import XCTest

struct MasterPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate func cell(at index: Int) -
        return table.cells.element(boundBy
    }

    fileprivate func deleteButton(for cell                    >
    XCUIElement {
        return cell.buttons["Delete"]
    }
    ...
}
```

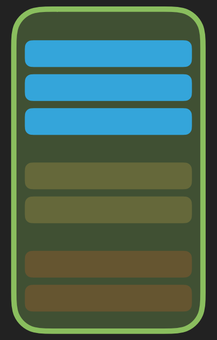




Button

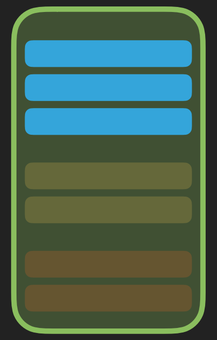
CODE

```
import XCTest

struct MasterPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate func cell(at index: Int) -> XCUIElement {
        return table.cells.element(boundBy: index)
    }

    fileprivate func deleteButton(for cell: XCUIElement) ->
    XCUIElement {
        return cell.buttons["Delete"]
    }
    ...
}
```



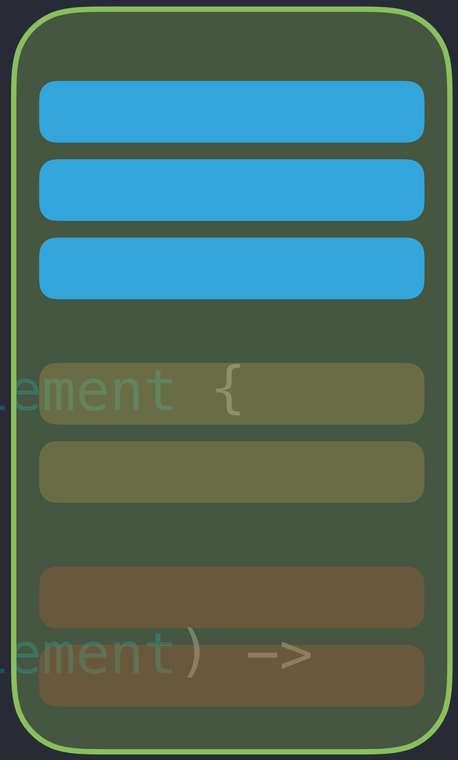
CODE

```
import XCTest

struct MasterPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate func cell(at index: Int) -> XCUIElement {
        return table.cells.element(boundBy: index)
    }

    fileprivate func deleteButton(for cell: XCUIElement) ->
    XCUIElement {
        return cell.buttons["Delete"]
    }
    ...
}
```



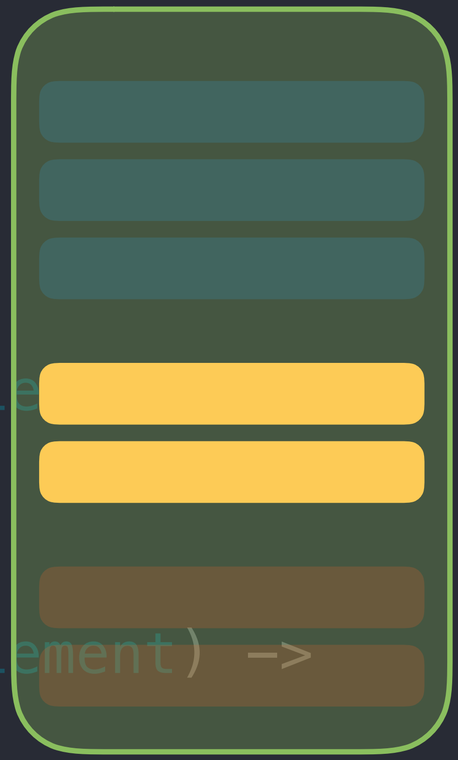
CODE

```
import XCTest

struct MasterPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate func cell(at index: Int) -> XCUIElement {
        return table.cells.element(boundBy: index)
    }

    fileprivate func deleteButton(for cell: XCUIElement) ->
    XCUIElement {
        return cell.buttons["Delete"]
    }
    ...
}
```

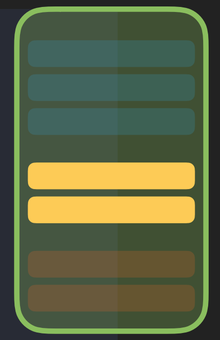


CODE

