**Software Module Test**

**Module Test: Railway Booking**

**B0529001 馮少迪**

目錄

一、V&V 動作摘要列表 1

二、測試案例 4

三、確認報告 6

四、測試報告 9

測試運行結果 21

軟體代碼： 30

# 

# 一、V&V 動作摘要列表

|  |  |  |  |
| --- | --- | --- | --- |
| 測試項目 | 單元階層 | 整合階層 | 系統階層 |
| 錯誤(Fault)、警示、危險(Hazard)測試 | ● |  |  |
| 錯誤(Error)、範圍確認、邊界值測試 | ● |  | ● |
| 時間選擇分析和測試 |  |  | ● |
| 特定的演算法測試與分析 | ● |  |  |
| 壓力測試 |  |  | ● |
| 裝置選擇、附件、結構測試 | ● | ● | ● |
| 溝通(Communication)測試 | ● | ● | ● |
| 記憶初始化測試 | ● | ● | ● |
| 接受度和試用版測試 |  |  |  |
| 回歸測試(Regression testing) | ● | ● | ● |

通過與失敗標準

|  |  |
| --- | --- |
| 測試項目 | 通過或失敗的原因 |
| 錯誤(Error)、範圍確認、邊界值測試 | ν 操作錯誤，觀察是否有警示訊息出現  ν 確認每個單元所限制的範圍(I/O)，嘗試以邊界值或超出限制範圍的值來測試單元，測驗其是否會出現錯誤 |
| 時間選擇分析和測試 | ν 在不允許的環因下做測試，觀察是否有警  示訊息的出現 |
| 特定的演算法測試與分析 | ν 比較特定演算法與臨床實驗的結果是否  符合 |
| 壓力測試 | ν 給予訂票系統軟體不正常的條件，觀察軟  體是否會出現錯誤  ν 不正常條件例如：給予不正常的值 |
| 裝置選擇、附件、結構測試 | ν 對照需求規格書，測試硬體的操作上是否  有問題 |
| 溝通(Communication)測試 | ν 測試單元與單元之間值的傳遞有無問題 |
| 回歸測試(Regression | ν 當把有錯誤的地方修改過後，再重複測試 |

# 

# 二、測試案例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Item** | **Description** | **Pass/Fail** | **Test Report** |
| 1 | 個人登入 | 輸入使用者帳號及密碼 | P | 0A-VV-001 |
| 3 | 個人訂票資料篩選 | 選擇車站，乘車日期，乘車時段，火車種類等然後按下確認button。 | P | 0A-VV-002 |
| 4 | 火車班次選擇 | 在班次選擇頁面選擇想要的班次 | P | 0A-VV-004 |
| 5 | 座位種類選擇 | 在座位選擇頁面選擇想要的座位種類 | P | 0A-VV-005 |
| 6 | 座位選擇 | 在座位選擇頁面選擇想要的座位 | P | 0A-VV-006 |
| 7 | 付款 | 在頁面顯示訂單資訊，並按下付款按鈕，資料庫中新增此訂單 | P | 0A-VV-007 |
| 8 | 個人訂票記錄查詢 | 在主頁選擇record按鈕,顯示個人訂單。 | P | 0A-VV-008 |
| 9 | 個人訂單取消 | 在訂單詳情頁面點擊取消訂票按鈕，訂單消失，資料庫中刪除此訂單。 | P | 0A-VV-009 |

# 

# 三、確認報告

3.1 針對 SAS 所列的需求

|  |  |  |
| --- | --- | --- |
| 需求編號 | 描述 | 完成度 |
| LG-001 | 使用者輸入賬號密碼身份證號碼進行註冊或登錄 | 100% |
| LG-002 | 註冊或登錄成功，進入系統主頁面。 | 100% |
| ST-001 | 選取欲搭乘之日期,跳出日期選擇器。。 | 100% |
| ST-002 | 選擇車種。 | 100% |
| ST-003 | 選擇起訖站。 | 100% |
| ST-004 | 點擊確認按鈕 | 100% |
| ST-005 | 選擇篩選後的車次 | 100% |
| ST-006 | 選擇座位種類 | 100% |
| ST-007 | 選擇座位，並單擊繼續按鈕 | 100% |
| ST-008 | 跳出訂單資訊，單擊支付訂單按鈕，成功後回到主頁面 | 100% |
| SS-001 | 單擊訂單按鈕，跳出個人訂單 | 100% |
| SS-002 | 單擊一個訂單，跳出訂單詳情頁面。 | 100% |
| SS-003 | 單擊取消訂單按鈕。 | 100% |

3.2 針對風險評估所列的風險

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 危害 | 原因敘述 | 風險等級 |
| 1 | 網路連線中斷 | 電信設備商故障 | 1 |

3.3 針對 SDS 所列的系統設計規格

|  |  |  |
| --- | --- | --- |
| 模組名稱 | 模組元件編碼 | 完成度 |
| 登入模組 | LG | 100% |
| firebase模組 | FB | 100% |
| 車次篩選模組 | TS | 100% |
| 座位種類選擇模組 | KS | 100% |
| 座位選擇模組 | SS | 100% |
| 訂單查詢模組 | ST | 100% |
| 訂票取消模組 | CT | 100% |

# 

# 四、測試報告

0A-VV-001

|  |  |
| --- | --- |
| 測試編號 | 0A-VV-001 |
| Firmware version | SC1-V8.00 |
| 測試日期 | 2019/12/27 |
| 測試人員 | 馮少迪 |
| 稽核人員 | 馮少迪 |
| 測試項目 | 登錄註冊系統 |
| 測試重點 | 能否正常輸入賬號密碼和身份證字號 |
| 測試輸入 | 賬號密碼身份證字號 |
| 輸出項目 | 成功後資料庫新增賬號或原有賬號登錄。 |
| 測試步驟 | 輸入一組已存在賬號或新的賬號 |
| 環境需求 | 1. MACOS 系統 2. XCODE SIMULATOR |
| 測試通過標準 | 1. 系統各UI模塊單擊無異常 2. FIREBASE 資料儲存與輸出正確無誤 |
| 測試結果 |  |

0A-VV-002

|  |  |
| --- | --- |
| 測試編號 | 0A-VV-002 |
| Firmware version | SC1-V8.00 |
| 測試日期 | 2019/12/27 |
| 測試人員 | 馮少迪 |
| 稽核人員 | 馮少迪 |
| 測試項目 | 個人訂票資料篩選 |
| 測試重點 | 能否正確選擇車站，車種，選擇出發日期，時間段 |
| 測試輸入 | 隨機選擇的車站，車種，出發日期，時間段 |
| 輸出項目 | 正常的選擇後結果頁面。火車篩選後的資料結果 |
| 測試步驟 | 根據原本UI設計選擇車站車站，用時間選擇器選擇日期。最後按下繼續按鈕 |
| 環境需求 | MACOS 系統  XCODE SIMULATOR |
| 測試通過標準 | 系統各UI模塊單擊無異常  班次頁面顯示的資料符合篩選 |
| 測試結果 |  |

0A-VV-003

|  |  |
| --- | --- |
| 測試編號 | 0A-VV-003 |
| Firmware version | SC1-V8.00 |
| 測試日期 | 2019/12/27 |
| 測試人員 | 馮少迪 |
| 稽核人員 | 馮少迪 |
| 測試項目 | 火車班次選擇 |
| 測試重點 | 能否正確顯示各班次的資料，點擊每份資料是否可以跳轉，tableview是否可以正常下滑。 |
| 測試輸入 | 班次篩選頁面傳過來的資料。隨機選擇的班次。 |
| 輸出項目 | 成功顯示篩選後的班次，送出班次資料。 |
| 測試步驟 | 隨機選擇一個班次 |
| 環境需求 | MACOS 系統  XCODE SIMULATOR |
| 測試通過標準 | 系統各UI模塊單擊無異常  班次選擇無異常，上一頁面傳過來的資料正常。 |
| 測試結果 |  |

0A-VV-004

|  |  |
| --- | --- |
| 測試編號 | 0A-VV-004 |
| Firmware version | SC1-V8.00 |
| 測試日期 | 2019/12/27 |
| 測試人員 | 馮少迪 |
| 稽核人員 | 馮少迪 |
| 測試項目 | 座位種類選擇 |
| 測試重點 | 正常顯示班次資料，篩選資料。正常顯示座位UI，座位選擇菜單。 |
| 測試輸入 | 隨機選擇一種座位。 |
| 輸出項目 | 送出座位種類資料。隨機選擇一種座位種類。 |
| 測試步驟 | 隨機選擇一個班次 |
| 環境需求 | MACOS 系統  XCODE SIMULATOR |
| 測試通過標準 | 系統各UI模塊單擊無異常  正常顯示班次資料，篩選資料。 |
| 測試結果 |  |

0A-VV-005

|  |  |
| --- | --- |
| 測試編號 | 0A-VV-005 |
| Firmware version | SC1-V8.00 |
| 測試日期 | 2019/12/27 |
| 測試人員 | 馮少迪 |
| 稽核人員 | 馮少迪 |
| 測試項目 | 座位選擇 |
| 測試重點 | 正常顯示班次資料，篩選資料。正常顯示座位3D UI，確認按鈕 |
| 測試輸入 | 隨機選擇一個座位。 |
| 輸出項目 | 隨機選擇的座位。 |
| 測試步驟 | 隨機選擇一個座位。然後按下確認按鈕。 |
| 環境需求 | MACOS 系統  XCODE SIMULATOR |
| 測試通過標準 | 系統各UI模塊單擊無異常  正常顯示座位3DUI, 選擇成功後座位可用信息改變。 |
| 測試結果 |  |