```
import XCTest

struct MasterPage: TestPage {

    let testCase: XCTestCase


    // MARK: - Elements

    fileprivate func cell(at index: Int) -> XCUIEle
        return table.cells.element(boundBy: index)
    }

    fileprivate func deleteButton(for cell: XCUIElement) ->
    XCUIElement {
        return cell.buttons["Delete"]
    }
    ...
}
```



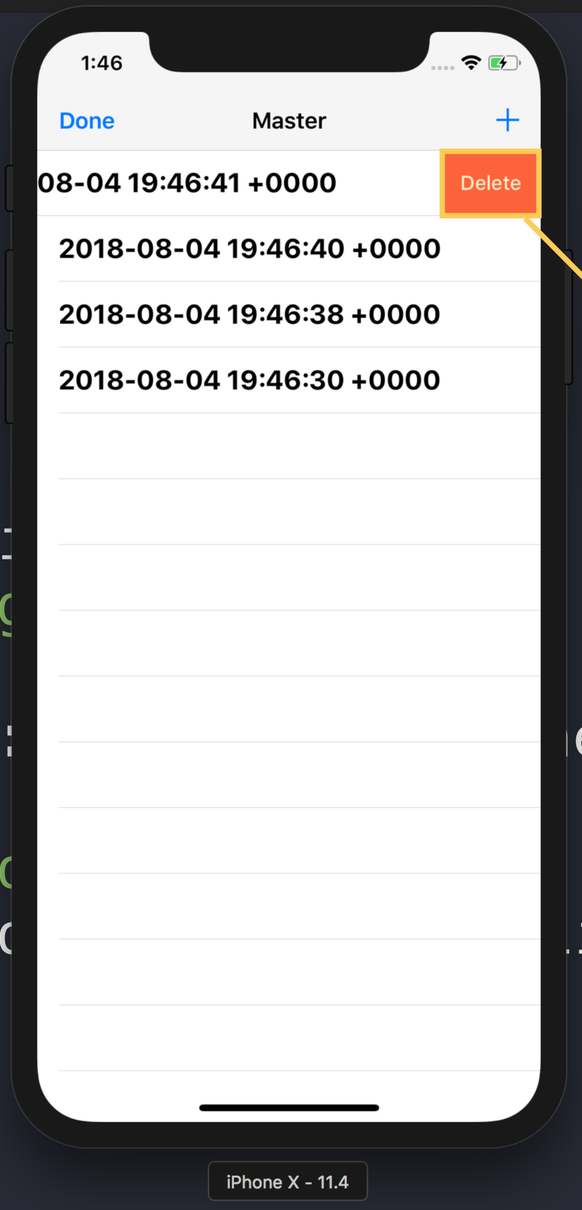
CODE

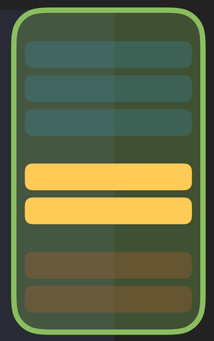


```
import XCTest

struct MasterPage: TestPage {
    ...

     // MARK: - Actions

    @discardableResult func deleteCell(at index: Int, file: String =
    #file, line: UInt = #line) -> MasterPage {
        let cell = self.cell(at: index)
        testCase.expect(exists: cell, file: file, line: line)
        cell.swipeLeft()
        let deleteButton = self.deleteButton(for: cell)
        testCase.expect(exists: deleteButton, file: file, line: line)
        deleteButton.tap()
        return self
    }
    ...
}
```

CODE

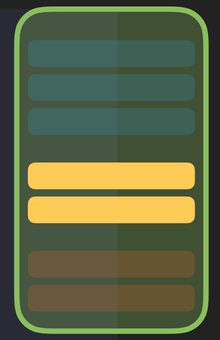
```
import XCTest

struct MasterPage: TestPage {
    ...