0A-VV-006

|  |  |
| --- | --- |
| 測試編號 | 0A-VV-006 |
| Firmware version | SC1-V8.00 |
| 測試日期 | 2019/12/27 |
| 測試人員 | 馮少迪 |
| 稽核人員 | 馮少迪 |
| 測試項目 | 付款確認 |
| 測試重點 | 確認訂票資訊符合之前篩選的結果，資料庫能否新增次訂票資料。 |
| 測試輸入 |  |
| 輸出項目 | 新增訂票資料 |
| 測試步驟 | 按下付款按鈕。 |
| 環境需求 | MACOS 系統  XCODE SIMULATOR |
| 測試通過標準 | 系統各UI模塊單擊無異常  正常顯示訂票資訊，資料庫新增訂票。 |
| 測試結果 |  |

0A-VV-007

|  |  |
| --- | --- |
| 測試編號 | 0A-VV-007 |
| Firmware version | SC1-V8.00 |
| 測試日期 | 2019/12/27 |
| 測試人員 | 馮少迪 |
| 稽核人員 | 馮少迪 |
| 測試項目 | 個人訂票記錄查詢 |
| 測試重點 | 顯示使用者的訂票資料 |
| 測試輸入 | 隨機選擇一張訂票。 |
| 輸出項目 | 無 |
| 測試步驟 | 隨機選擇一張訂票。 |
| 環境需求 | MACOS 系統  XCODE SIMULATOR |
| 測試通過標準 | 系統各UI模塊單擊無異常  正常顯示訂票UI |
| 測試結果 |  |

0A-VV-008

|  |  |
| --- | --- |
| 測試編號 | 0A-VV-008 |
| Firmware version | SC1-V8.00 |
| 測試日期 | 2019/12/27 |
| 測試人員 | 馮少迪 |
| 稽核人員 | 馮少迪 |
| 測試項目 | 個人訂單取消 |
| 測試重點 | 正常顯示訂票詳細資料。取消button |
| 測試輸入 | 按下取消訂票按鈕。 |
| 輸出項目 | 刪除選擇的訂票資料。 |
| 測試步驟 | 按下取消訂票按鈕。 |
| 環境需求 | MACOS 系統  XCODE SIMULATOR |
| 測試通過標準 | 系統各UI模塊單擊無異常  正常顯示訂票UI，刪除訂票資料。 |
| 測試結果 |  |

# 

# **測試運行結果**

**Test Case '-**[**Shinkansen3DSeatBookingPrototypeUITests.Shinkansen3DSeatBookingPrototypeUITests testLaunchPerformance**]**' passed** (**37.101 seconds**)**.**

**Test Case '-**[**Shinkansen3DSeatBookingPrototypeUITests.Shinkansen3DSeatBookingPrototypeUITests testUI**]**' started.**