     // MARK: - Actions

    @discardableResult func deleteCell(at :                String =
    #file, line: UInt = #line) -> MasterPag
        let cell = self.cell(at: index)
        testCase.expect(exists: cell, file:                e)
        cell.swipeLeft()
        let deleteButton = self.deleteButto
        testCase.expect(exists: deleteButto                ine: line)
        deleteButton.tap()
        return self
    }
    ...
}
```





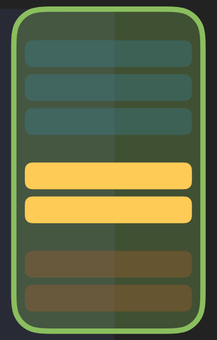
Tap

CODE



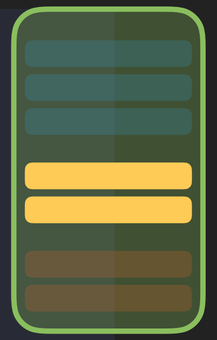
```
import XCTest

struct MasterPage: TestPage {
    ...

     // MARK: - Actions

    @discardableResult func deleteCell(at index: Int, file: String =
    #file, line: UInt = #line) -> MasterPage {
        let cell = self.cell(at: index)
        testCase.expect(exists: cell, file: file, line: line)
        cell.swipeLeft()
        let deleteButton = self.deleteButton(for: cell)
        testCase.expect(exists: deleteButton, file: file, line: line)
        deleteButton.tap()
        return self
    }
    ...
}
```

CODE



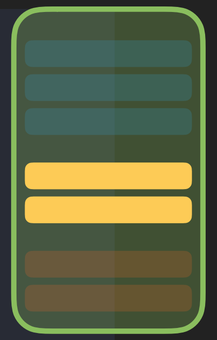
```
import XCTest

struct MasterPage: TestPage {
    ...

     // MARK: - Actions

    @discardableResult func deleteCell(at index: Int, file: String =
    #file, line: UInt = #line) -> MasterPage {
        let cell = self.cell(at: index)
        testCase.expect(exists: cell, file: file, line: line)
        cell.swipeLeft()
        let deleteButton = self.deleteButton(for: cell)
        testCase.expect(exists: deleteButton, file: file, line: line)
        deleteButton.tap()
        return self
    }
    ...
}
```

CODE



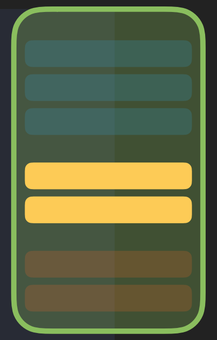
```
import XCTest

struct MasterPage: TestPage {
    ...

     // MARK: - Actions

    @discardableResult func deleteCell(at index: Int, file: String =
    #file, line: UInt = #line) -> MasterPage {
        let cell = self.cell(at: index)
        testCase.expect(exists: cell, file: file, line: line)
        cell.swipeLeft()
        let deleteButton = self.deleteButton(for: cell)
        testCase.expect(exists: deleteButton, file: file, line: line)
        deleteButton.tap()
        return self
    }
    ...
}
```

CODE



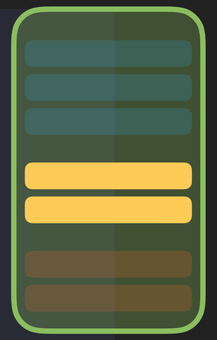
```
import XCTest

struct MasterPage: TestPage {
    ...

     // MARK: - Actions

    @discardableResult func deleteCell(at index: Int, file: String =
    #file, line: UInt = #line) -> MasterPage {
        let cell = self.cell(at: index)
        testCase.expect(exists: cell, file: file, line: line)
        cell.swipeLeft()
        let deleteButton = self.deleteButton(for: cell)
        testCase.expect(exists: deleteButton, file: file, line: line)
        deleteButton.tap()
        return self
    }
    ...
}
```

CODE



```
import XCTest

struct MasterPage: TestPage {
    ...

     // MARK: - Actions

    @discardableResult func deleteCell(at index: Int, file: String =
    #file, line: UInt = #line) -> MasterPage {
        let cell = self.cell(at: index)
        testCase.expect(exists: cell, file: file, line: line)
        cell.swipeLeft()
        let deleteButton = self.deleteButton(for: cell)
        testCase.expect(exists: deleteButton, file: file, line: line)
        deleteButton.tap()
        return self
    }
    ...
}
```

CODE



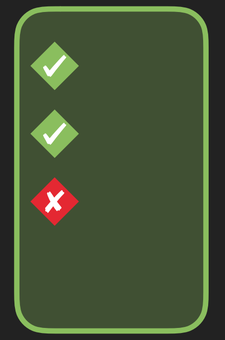
```
import XCTest

struct MasterPage: TestPage {
    ...

     // MARK: - Actions

    @discardableResult func deleteCell(at index: Int, file: String =
    #file, line: UInt = #line) -> MasterPage {
        let cell = self.cell(at: index)
        testCase.expect(exists: cell, file: file, line: line)
        cell.swipeLeft()
        let deleteButton = self.deleteButton(for: cell)
        testCase.expect(exists: deleteButton, file: file, line: line)
        deleteButton.tap()
        return self
    }
    ...
}
```

CODE

```
import XCTest

struct MasterPage: TestPage {
    ...

     // MARK: - Actions

    @discardableResult func deleteCell(at index: Int, file: String =
    #file, line: UInt = #line) -> MasterPage {
        let cell = self.cell(at: index)
        testCase.expect(exists: cell, file: file, li
        cell.swipeLeft()
        let deleteButton = self.deleteButton(for: cell)
        testCase.expect(exists: deleteButton, file: file, line: line)
        deleteButton.tap()
        return self
    }
    ...
}
```

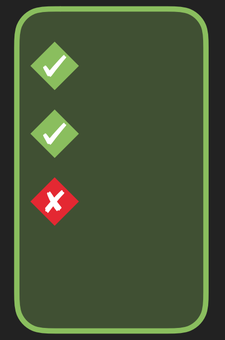
CODE

```
...
func testAddingAndDeletingEntry() {
    let zeroLabel = "0"

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: zeroLabel)
        .tapOnCell(at: 0)

        // Detail page
        .verifyDetailPageIsShowing()
        .verifyLabelText(is: zeroLabel)
        .tapOnBackButton()

        // Master page
        .verifyMasterPageIsShowing()
}
```



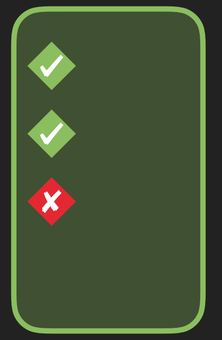
CODE

```
...
func testAddingAndDeletingEntry() {
    let zeroLabel = "0"

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: zeroLabel)
        .tapOnCell(at: 0)

        ...

        // Master page
        .verifyMasterPageIsShowing()



}
```



CODE

```
...
func testAddingAndDeletingEntry() {
    let zeroLabel = "0"

    MasterPage(testCase: self)
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 0)
        .tapOnAddButton()
        .verifyTableCellCount(is: 1)
        .verifyCell(at: 0, hasLabel: zeroLabel)
        .tapOnCell(at: 0)

        ...