**t = 0.00s Start Test at 2019-12-28 22:35:42.761**

**t = 0.03s Set Up**

**t = 0.04s Tap "IDNUMBER" TextField**

**t = 0.04s Wait for com.fsd.railwaybooking to idle**

**t = 0.10s Find the "IDNUMBER" TextField**

**t = 0.25s Check for interrupting elements affecting "IDNUMBER" TextField**

**t = 0.33s Synthesize event**

**t = 0.47s Wait for com.fsd.railwaybooking to idle**

**t = 0.77s Tap "EMAIL" TextField**

**t = 0.77s Wait for com.fsd.railwaybooking to idle**

**t = 0.95s Find the "EMAIL" TextField**

**t = 1.02s Check for interrupting elements affecting "EMAIL" TextField**

**t = 1.08s Synthesize event**

**t = 1.17s Wait for com.fsd.railwaybooking to idle**

**t = 1.32s Tap "PASSWORD" SecureTextField**

**t = 1.32s Wait for com.fsd.railwaybooking to idle**

**t = 1.35s Find the "PASSWORD" SecureTextField**

**t = 1.39s Check for interrupting elements affecting "PASSWORD" SecureTextField**

**t = 1.43s Synthesize event**

**t = 1.53s Wait for com.fsd.railwaybooking to idle**

**t = 1.60s Tap "LOGIN" Button**

**t = 1.60s Wait for com.fsd.railwaybooking to idle**

**t = 1.64s Find the "LOGIN" Button**

**t = 1.68s Check for interrupting elements affecting "LOGIN" Button**

**t = 1.72s Synthesize event**

**t = 1.82s Wait for com.fsd.railwaybooking to idle**

**t = 2.41s Tap "BOOKING" Button**

**t = 2.41s Wait for com.fsd.railwaybooking to idle**

**t = 2.46s Find the "BOOKING" Button**

**t = 2.49s Check for interrupting elements affecting "BOOKING" Button**

**t = 2.52s Synthesize event**

**t = 2.61s Wait for com.fsd.railwaybooking to idle**

**t = 3.52s Tap Other**

**t = 3.52s Wait for com.fsd.railwaybooking to idle**

**t = 3.60s Find the Other**

**t = 3.68s Check for interrupting elements affecting Other**

**t = 3.75s Synthesize event**

**t = 3.90s Wait for com.fsd.railwaybooking to idle**

**t = 4.45s Tap "** 臺北市**" StaticText**

**t = 4.45s Wait for com.fsd.railwaybooking to idle**

**t = 4.54s Find the "** 臺北市**" StaticText**

**t = 4.71s Check for interrupting elements affecting "** 臺北市**" StaticText**

**t = 4.80s Synthesize event**

**t = 4.92s Wait for com.fsd.railwaybooking to idle**

**t = 5.44s Tap Other**

**t = 5.45s Wait for com.fsd.railwaybooking to idle**

**t = 5.53s Find the Other**

**t = 5.62s Check for interrupting elements affecting Other**

**t = 5.70s Synthesize event**

**t = 5.84s Wait for com.fsd.railwaybooking to idle**

**t = 6.36s Tap "** 臺中市**" StaticText**

**t = 6.36s Wait for com.fsd.railwaybooking to idle**

**t = 6.43s Find the "** 臺中市**" StaticText**

**t = 6.56s Check for interrupting elements affecting "** 臺中市**" StaticText**

**t = 6.62s Synthesize event**

**t = 6.74s Wait for com.fsd.railwaybooking to idle**

**t = 7.24s Tap Other**

**t = 7.24s Wait for com.fsd.railwaybooking to idle**

**t = 7.32s Find the Other**

**t = 7.39s Check for interrupting elements affecting Other**

**t = 7.45s Synthesize event**

**t = 7.57s Wait for com.fsd.railwaybooking to idle**

**t = 7.94s Tap Other**

**t = 7.94s Wait for com.fsd.railwaybooking to idle**

**t = 8.00s Find the Other**

**t = 8.05s Check for interrupting elements affecting Other**

**t = 8.10s Synthesize event**

**t = 8.24s Wait for com.fsd.railwaybooking to idle**

**t = 8.42s Tap "Afternoon" StaticText**

**t = 8.42s Wait for com.fsd.railwaybooking to idle**

**t = 8.50s Find the "Afternoon" StaticText**

**t = 8.55s Check for interrupting elements affecting "Afternoon" StaticText**

**t = 8.61s Synthesize event**

**t = 8.72s Wait for com.fsd.railwaybooking to idle**

**t = 8.92s Tap "Evening" StaticText**

**t = 8.92s Wait for com.fsd.railwaybooking to idle**

**t = 8.99s Find the "Evening" StaticText**

**t = 9.04s Check for interrupting elements affecting "Evening" StaticText**

**t = 9.09s Synthesize event**

**t = 9.22s Wait for com.fsd.railwaybooking to idle**

**t = 9.41s Tap Other**

**t = 9.41s Wait for com.fsd.railwaybooking to idle**

**t = 9.47s Find the Other**

**t = 9.54s Check for interrupting elements affecting Other**

**t = 9.60s Synthesize event**

**t = 9.73s Wait for com.fsd.railwaybooking to idle**

**t = 9.94s Tap "**自強號**" StaticText**

**t = 9.94s Wait for com.fsd.railwaybooking to idle**

**t = 10.00s Find the "**自強號**" StaticText**

**t = 10.06s Check for interrupting elements affecting "**自強號**" StaticText**

**t = 10.10s Synthesize event**

**t = 10.22s Wait for com.fsd.railwaybooking to idle**

**t = 10.42s Tap "**莒光號**" StaticText**

**t = 10.43s Wait for com.fsd.railwaybooking to idle**

**t = 10.47s Find the "**莒光號**" StaticText**

**t = 10.53s Check for interrupting elements affecting "**莒光號**" StaticText**

**t = 10.58s Synthesize event**

**t = 10.70s Wait for com.fsd.railwaybooking to idle**

**t = 10.90s Tap Other**

**t = 10.90s Wait for com.fsd.railwaybooking to idle**

**t = 10.95s Find the Other**

**t = 11.01s Check for interrupting elements affecting Other**

**t = 11.06s Synthesize event**

**t = 11.18s Wait for com.fsd.railwaybooking to idle**

**t = 11.37s Tap "**自強號**" StaticText**

**t = 11.37s Wait for com.fsd.railwaybooking to idle**

**t = 11.42s Find the "**自強號**" StaticText**

**t = 11.47s Check for interrupting elements affecting "**自強號**" StaticText**

**t = 11.52s Synthesize event**

**t = 11.65s Wait for com.fsd.railwaybooking to idle**

**t = 11.84s Tap "Search for Tickets" Button**

**t = 11.84s Wait for com.fsd.railwaybooking to idle**

**t = 11.89s Find the "Search for Tickets" Button**

**t = 11.94s Check for interrupting elements affecting "Search for Tickets" Button**

**t = 11.99s Synthesize event**

**t = 12.12s Wait for com.fsd.railwaybooking to idle**

**t = 12.21s Tap "9:24 PM – 9:48 PM" StaticText**

**t = 12.22s Wait for com.fsd.railwaybooking to idle**

**t = 12.80s Find the "9:24 PM – 9:48 PM" StaticText**

**t = 13.84s Find the "9:24 PM – 9:48 PM" StaticText** (**retry 1**)

**t = 14.00s Check for interrupting elements affecting "9:24 PM – 9:48 PM" StaticText**

**t = 14.05s Synthesize event**

**t = 14.18s Wait for com.fsd.railwaybooking to idle**

**t = 14.39s Tap "**殘障座**" StaticText**

**t = 14.39s Wait for com.fsd.railwaybooking to idle**

**t = 14.99s Find the "**殘障座**" StaticText**

**t = 15.08s Check for interrupting elements affecting "**殘障座**" StaticText**

**t = 15.13s Synthesize event**

**t = 15.28s Wait for com.fsd.railwaybooking to idle**

**t = 16.53s Tap Other**

**t = 16.54s Wait for com.fsd.railwaybooking to idle**

**t = 17.14s Find the Other**

**t = 17.23s Check for interrupting elements affecting Other**

**t = 17.32s Synthesize event**

**t = 17.44s Wait for com.fsd.railwaybooking to idle**

**t = 17.49s Tap "Please Select a Seat" Button**

**t = 17.49s Wait for com.fsd.railwaybooking to idle**

**t = 17.52s Find the "Please Select a Seat" Button**

**t = 17.59s Check for interrupting elements affecting "Please Select a Seat" Button**

**t = 17.63s Synthesize event**

**t = 17.73s Wait for com.fsd.railwaybooking to idle**

**t = 17.77s Tear Down**

**Test Case '-**[**Shinkansen3DSeatBookingPrototypeUITests.Shinkansen3DSeatBookingPrototypeUITests testUI**]**' passed** (**17.771 seconds**)**.**

**Test Suite 'Shinkansen3DSeatBookingPrototypeUITests' passed at 2019-12-28 22:36:00.531.**

**Executed 4 tests**, **with 0 failures** (**0 unexpected**) **in 61.090** (**61.098**) **seconds**

**Test Suite 'Shinkansen3DSeatBookingPrototypeUITests.xctest' passed at 2019-12-28 22:36:00.533.**

**Executed 4 tests**, **with 0 failures** (**0 unexpected**) **in 61.090** (**61.103**) **seconds**

**Test Suite 'All tests' passed at 2019-12-28 22:36:00.546.**

**Executed 4 tests**, **with 0 failures** (**0 unexpected**) **in 61.090** (**61.118**) **seconds**

測試代碼：  
**func** testUI(){

**let** app = XCUIApplication()

app.textFields["IDNUMBER"].tap()

app.textFields["EMAIL"].tap()

app.secureTextFields["PASSWORD"].tap()

app.buttons["LOGIN"].tap()

app.buttons["BOOKING"].tap()

**let** element5 = app.children(matching: .window).element(boundBy: 0).children(matching: .other).element.children(matching: .other).element.children(matching: .other).element.children(matching: .other).element.children(matching: .other).element.children(matching: .other).element(boundBy: 1)

**let** element3 = element5.children(matching: .other).element

**let** element = element3.children(matching: .other).element(boundBy: 0)

element.children(matching: .other).element(boundBy: 0).children(matching: .other).element.tap()

**let** tablesQuery2 = app.tables

**let** tablesQuery = tablesQuery2

tablesQuery/\*@START\_MENU\_TOKEN@\*/.staticTexts[" 臺北市"]/\*[[".cells.staticTexts[\" 臺北市\"]",".staticTexts[\" 臺北市\"]"],[[[-1,1],[-1,0]]],[0]]@END\_MENU\_TOKEN@\*/.tap()

element.children(matching: .other).element(boundBy: 2).children(matching: .other).element.tap()

tablesQuery/\*@START\_MENU\_TOKEN@\*/.staticTexts[" 臺中市"]/\*[[".cells.staticTexts[\" 臺中市\"]",".staticTexts[\" 臺中市\"]"],[[[-1,1],[-1,0]]],[0]]@END\_MENU\_TOKEN@\*/.tap()

**let** element2 = element3.children(matching: .other).element(boundBy: 1).children(matching: .other).element

element2.tap()

//element2.tap()

//app.collectionViews/\*@START\_MENU\_TOKEN@\*/.staticTexts["28"]/\*[[".cells.staticTexts[\"28\"]",".staticTexts[\"28\"]"],[[[-1,1],[-1,0]]],[0]]@END\_MENU\_TOKEN@\*/.tap()

//app.buttons["SELECT"].tap()

element3.children(matching: .other).element(boundBy: 2).children(matching: .other).element.tap()

app.staticTexts["Afternoon"].tap()

app.staticTexts["Evening"].tap()

**let** element4 = element3.children(matching: .other).element(boundBy: 3).children(matching: .other).element

element4.tap()

**let** staticText = app.staticTexts["自強號"]

staticText.tap()

app.staticTexts["莒光號"].tap()

element4.tap()

//element4.tap()

staticText.tap()

app.buttons["Search for Tickets"].tap()

tablesQuery2.children(matching: .cell).element(boundBy: 0).staticTexts["9:24 PM – 9:48 PM"].tap()

tablesQuery/\*@START\_MENU\_TOKEN@\*/.staticTexts["殘障座"]/\*[[".cells.staticTexts[\"殘障座\"]",".staticTexts[\"殘障座\"]"],[[[-1,1],[-1,0]]],[0]]@END\_MENU\_TOKEN@\*/.tap()

element5.children(matching: .other).element(boundBy: 1).children(matching: .other).element.children(matching: .other).element.children(matching: .other).element.tap()

app.buttons["Please Select a Seat"].tap()

**let** purchaseThisTicketButton = app.buttons["Purchase this Ticket"]

purchaseThisTicketButton.tap()

element5.tap()

app.buttons["RECORD"].tap()

}

# **軟體代碼：**

//

// signupview.swift

//

import UIKit

import Firebase

import FirebaseDatabase

//struct HeadInfo:Codable {

// var dayOfWeek:String = ""

// var date:String = ""

// var fromStation:String = ""

// var toStation:String = ""

//}

struct Ticket:Convertable{

var Train:TrainSchedule?

var seat:SeatClassEntity?

var head:HeaderInformation?

}

protocol Convertable: Codable {

}

extension Convertable {

/// 直接将Struct或Class转成Dictionary

func convertToDict() -> Dictionary<String, Any>? {

var dict: Dictionary<String, Any>? = nil

do {

print("init student")

let encoder = JSONEncoder()

let data = try encoder.encode(self)

print("struct convert to data")

dict = try JSONSerialization.jsonObject(with: data, options: .allowFragments) as? Dictionary<String, Any>

} catch {

print(error)

}

return dict

}

}

var idcard:String = ""

var email:String = ""

var password:String = ""

var Myticket:[Ticket] = []

struct TotalTicket:Convertable{

var ticketlist:[Ticket] = []

}

var TotalTicketlist = TotalTicket()

class SignUpViewController: UIViewController {

@IBOutlet weak var PASSWORD: UITextField!

@IBOutlet weak var EMAIL: UITextField!

@IBOutlet weak var ID: UITextField!

@IBOutlet weak var login: LineButton!

var ref: DatabaseReference! = Database.database().reference()

@IBAction func Userlogin(\_ sender: Any) {

idcard = ID.text ?? "none"

email = EMAIL.text ?? "none"

password = PASSWORD.text ?? "none"

Auth.auth().createUser(withEmail: email, password: password) { authResult, error in

// ...

}

Auth.auth().signIn(withEmail: email, password: password) { [weak self] authResult, error in

guard let strongSelf = self else { return }

// ...

}

print("end")

let userID = Auth.auth().currentUser?.uid

print("userID: \(userID)")

ref.child("users").observeSingleEvent(of: .value, with: { (snapshot) in

if snapshot.hasChild(userID!){

self.ref.child("users").child(userID!).observeSingleEvent(of: .value, with: { (snapshot) in

if snapshot.hasChild("tickets"){

self.ref.child("users").child(userID!).child("tickets").observeSingleEvent(of: .value, with: { (snapshot) in

// Get user value

//let value = snapshot.value as? NSDictionary

//Myticket = value?["TICKET"] as? [Ticket] ?? []

let dic = snapshot.value as? NSDictionary

let data = try! JSONSerialization.data(withJSONObject: dic, options: [])

TotalTicketlist = try! JSONDecoder().decode(TotalTicket.self, from: data)

Myticket = TotalTicketlist.ticketlist

print("Myticket\(Myticket)")

//let username = value?["username"] as? String ?? ""r

//let user = User(username: username)

print("value")

print(dic)

// ...

}) { (error) in

print("error")

print(error.localizedDescription)

}

}

})

}

else{

self.ref.child("users").child(userID!).setValue(["IDCARD": idcard])

//self.ref.child("users").child(userID!).setValue(["TICKET": Myticket])

print("INITIAL")

}

})

let vc = UIStoryboard(name: "Storyboard", bundle: nil)

.instantiateViewController(withIdentifier: "selectview") as! SelectView

let presentedViewController = NavigationController(rootViewController: vc)

presentedViewController.modalPresentationStyle = .fullScreen

present(presentedViewController, animated: true, completion: nil)

//

//

// present(viewController, animated: true, completion: nil)

// let vc = SelectView()

//self.present(vc, animated: true, completion: nil)

//self.view.present(vc, animated: true, completion: nil)

print("fsd")

}

override func viewDidLoad() {

super.viewDidLoad()

// Do any additional setup after loading the view.

}

/\*

// MARK: - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation

override func prepare(for segue: UIStoryboardSegue, sender: Any?) {

// Get the new view controller using segue.destination.

// Pass the selected object to the new view controller.

}

\*/

}//

// SelectView.swift

// Shinkansen3DSeatBookingPrototype

//

// Created by shawn on 2019/12/24.

// Copyright © 2019 Virakri Jinangkul. All rights reserved.

//

import UIKit

class SelectView: UIViewController {

@IBAction func Booking(\_ sender: Any) {

let vc = BookingCriteriaViewController()

self.navigationController?.pushViewController(vc, animated: true)

}

@IBAction func Record(\_ sender: Any) {

let vc = MyticketViewcontroller()

self.navigationController?.pushViewController(vc,animated:true)

}

override func viewDidLoad() {

super.viewDidLoad()

// Do any additional setup after loading the view.

}

/\*

// MARK: - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation

override func prepare(for segue: UIStoryboardSegue, sender: Any?) {

// Get the new view controller using segue.destination.

// Pass the selected object to the new view controller.

}

\*/

}

//

// BookingNavigationController.swift

import UIKit

class NavigationController: UINavigationController {

override func motionBegan(\_ motion: UIEvent.EventSubtype, with event: UIEvent?) {

displayAlertToDismiss()

}

override func viewDidLoad() {

super.viewDidLoad()

currentColorTheme = .dark

NotificationCenter.default.addObserver(self,

selector: #selector(preferredColorThemeChanged(\_:)),

name: .didColorThemeChange,

object: nil)

delegate = self

}

override func viewWillAppear(\_ animated: Bool) {

super.viewWillAppear(animated)

setNavigationBarHidden(true, animated: false)

}

public func displayAlertToDismiss(isEndOfPrototype: Bool = false) {

let title, message: String

if isEndOfPrototype {

title = "You're about to exit the Prototype."

message = "Are you sure you want to proceed this action?"

}else{

title = "This is the end of this prototype."

message = "Are you sure you want to exit the prototype?"

}

let isIPhone = UIDevice.current.userInterfaceIdiom == .phone

let alert = UIAlertController(title: title,

message: message,

preferredStyle: isIPhone ? .actionSheet : .alert)

alert.addAction(UIAlertAction(title: "Exit",

style: .destructive,

handler: { \_ in

self.dismiss(animated: true, completion: nil)

}))

alert.addAction(UIAlertAction(title: "Cancel",

style: .cancel,

handler: nil))

self.present(alert, animated: true, completion: nil)

}

@objc func preferredColorThemeChanged(\_ notification: Notification) {

view.tintColor = currentColorTheme.componentColor.callToAction

}

}

extension NavigationController: UIViewControllerTransitioningDelegate, UINavigationControllerDelegate {

func navigationController(\_ navigationController: UINavigationController, animationControllerFor operation: UINavigationController.Operation, from fromVC: UIViewController, to toVC: UIViewController) -> UIViewControllerAnimatedTransitioning? {

return ViewControllerAnimatedTransitioning(isPresenting: operation == .push)

}

}

//

// BookingSummaryViewController.swift

import UIKit

import Firebase

import FirebaseDatabase

class BookingConfirmationViewController: BookingViewController {

var ref: DatabaseReference! = Database.database().reference()

var thistrainSchedule:TrainSchedule?

var seatClassEntity: SeatClassEntity?

var summaryPreviewView: SummaryPreviewView!

var mainCardView: CardControl!

var dateLabelContainerView: UIView!

var seatClassType: SeatClassType?

override func viewDidLoad() {

super.viewDidLoad()

setupStaticContent()

//mainCardView.isHidden = true

}

override func setupView() {

super.setupView()

mainViewType = .view

dateLabelContainerView = UIView(containingView: dateLabel,

withConstaintEquals: [.centerHorizontal, .top, .bottom],

insetsConstant: .init(top: 4, leading: 0, bottom: 0, trailing: 0))

let placeholderView = UIView()

mainStackView.removeArrangedSubview(datePlaceholderLabel)

mainStackView.removeArrangedSubview(mainContentView)

mainStackView.insertArrangedSubview(mainContentView, at: 0)

mainStackView.insertArrangedSubview(dateLabelContainerView, at: 1)

mainStackView.addArrangedSubview(UIView())

mainStackView.addArrangedSubview(placeholderView)

view.addSubview(datePlaceholderLabel, withConstaintEquals: [.topSafeArea, .centerHorizontal])

view.addConstraints(toView: datePlaceholderLabel, withConstaintGreaterThanOrEquals: [.leadingMargin, .trailingMargin])

let datePlaceholderLabelWidthConstraint = datePlaceholderLabel.widthAnchor.constraint(equalToConstant: DesignSystem.layout.maximumWidth)

datePlaceholderLabelWidthConstraint.priority = .defaultHigh

datePlaceholderLabelWidthConstraint.isActive = true

mainCardView = CardControl(type: .large)

mainContentView.addSubview(mainCardView,

withConstaintEquals: .edges,

insetsConstant: .init(top: -mainCardView.layer.cornerRadius))

mainContentView.heightAnchor.constraint(lessThanOrEqualTo: mainContentView.widthAnchor, multiplier: 1).isActive = true

summaryPreviewView = SummaryPreviewView()

summaryPreviewView.setupContent(withSeatClassType: seatClassType)

// Setup Main Card View

mainCardView.contentView

.addSubview(summaryPreviewView,

withConstaintEquals: .edges)

mainCardView.contentView

.isUserInteractionEnabled = true

placeholderView.translatesAutoresizingMaskIntoConstraints = false

placeholderView.topAnchor.constraint(equalTo: mainCallToActionButton.topAnchor).isActive = true

setHeaderInformationValue(headerInformation)

view.bringSubviewToFront(backButton)

}

override func setupTheme() {

super.setupTheme()

}

override func setupInteraction() {

super.setupInteraction()

mainCallToActionButton.addTarget(self,

action: #selector(mainCallToActionButtonDidTouch(\_:)),

for: .touchUpInside)

backButton.addTarget(self,

action: #selector(backButtonDidTouch(\_:)),

for: .touchUpInside)

}

override func setHeaderInformationValue(\_ headerInformation: HeaderInformation?) {

super.setHeaderInformationValue(headerInformation)

}

private func setupStaticContent() {

//mainCallToActionButton.setTitle("Purchase this Ticket—\(headerInformation?.price ?? "")")

mainCallToActionButton.setTitle("Purchase this Ticket")

}

@objc func mainCallToActionButtonDidTouch(\_ sender: Button) {

//guard let navigationController = navigationController as? NavigationController else { return }

//navigationController.displayAlertToDismiss(isEndOfPrototype: true)

// let controller = UIAlertController(title: "預訂成功", message: "", preferredStyle: .alert)

// let okAction = UIAlertAction(title: "OK", style: .default, handler: nil)

// controller.addAction(okAction)

// present(controller, animated: true, completion: nil)

let sumvc = SummaryViewController()

var thisticket = Ticket(Train: thistrainSchedule!, seat:seatClassEntity!,head:self.headerInformation!)

Myticket.append(thisticket)

//self.navigationController?.pushViewController(sumvc, animated: true)

//self.present(sumvc, animated: true, completion: nil)

print(Myticket)

let userID = Auth.auth().currentUser?.uid

self.ref.child("users").child(userID!).setValue(["IDCARD": idcard])

for i in Myticket{

TotalTicketlist.ticketlist.append(i)

}

self.ref.child("users").child(userID!).child("tickets").setValue(TotalTicketlist.convertToDict())

self.navigationController?.popToRootViewController(animated: true)

}

@objc func backButtonDidTouch(\_ sender: Button) {

navigationController?.popViewController(animated: true)

}

}//

// BookingCriteriaViewController.swift

import UIKit

class BookingCriteriaViewController: BookingViewController,DateTimePickerDelegate,UITextFieldDelegate {

var fromStation = "台北"

var destinationStation = "台南"

let traintypes = ["普悠瑪號","太魯閣號","自強號","莒光號","區間車"]

var stackView: UIStackView!

var headerStackView: UIStackView!

var inputStackView: UIStackView!

var headlineLabel: Label!

var logoImageView: UIImageView!

var horizontalStationStackView: UIStackView!

var fromStationContainerView: HeadlineWithContainerView!

var fromStationCardControl: StationCardControl!

var destinationStationContainerView: HeadlineWithContainerView!

var destinationStationCardControl: StationCardControl!

var arrowImageView: UIImageView!

var dateSegmentedContainerView: HeadlineWithContainerView!

var timeSegmentedContainerView: HeadlineWithContainerView!

var trainTypeContainerView: HeadlineWithContainerView!

var dateSegmentedControl : SegmentedControl!

var timeSegmentedControl: SegmentedControl!

var trainTypeControl: SegmentedControl!

static let errorGeneratorFeedback = UINotificationFeedbackGenerator()

private var logoImageAlignmentConstraint: NSLayoutConstraint!

override func viewDidLoad() {

//trainTypeContainerView.isHidden = false

super.viewDidLoad()

setupStaticContent()