        // Master page
        .verifyMasterPageIsShowing()
        .verifyTableCellCount(is: 1)
        .deleteCell(at: 0)
        .verifyTableCellCount(is: 0)
}
```



CODE

TIPS & TRICKS

WWDC

2018

Testing Tips & Tricks

2017

Engineering for Testability

TIPS & TRICKS

LOCAL DATA

TIPS & TRICKS

LOCAL DATA

AVOID DUPLICATION

TIPS & TRICKS

LOCAL DATA

AVOID DUPLICATION

USE MULTIPLE SCHEMES

TIPS & TRICKS

GETTING STARTED

NEXT BUG/ FEATURE

GETTING STARTED

NEXT BUG/
FEATURE

IMPORTANT
FREQUENT

GETTING STARTED

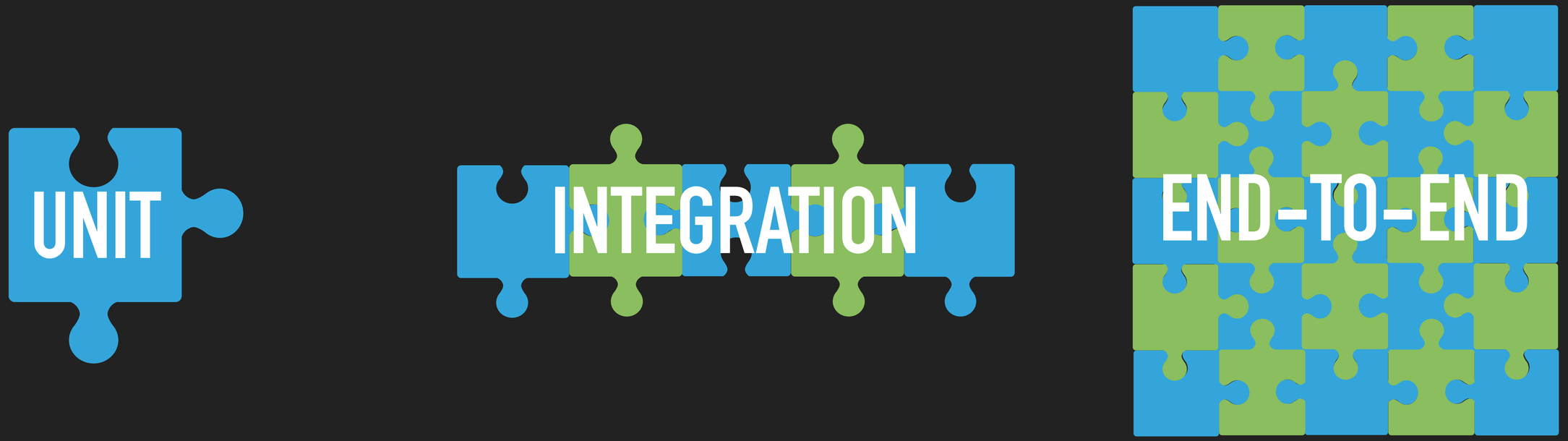
NEXT BUG/
FEATURE

IMPORTANT
FREQUENT

IMPORTANT
INFREQUENT

GETTING STARTED

SUMMARY



AUTOMATED

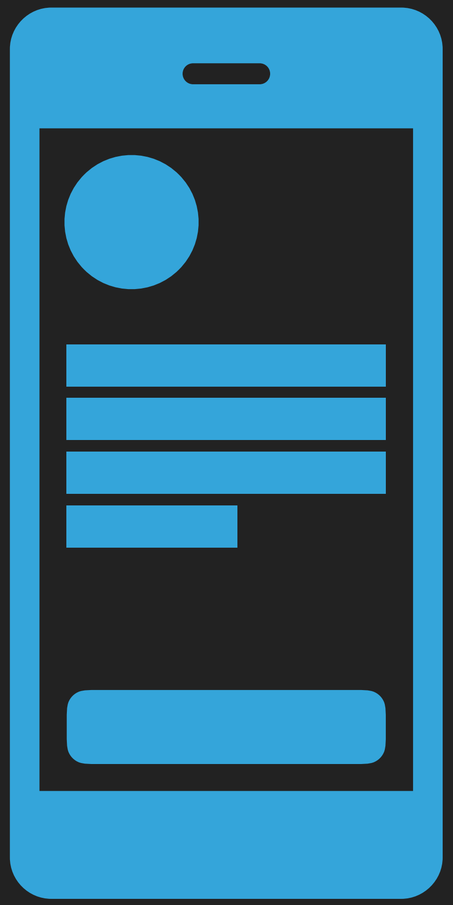
SOFTWARE TESTING

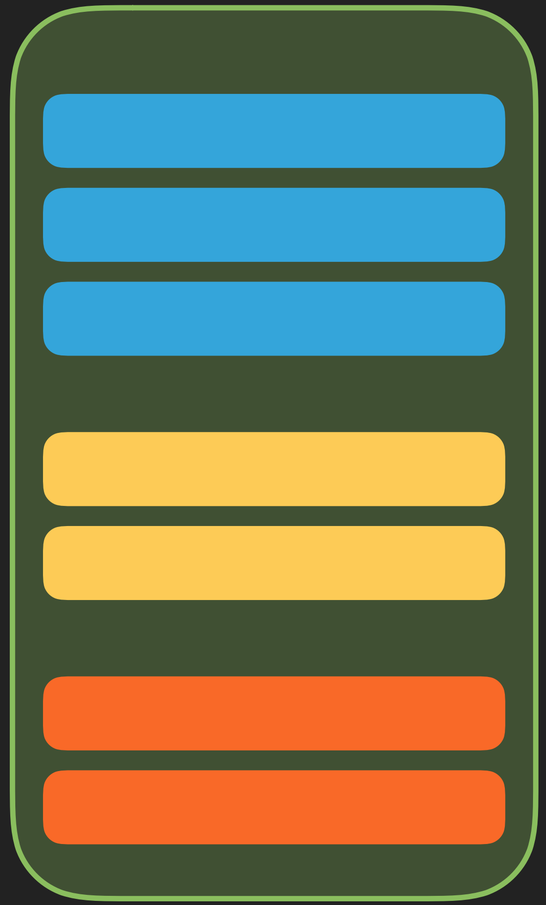
VERIFY BEHAVIOR

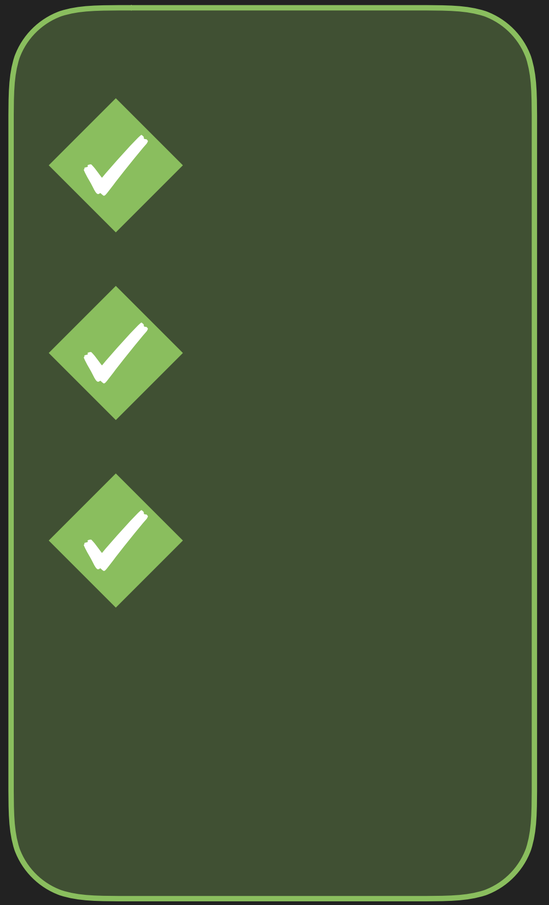
BUILT ON ACCESSIBILITY

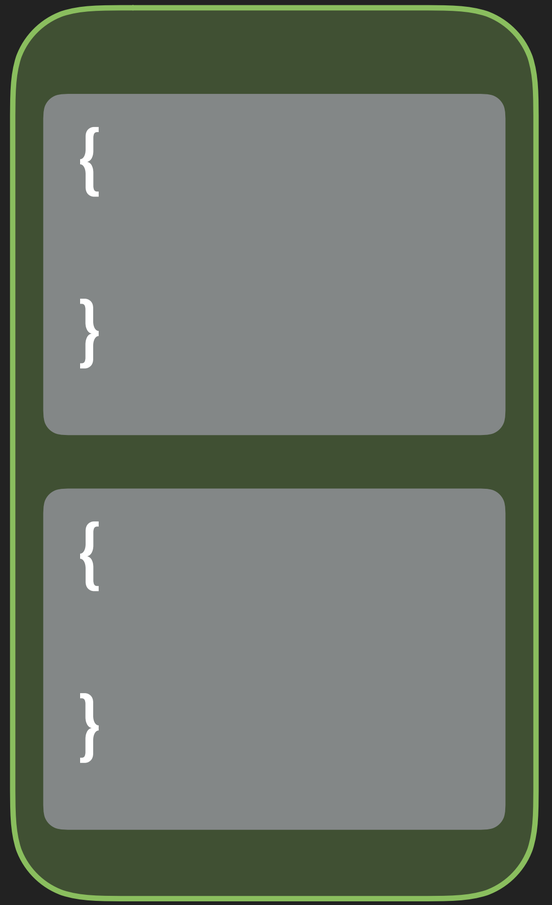
RECORD TO LEARN

UI TESTS









TDD

//BSN.DESIGN

COCOAHEADS • AUG 2018

UI-TDD