}

override func setupView() {

super.setupView()

dateLabel.isHidden = true

headerRouteInformationView.isHidden = true

mainStackView.isHidden = true

backButton.isHidden = true

//

headlineLabel = Label()

headlineLabel.numberOfLines = 0

logoImageView = UIImageView()

logoImageView.setContentHuggingPriority(.required, for: .horizontal)

let logoImageContainerView = UIView(containingView: logoImageView, withConstaintEquals: [.leading, .trailing])

headerStackView = UIStackView([headlineLabel, logoImageContainerView],

axis: .horizontal,

distribution: .fill,

alignment: .top)

headerStackView.isHidden=true

logoImageAlignmentConstraint = logoImageView.topAnchor.constraint(equalTo: headlineLabel.firstBaselineAnchor)

logoImageAlignmentConstraint.isActive = true

fromStationCardControl = StationCardControl()

destinationStationCardControl = StationCardControl()

fromStationContainerView = HeadlineWithContainerView(containingView: fromStationCardControl)

destinationStationContainerView = HeadlineWithContainerView(containingView: destinationStationCardControl)

arrowImageView = UIImageView()

arrowImageView.setContentHuggingPriority(.required, for: .horizontal)

let arrowImageContainerView = UIView(containingView: arrowImageView, withConstaintEquals: [.leading, .trailing])

horizontalStationStackView = UIStackView([fromStationContainerView,

arrowImageContainerView,

destinationStationContainerView],

axis: .horizontal,

distribution: .fill,

alignment: .fill,

spacing: 12)

arrowImageView.centerYAnchor.constraint(equalTo: fromStationContainerView.view.centerYAnchor).isActive = true

fromStationContainerView.widthAnchor.constraint(equalTo: destinationStationContainerView.widthAnchor).isActive = true

dateSegmentedControl = SegmentedControl()

dateSegmentedContainerView = HeadlineWithContainerView(containingView: dateSegmentedControl)

timeSegmentedControl = SegmentedControl()

timeSegmentedContainerView = HeadlineWithContainerView(containingView: timeSegmentedControl)

trainTypeControl = SegmentedControl()

trainTypeContainerView = HeadlineWithContainerView(containingView: trainTypeControl)

inputStackView = UIStackView([horizontalStationStackView,

dateSegmentedContainerView,

timeSegmentedContainerView,

trainTypeContainerView],

axis: .vertical,

distribution: .fill,

alignment: .fill,

spacing: 24)

stackView = UIStackView([headerStackView,

inputStackView],

axis: .vertical,

distribution: .fill,

alignment: .fill, spacing: DesignSystem.isNarrowScreen ? 24 : 48)

view.addSubview(stackView,

withConstaintEquals: [.topSafeArea, .centerHorizontal],

insetsConstant: .init(top: 24))

view.addConstraints(toView: stackView, withConstaintGreaterThanOrEquals: [.leadingMargin, .trailingMargin])

let stackViewWidthConstraint = stackView.widthAnchor.constraint(equalToConstant: DesignSystem.layout.maximumWidth)

stackViewWidthConstraint.priority = .init(999)

stackViewWidthConstraint.isActive = true

}

func textFieldShouldReturn(textField: UITextField) -> Bool {

// 結束編輯 把鍵盤隱藏起來

self.view.endEditing(true)

return true

}

override func setupTheme() {

super.setupTheme()

headlineLabel.textStyle = textStyle.largeTitle()

headlineLabel.textColor = currentColorTheme.componentColor.callToAction

//headlineLabel.isHidden=true

logoImageAlignmentConstraint.constant = -textStyle.largeTitle().font.capHeight

fromStationContainerView.setupTheme()

destinationStationContainerView.setupTheme()

dateSegmentedContainerView.setupTheme()

timeSegmentedContainerView.setupTheme()

trainTypeContainerView.setupTheme()

fromStationCardControl.setupTheme()

destinationStationCardControl.setupTheme()

dateSegmentedControl.setupTheme()

timeSegmentedControl.setupTheme()

}

override func setupInteraction() {

super.setupInteraction()

//MARK:車站選擇button函式

fromStationCardControl

.addTarget(self,

action: #selector(stationCardControlDidTouch),

for: .touchUpInside)

destinationStationCardControl

.addTarget(self,

action: #selector(stationCardControlDidTouch2),

for: .touchUpInside)

dateSegmentedControl

.addTarget(self,

action: #selector(reloadTimeSegemtnedControl),

for: .valueChanged)

mainCallToActionButton

.addTarget(self,

action: #selector(mainCallToActionButtonDidTouch(\_:)),

for: .touchUpInside)

}

func setupStaticContent() {

//MARK:標題文字

headlineLabel.text = "Reserve \nShinkansen \nTickets"

logoImageView.image = #imageLiteral(resourceName: "Logo JR East")

arrowImageView.image = #imageLiteral(resourceName: "Icon Arrow Right")

fromStationContainerView.setTitle(title: "From")

destinationStationContainerView.setTitle(title: "Destination")

//MARK:車站選擇

fromStationCardControl.setupValue(stationNameJP: fromStation, stationName: fromStation)

destinationStationCardControl.setupValue(stationNameJP: destinationStation, stationName: destinationStation)

dateSegmentedContainerView.setTitle(title: "Date")

let today = Date()

let tomorrow = today.addingTimeInterval(60 \* 60 \* 24)

let formatter = FullDateFormatter()

//MARK: 日期片段

dateSegmentedControl.items = [(title: "Today", subtitle: formatter.string(from: today), true),

(title: "Tomorrow", subtitle: formatter.string(from: tomorrow), true),

(title: "Pick a Date", subtitle: " ", true)]

dateSegmentedControl

.addTarget(self,

action: #selector(dateSelect),

for: .touchUpInside)

timeSegmentedContainerView.setTitle(title: "Time")

trainTypeContainerView.setTitle(title: "Train")

reloadTimeSegemtnedControl()

reloadtrainTypeControl()

mainCallToActionButton.setTitle("Search for Tickets")

}

@objc private func stationCardControlDidTouch() {

let view = StationSelectTableViewController()

view.type = 0

view.mainvc = self

self.present(view, animated: true, completion: nil)

}

@objc private func stationCardControlDidTouch2() {

let view = StationSelectTableViewController()

view.type = 1

view.mainvc = self

self.present(view, animated: true, completion: nil)

}

@objc private func dateSelect(){

let min = Date().addingTimeInterval(-60 \* 60 \* 24 \* 4)

let max = Date().addingTimeInterval(60 \* 60 \* 24 \* 4)

let picker = DateTimePicker.create(minimumDate: min, maximumDate: max)

// customize your picker

// picker.timeInterval = DateTimePicker.MinuteInterval.thirty

// picker.locale = Locale(identifier: "en\_GB")

//

// picker.todayButtonTitle = "Today"

// picker.is12HourFormat = true

// picker.dateFormat = "hh:mm aa dd/MM/YYYY"

// picker.isTimePickerOnly = true

picker.includeMonth = true // if true the month shows at bottom of date cell

picker.highlightColor = UIColor(red: 255.0/255.0, green: 138.0/255.0, blue: 138.0/255.0, alpha: 1)

picker.darkColor = UIColor.darkGray

picker.doneButtonTitle = "SELECT"

picker.doneBackgroundColor = UIColor(red: 255.0/255.0, green: 138.0/255.0, blue: 138.0/255.0, alpha: 1)

picker.completionHandler = { date in

let formatter = DateFormatter()

//formatter.dateFormat = "hh:mm aa dd/MM/YYYY"

formatter.dateFormat = "E, d MMM yyyy HH:mm:ss Z"

//self.title = formatter.string(from: date)

//self.CreateTime = formatter.string(from: date)

}

picker.delegate = self

// add picker to your view

// don't try to make customize width and height of the picker,

// you'll end up with corrupted looking UI

// picker.frame = CGRect(x: 0, y: 100, width: picker.frame.size.width, height: picker.frame.size.height)

// set a dismissHandler if necessary

// picker.dismissHandler = {

// picker.removeFromSuperview()

// }

// self.view.addSubview(picker)

// or show it like a modal

if(dateSegmentedControl.selectedIndex==2){

picker.show()

}

}

func dateTimePicker(\_ picker: DateTimePicker, didSelectDate: Date) {

//dateSegmentedControl.items[2] = [(title: "Pick a Date", subtitle: "\(picker.selectedDateString) ", true)]

// dateSegmentedControl.items = [(title: "Today", subtitle: formatter.string(from: today), true),

// (title: "Tomorrow", subtitle: formatter.string(from: tomorrow), true),

// (title: "Pick a Date", subtitle: "\(picker.selectedDateString) ", true)]

let today = Date()

let tomorrow = today.addingTimeInterval(60 \* 60 \* 24)

let formatter = FullDateFormatter()

print(picker.selectedDateString)

dateSegmentedControl.items = [(title: "Today", subtitle: formatter.string(from: today), true),

(title: "Tomorrow", subtitle: formatter.string(from: tomorrow), true),

(title: "Pick a Date", subtitle: "\(picker.selectedDateString)", true)]

}

@objc private func reloadTimeSegemtnedControl() {

let timeInterval: TimeInterval

switch dateSegmentedControl.selectedIndex {

case 0:

timeInterval = 0

case 1:

timeInterval = 60 \* 60 \* 24

default:

timeInterval = 60 \* 60 \* 24

// BookingCriteriaViewController

// .errorGeneratorFeedback

// .notificationOccurred(.error)

// showErrorMessage("Oops! Picking a particular date feature doesn't work as this prototype doesn't have it implemented yet.")

}

let morning = (Date(byHourOf: 6)...Date(byHourOf: 12)).addingTimeInterval(timeInterval)

let afternoon = (Date(byHourOf: 12)...Date(byHourOf: 18)).addingTimeInterval(timeInterval)

let evening = (Date(byHourOf: 18)...Date(byHourOf: 24)).addingTimeInterval(timeInterval)

let now = Date()

//MARK:時間選擇

timeSegmentedControl.items = [(title: "Morning",

subtitle: morning.toString(),

isEnabled: now < morning.upperBound),

(title: "Afternoon",

subtitle: afternoon.toString(),

isEnabled: now < afternoon.upperBound),

(title: "Evening",

subtitle: evening.toString(),

isEnabled: now < evening.upperBound)]

}

@objc private func reloadtrainTypeControl() {

//普悠瑪號/太魯閣號/自強號/莒光號/區間車

\_ = ["普悠瑪號","太魯閣號","自強號","莒光號","區間車"]

trainTypeControl.items = [(title: "普悠瑪",

subtitle: "",

isEnabled: true),

(title: "太魯閣",

subtitle: "",

isEnabled: true),

(title: "自強號",

subtitle: "",

isEnabled: true),

(title: "莒光號",

subtitle: "",

isEnabled: true),

(title: "區間車",

subtitle: "",

isEnabled: true)]

}

@objc func mainCallToActionButtonDidTouch(\_ sender: Button) {

trainTypeContainerView.isHidden = true

let dateOffset: TimeInterval

switch dateSegmentedControl.selectedIndex {

case 0:

dateOffset = 0

case 1:

dateOffset = 60 \* 60 \* 24

default:

dateOffset = 60 \* 60 \* 24 \* 2

}

let selectedDate = Date(timeIntervalSinceNow: dateOffset)

let timeOffset: TimeInterval

switch timeSegmentedControl.selectedIndex {

case 0:

timeOffset = 0

case 1:

timeOffset = TimeInterval(60 \* 60 \* 6)

default:

timeOffset = TimeInterval(60 \* 60 \* 12)

}

let trainSelectionVC = TrainSelectionViewController()

let formatter = FullDateFormatter()

let dayOfWeek = Calendar.current.weekdaySymbols[Calendar.current.component(.weekday, from: selectedDate) - 1]

let date = formatter.string(from: selectedDate)

let fromStation = self.fromStation

let toStation = self.destinationStation

//let head = HeadInfo(dayOfWeek:dayOfWeek,date:date,fromStation:fromStation,toStation:toStation)

trainSelectionVC.headerInformation =

HeaderInformation(dayOfWeek: dayOfWeek,

date: date,

fromStation: fromStation,

toStation: toStation)

trainSelectionVC.dateOffset = dateOffset

trainSelectionVC.timeOffset = timeOffset

trainSelectionVC.traintype = traintypes[trainTypeControl.selectedIndex]

trainSelectionVC.mainvc = self

navigationController?.pushViewController(trainSelectionVC, animated: true)

}

}

extension ClosedRange where Bound == Date {

func toString() -> String {

return "\(lowerBound.timeHour) - \(upperBound.timeHour)"

}

func addingTimeInterval(\_ timeInterval: TimeInterval) -> ClosedRange {

return lowerBound.addingTimeInterval(timeInterval)...upperBound.addingTimeInterval(timeInterval)

}

}//

// BookingViewController.swift

import UIKit

struct HeaderInformation:Codable{

var dayOfWeek: String

var date: String

var fromStation: String

var fromTime: String? = nil

var toStation: String

var toTime: String? = nil

var trainNumber: String? = nil

var trainName: String? = nil

var carNumber: String? = nil

var className: String? = nil

var seatNumber: String? = nil

var price: String? = nil

init(dayOfWeek: String, date: String, fromStation: String, toStation: String) {

self.dayOfWeek = dayOfWeek

self.date = date

self.fromStation = fromStation

self.toStation = toStation

fromTime = nil

toTime = nil

trainNumber = nil

trainName = nil

carNumber = nil

className = nil

seatNumber = nil

price = nil

}

}

class BookingViewController: ViewController {

enum MainViewType {

case tableView

case view

}

var headerInformation: HeaderInformation? {

didSet {

setHeaderInformationValue(headerInformation)

}

}

var mainViewType: MainViewType = .tableView {

didSet {

switch mainViewType {

case .tableView:

mainTableView.isHidden = false

mainContentView.isHidden = true

mainCallToActionButton.isHidden = true

case .view:

mainTableView.isHidden = true

mainContentView.isHidden = false

mainCallToActionButton.isHidden = false

}

}

}

static let feedbackGenerator = UIImpactFeedbackGenerator(style: .medium)

var isPopPerforming: Bool = false {

didSet {

if oldValue != isPopPerforming && isPopPerforming == true {

navigationController?.popViewController(animated: true)

BookingViewController.feedbackGenerator.impactOccurred()

}

}

}

var mainCallToActionButton: Button!

var mainStackView: UIStackView!

var headerWithTopBarStackView: UIStackView!

var backButton: BackButtonControl!

var dateLabel: Label!

var datePlaceholderLabel: Label!

var headerRouteInformationView: HeaderRouteInformationView!

var mainContentView: UIView!

var mainTableView: UITableView!

var interactivePopOverlayView: InteractivePopOverlayView!

override func viewDidLoad() {

super.viewDidLoad()

}

override func viewWillDisappear(\_ animated: Bool) {

super.viewWillDisappear(animated)

view.subviews.forEach {

if $0.tag == 1 << 3 {

$0.removeFromSuperview()

}

}

}

override func setupView() {

super.setupView()

dateLabel = Label()

datePlaceholderLabel = Label()

datePlaceholderLabel

.translatesAutoresizingMaskIntoConstraints = false

datePlaceholderLabel.heightAnchor

.constraint(equalToConstant: 44)

.isActive = true

datePlaceholderLabel.text = " "

headerRouteInformationView = HeaderRouteInformationView(fromStation: " ", toStation: " ")

headerWithTopBarStackView = UIStackView([datePlaceholderLabel,

headerRouteInformationView],

axis: .vertical,

distribution: .fill,

alignment: .fill,

spacing: 8)

let headerWithTopBarContainerView = UIView(containingView: headerWithTopBarStackView, withConstaintEquals: [.topSafeArea, .leadingMargin, .trailingMargin, .bottom])

headerWithTopBarContainerView.preservesSuperviewLayoutMargins = true

let headerWithTopBarContainerViewWidthConstraint = headerWithTopBarContainerView.widthAnchor.constraint(equalToConstant: DesignSystem.layout.maximumWidth)

headerWithTopBarContainerViewWidthConstraint.priority = .defaultHigh

headerWithTopBarContainerViewWidthConstraint.isActive = true

mainContentView = UIView()

mainContentView.preservesSuperviewLayoutMargins = true

mainContentView.backgroundColor = .clear

mainContentView.isHidden = true

mainTableView = UITableView(frame: .zero, style: .plain)

mainTableView.preservesSuperviewLayoutMargins = true

mainTableView.showsVerticalScrollIndicator = false

mainTableView.backgroundColor = .clear

mainTableView.separatorStyle = .none

mainTableView.contentInset.top = 12

mainTableView.contentOffset.y = -12

mainTableView.delegate = self

let mainTableViewWidthConstraint = mainTableView.widthAnchor.constraint(equalToConstant: DesignSystem.layout.maximumWidth)

mainTableViewWidthConstraint.priority = .defaultHigh

mainTableViewWidthConstraint.isActive = true

mainStackView = UIStackView([headerWithTopBarContainerView,

mainContentView,

mainTableView],

axis: .vertical,

distribution: .fill,

alignment: .center,

spacing: 20)

mainStackView.preservesSuperviewLayoutMargins = true

mainContentView.widthAnchor.constraint(equalTo: mainStackView.widthAnchor).isActive = true

interactivePopOverlayView = InteractivePopOverlayView()

view.addSubview(interactivePopOverlayView, withConstaintEquals: .edges)

view.addSubview(mainStackView, withConstaintEquals: .edges)

// MARK: Setup Button

mainCallToActionButton = Button(type: .contained)

view.addSubview(mainCallToActionButton, withConstaintEquals: .centerHorizontal)

view.addConstraints(toView: mainCallToActionButton, withConstaintGreaterThanOrEquals: [.leadingMargin, .trailingMargin])

view.constraintBottomSafeArea(to: mainCallToActionButton,

withGreaterThanConstant: 16,

minimunConstant: 8)

let mainCallToActionButtonWidthConstraint = mainCallToActionButton.widthAnchor.constraint(equalToConstant: DesignSystem.layout.maximumWidth)

mainCallToActionButtonWidthConstraint.priority = .defaultHigh

mainCallToActionButtonWidthConstraint.isActive = true

view.addSubview(dateLabel)

datePlaceholderLabel

.addConstraints(toView: dateLabel,

withConstaintEquals: .center)

backButton = BackButtonControl()

view.addSubview(backButton)

backButton

.centerYAnchor

.constraint(equalTo:

datePlaceholderLabel

.centerYAnchor)

.isActive = true

backButton

.leadingAnchor

.constraint(equalTo:

datePlaceholderLabel

.leadingAnchor,

constant: -10)

.isActive = true

setHeaderInformationValue(headerInformation)

}

override func setupTheme() {

super.setupTheme()

mainCallToActionButton.setupTheme()

backButton.setupTheme()

headerRouteInformationView.setupTheme()

mainTableView.setupTheme()

interactivePopOverlayView.setupTheme()

dateLabel.textStyle = textStyle.headline()

dateLabel.textAlignment = .center

dateLabel.textColor = currentColorTheme.componentColor.primaryText

datePlaceholderLabel.textStyle = dateLabel.textStyle

}

override func setupInteraction() {

super.setupInteraction()

let screenEdgePanGesture =

UIScreenEdgePanGestureRecognizer(target: self,

action: #selector(screenEdgePanGestureDidPan))

screenEdgePanGesture.edges = .left

view.addGestureRecognizer(screenEdgePanGesture)

}

func setHeaderInformationValue(\_ headerInformation: HeaderInformation?) {

guard let headerInformation = headerInformation,

let dateLabel = dateLabel,

let headerRouteInformationView = headerRouteInformationView else { return }

dateLabel.text = "\(headerInformation.dayOfWeek), \(headerInformation.date)"

headerRouteInformationView.setupValue(fromStation: headerInformation.fromStation,

fromTime: headerInformation.fromTime,

toStation: headerInformation.toStation,

toTime: headerInformation.toTime,

trainNumber: headerInformation.trainNumber,

trainName: headerInformation.trainName,

carNumber: headerInformation.carNumber,

className: headerInformation.className,

seatNumber: headerInformation.seatNumber,

price: headerInformation.price)

}

func showErrorMessage(\_ message: String, delay: TimeInterval = 3) {

view.subviews.forEach {

if $0.tag == 1 << 3 {

$0.removeFromSuperview()

}

}

let containerView = UIView()

containerView.tag = 1 << 3

containerView.preservesSuperviewLayoutMargins = true

containerView.backgroundColor = currentColorTheme.componentColor.errorBackground

view.addSubview(containerView,

withConstaintEquals: [.top,

.leading,

.trailing])

let label = Label()

label.textStyle = textStyle.caption1()

label.numberOfLines = 0

label.textColor = currentColorTheme.componentColor.primaryText

label.textAlignment = .center

label.text = message

containerView.addSubview(label,

withConstaintEquals: [.topMargin,

.bottomMargin,

.centerHorizontal],

insetsConstant: .init(bottom: 8))

containerView.addConstraints(toView: label,

withConstaintGreaterThanOrEquals: [.leadingMargin, .trailingMargin])

let labelWidthConstraint = label.widthAnchor.constraint(equalToConstant: DesignSystem.layout.maximumWidth)

labelWidthConstraint.priority = .defaultHigh

labelWidthConstraint.isActive = true

/// Animating Container View

containerView.layoutIfNeeded()

containerView.transform.ty = -containerView.bounds.height

UIView.animate(withStyle: .normalAnimationStyle,

animations: {

containerView.transform.ty = 0

}, completion: {

finished in

UIView.animate(withStyle: .normalAnimationStyle,

delay: delay,

animations: {

containerView.transform.ty = -containerView.bounds.height

},

completion: {

finished in

containerView.removeFromSuperview()

})

})

}

@objc func screenEdgePanGestureDidPan(\_ sender: UIScreenEdgePanGestureRecognizer) {

guard self != navigationController?.viewControllers[0],

!isPopPerforming else { return }

view.bringSubviewToFront(interactivePopOverlayView)

let state = sender.state

let location = sender.location(in: sender.view!)

let translate = CGPoint(x: max(sender.translation(in: sender.view!).x, 0),

y: location.y)

let velocity = sender.velocity(in: sender.view!)

let dismissXTranslateThreshold: CGFloat = view.bounds.width / 5

let alphaXTranslateThreshold: CGFloat = dismissXTranslateThreshold

interactivePopOverlayView.dismissXTranslateThreshold = dismissXTranslateThreshold

func setAlpha(to alpha: CGFloat) {

interactivePopOverlayView.overlayAlpha = 1 - alpha

}

switch state {

case .changed:

if !isPopPerforming {

backButton.transform.tx = min(translate.x / 3, view.bounds.width / 9)

let alpha = max(((1 - DesignSystem.alpha.disabled) \*

(1 - translate.x / alphaXTranslateThreshold)), 0) +

DesignSystem.alpha.disabled

setAlpha(to: alpha)

interactivePopOverlayView.currentTranslation = translate

}

case .ended:

UIView

.animate(withStyle: .normalAnimationStyle,

animations: {

[weak self] in

self?.interactivePopOverlayView.currentTranslation?.x = 0

setAlpha(to: 1)

self?.backButton.transform.tx = 0

})

if velocity.x > 128 {

isPopPerforming = true

} else {

if translate.x > dismissXTranslateThreshold {

isPopPerforming = true

} else {

}

}

default:

break

}

}

}

extension BookingViewController: UITableViewDelegate {

func scrollViewDidScroll(\_ scrollView: UIScrollView) {

let verticalContentOffset = scrollView.contentOffset.y + scrollView.contentInset.top

if !isPopPerforming {

headerRouteInformationView.verticalRubberBandEffect(byVerticalContentOffset: verticalContentOffset)

}

}

}//

// TrainSelectionViewController.swift

import UIKit

import Kumi

class MyticketViewcontroller: BookingViewController {

//var traintype = "自強號"

var didFirstLoad: Bool = false

var selectedIndexPath: IndexPath?

var loadingActivityIndicatorView: UIActivityIndicatorView!

// var trainCriteria: TrainCriteria? {

// didSet {

// let now = Date()

// trainSchedules = (trainCriteria?.trainSchedules ?? []).filter {

// let component = Calendar.current.dateComponents(in: Date.JPCalendar.timeZone, from: $0.fromTime)

// let date = Date(byHourOf: component.hour, minute: component.minute, second: component.second).addingTimeInterval(dateOffset + timeOffset)

// return now < date

// }

// mainTableView.reloadData()

// }

// }

private var trainSchedules: [TrainSchedule] = []

var dateOffset: TimeInterval = 0

var timeOffset: TimeInterval = 0

override func viewDidLoad() {

super.viewDidLoad()

setupTableView()

self.headerRouteInformationView.isHidden = true

//selectTraintype()

}

override func viewDidAppear(\_ animated: Bool) {

super.viewDidAppear(animated)

if !didFirstLoad {

// TrainCriteria.fetchData { [weak self] result in

// if case .success(let trainCriteria) = result {

// DispatchQueue.main.async {

// [weak self] in

for i in Myticket{

self.trainSchedules.append(i.Train!)

}

print("-----------")

print(trainSchedules)

mainTableView.reloadData()

self.mainTableView.visibleCells.enumerated().forEach { (index, cell) in

guard let cell = cell as? TrainScheduleTableViewCell else { return }

cell.preparePropertiesForAnimation()

cell.transform.ty = 24 \* CGFloat(index)

var animationStyle = UIViewAnimationStyle.transitionAnimationStyle

animationStyle.duration = 0.05 \* TimeInterval(index) + 0.5

UIView.animate(withStyle: animationStyle, animations: {

cell.setPropertiesToIdentity()

cell.transform.ty = 0

})

}

self.didFirstLoad = true

self.mainTableView.isUserInteractionEnabled = true

self.loadingActivityIndicatorView.stopAnimating()

/// Check if there is no result

if self.mainTableView.visibleCells.count == 0 {

let label = Label()

label.numberOfLines = 0

label.text = "No train scheduled in the time range you selected"

label.textStyle = textStyle.caption1()

label.textColor = currentColorTheme.componentColor.secondaryText

label.textAlignment = .center

self.mainTableView

.addSubview(label,

withConstaintEquals: [.center, .trailingMargin, .leadingMargin],

insetsConstant: .init(bottom: 24))

}

// }

//

//

//

// }

//}

}

}

override func setupView() {

super.setupView()

loadingActivityIndicatorView = UIActivityIndicatorView(style: .whiteLarge)

loadingActivityIndicatorView.color = currentColorTheme.componentColor.secondaryText

loadingActivityIndicatorView.startAnimating()

mainViewType = .tableView

mainTableView.isUserInteractionEnabled = false

mainTableView.addSubview(loadingActivityIndicatorView,

withConstaintEquals: .centerSafeArea)

}

override func setupInteraction() {

super.setupInteraction()

backButton.addTarget(self,

action: #selector(backButtonDidTouch(\_:)),

for: .touchUpInside)

}

private func setupTableView() {

mainTableView.dataSource = self

mainTableView.delegate = self

mainTableView.register(TrainScheduleTableViewCell.self, forCellReuseIdentifier: "TrainScheduleTableViewCell")

}

var mainvc: BookingCriteriaViewController?

@objc func backButtonDidTouch(\_ sender: Button) {

mainvc?.trainTypeContainerView.isHidden = false

navigationController?.popViewController(animated: true)

}

}

extension MyticketViewcontroller: UITableViewDataSource {

func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

return trainSchedules.count

}

func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

guard let cell = tableView.dequeueReusableCell(withIdentifier: "TrainScheduleTableViewCell",

for: indexPath) as? TrainScheduleTableViewCell else { return UITableViewCell() }

//selectTraintype()

let trainSchedule = trainSchedules[indexPath.row]

let granClassObject = trainSchedule.seatClasses.first(where: {

$0.seatClass == .granClass

})

let greenObject = trainSchedule.seatClasses.first(where: {

$0.seatClass == .green

})

let ordinaryObject = trainSchedule.seatClasses.first(where: {

$0.seatClass == .ordinary

})

let availableObjects = [granClassObject, greenObject, ordinaryObject].compactMap({$0})

let cheapestPrice = availableObjects.sorted(by: { (classL, classR) -> Bool in

return classL.price < classR.price

}).first?.price

// MARK: Offset of time is only for a sake of mock data

let fromTimeString = trainSchedule.fromTime.addingTimeInterval(timeOffset).time

let toTimeString = trainSchedule.toTime.addingTimeInterval(timeOffset).time

cell.setupValue(time: "\(fromTimeString) – \(toTimeString)",

amountOfTime: trainSchedule.toTime.offset(from: trainSchedule.fromTime),

trainNumber: trainSchedule.trainNumber,

trainName: trainSchedule.trainName,

showGranClassIcon: granClassObject != nil,

isGranClassAvailable: granClassObject?.isAvailable ?? false,

showGreenIcon: greenObject != nil,

isGreenAvailable: greenObject?.isAvailable ?? false,

showOrdinaryIcon: ordinaryObject != nil,

isOrdinaryAvailable: ordinaryObject?.isAvailable ?? false,

price: "from \((cheapestPrice ?? 50)/50 )",

trainImage: UIImage(named: trainSchedule.trainImageName))

cell.contentView.alpha = didFirstLoad ? 1 : 0

//self.traintype

// let sessionId = "this is a test"

// let index = sessionId.index(sessionId.startIndex, offsetBy: 2)

// let prefix = sessionId.substring(to: index)

return cell

}

func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

//

selectedIndexPath = indexPath

let trainSchedule = trainSchedules[indexPath.row]

let seat = Myticket[indexPath.row].seat

let bookingConfirmationVC = RecordDetailViewController()

//bookingConfirmationVC.seatClassType = seatClassEntity?.seatClass

bookingConfirmationVC.seatClassType = Myticket[indexPath.row].seat?.seatClass

bookingConfirmationVC.headerInformation = Myticket[indexPath.row].head

bookingConfirmationVC.headerInformation?.seatNumber = Myticket[indexPath.row].seat?.name

bookingConfirmationVC.headerInformation?.price = headerInformation?.price

bookingConfirmationVC.thistrainSchedule = trainSchedule

bookingConfirmationVC.seatClassEntity = Myticket[indexPath.row].seat

bookingConfirmationVC.Index = indexPath.row

bookingConfirmationVC.mainvc = self

navigationController?.pushViewController(bookingConfirmationVC, animated: true)

// let seatClasses = trainSchedule.seatClasses

// let seatClassSelectionVC = SeatClassSelectionViewController()

// seatClassSelectionVC.seatClasses = seatClasses

// seatClassSelectionVC.trainImage = UIImage(named: trainSchedule.trainImageName)

// seatClassSelectionVC.headerInformation = headerInformation

// seatClassSelectionVC.headerInformation?.fromTime = trainSchedule.fromTime.addingTimeInterval(timeOffset).time

// seatClassSelectionVC.headerInformation?.toTime = trainSchedule.toTime.addingTimeInterval(timeOffset).time

// seatClassSelectionVC.headerInformation?.trainNumber = trainSchedule.trainNumber

// seatClassSelectionVC.headerInformation?.trainName = trainSchedule.trainName

//

//

// seatClassSelectionVC.thistrainSchedule = trainSchedule

//navigationController?.pushViewController(seatClassSelectionVC, animated: true)

}

// func selectTraintype(){

// var unremovelist = [Int]()

// var buffer:[TrainSchedule] = []

// if(self.trainSchedules.count>=1){

// print("first")

// for i in 0...(self.trainSchedules.count-1){

// let x = self.trainSchedules[i].trainNumber

// let y = self.traintype

// let index = x.index(x.startIndex , offsetBy: 1)

// let index2 = y.index(y.startIndex, offsetBy: 1)

// let a = x.substring(to: index)

// let b = y.substring(to: index2)

//

// if( a == b){

//// print(x)

//// print(y)

//// print(a)

//// print(b)

//// print("equal")

// unremovelist.append(i)

// //print(self.trainSchedules[i].trainNumber)

// buffer.append(self.trainSchedules[i])

// //self.trainSchedules[i].id = -1

// //cell.removeFromSuperview()

// }

//

// }

// }

// print(unremovelist.count)

// print(self.trainSchedules.count)

// print(unremovelist)

// if(unremovelist.count>=1){

// for i in 0...(unremovelist.count-1){

// print(self.trainSchedules[i].trainNumber)

// buffer.append(self.trainSchedules[i])

// //print(self.trainSchedules[i])

// //self.trainSchedules.remove(at: i)

// }

// }

// self.trainSchedules = buffer

}//

// BookingSummaryViewController.swift

import UIKit

import Firebase

import FirebaseDatabase

class RecordDetailViewController: BookingViewController {

var Index = 0

var thistrainSchedule:TrainSchedule?

var seatClassEntity: SeatClassEntity?

var summaryPreviewView: SummaryPreviewView!

var mainCardView: CardControl!

var dateLabelContainerView: UIView!

var seatClassType: SeatClassType?

var mainvc: MyticketViewcontroller?

override func viewDidLoad() {

super.viewDidLoad()

setupStaticContent()

//mainCardView.isHidden = true

}

override func setupView() {

super.setupView()

mainViewType = .view

dateLabelContainerView = UIView(containingView: dateLabel,

withConstaintEquals: [.centerHorizontal, .top, .bottom],

insetsConstant: .init(top: 4, leading: 0, bottom: 0, trailing: 0))

let placeholderView = UIView()

mainStackView.removeArrangedSubview(datePlaceholderLabel)

mainStackView.removeArrangedSubview(mainContentView)

mainStackView.insertArrangedSubview(mainContentView, at: 0)

mainStackView.insertArrangedSubview(dateLabelContainerView, at: 1)

mainStackView.addArrangedSubview(UIView())

mainStackView.addArrangedSubview(placeholderView)

view.addSubview(datePlaceholderLabel, withConstaintEquals: [.topSafeArea, .centerHorizontal])

view.addConstraints(toView: datePlaceholderLabel, withConstaintGreaterThanOrEquals: [.leadingMargin, .trailingMargin])

let datePlaceholderLabelWidthConstraint = datePlaceholderLabel.widthAnchor.constraint(equalToConstant: DesignSystem.layout.maximumWidth)

datePlaceholderLabelWidthConstraint.priority = .defaultHigh

datePlaceholderLabelWidthConstraint.isActive = true

mainCardView = CardControl(type: .large)

mainContentView.addSubview(mainCardView,

withConstaintEquals: .edges,

insetsConstant: .init(top: -mainCardView.layer.cornerRadius))

mainContentView.heightAnchor.constraint(lessThanOrEqualTo: mainContentView.widthAnchor, multiplier: 1).isActive = true

summaryPreviewView = SummaryPreviewView()

summaryPreviewView.setupContent(withSeatClassType: seatClassType)

// Setup Main Card View

mainCardView.contentView

.addSubview(summaryPreviewView,

withConstaintEquals: .edges)

mainCardView.contentView

.isUserInteractionEnabled = true

placeholderView.translatesAutoresizingMaskIntoConstraints = false

placeholderView.topAnchor.constraint(equalTo: mainCallToActionButton.topAnchor).isActive = true

setHeaderInformationValue(headerInformation)

view.bringSubviewToFront(backButton)

}

override func setupTheme() {

super.setupTheme()

}

override func setupInteraction() {

super.setupInteraction()

mainCallToActionButton.addTarget(self,

action: #selector(mainCallToActionButtonDidTouch(\_:)),

for: .touchUpInside)

backButton.addTarget(self,

action: #selector(backButtonDidTouch(\_:)),

for: .touchUpInside)

}

override func setHeaderInformationValue(\_ headerInformation: HeaderInformation?) {

super.setHeaderInformationValue(headerInformation)

}

private func setupStaticContent() {

mainCallToActionButton.setTitle("Cancel this Ticket")

}

@objc func mainCallToActionButtonDidTouch(\_ sender: Button) {

Myticket.remove(at: Index)

TotalTicketlist.ticketlist.remove(at: Index)

var ref: DatabaseReference! = Database.database().reference()

let userID = Auth.auth().currentUser?.uid

ref.child("users").child(userID!).child("tickets").setValue(TotalTicketlist.convertToDict())

//mainvc?.mainTableView.reloadData()

//navigationController?.popViewController(animated: true)

navigationController?.popToRootViewController(animated: true)

}

@objc func backButtonDidTouch(\_ sender: Button) {

navigationController?.popViewController(animated: true)

}

}//

// SeatClassSelectionViewController.swift

import UIKit

class SeatClassSelectionViewController: BookingViewController {

var thistrainSchedule:TrainSchedule?

var trainImage: UIImage?

var trainImageView: UIImageView!

var selectedIndexPath: IndexPath?

var seatClasses: [SeatClass] = []

var seatMap: SeatMap?

override func viewDidLoad() {

super.viewDidLoad()

setupTableView()

setupStaticContent()

obtainData()

}

override func setupView() {

super.setupView()

mainViewType = .tableView

headerRouteInformationView

.descriptionSetView.addArrangedSubview(UIView())

trainImageView = UIImageView()

headerRouteInformationView

.descriptionSetView

.addSubview(trainImageView)

trainImageView

.translatesAutoresizingMaskIntoConstraints = false

trainImageView

.leadingAnchor

.constraint(

greaterThanOrEqualTo:

headerRouteInformationView

.descriptionSetView

.trainNumberSetView

.trailingAnchor,

constant: 48)

.isActive = true

let trainImageViewTrailingConstraint =

trainImageView

.trailingAnchor

.constraint(

equalTo:

headerRouteInformationView.trailingAnchor)

trainImageViewTrailingConstraint.priority = .defaultLow

trainImageViewTrailingConstraint.isActive = true

trainImageView

.topAnchor

.constraint(

equalTo: headerRouteInformationView

.stationPairView

.bottomAnchor,

constant: 12)

.isActive = true

headerRouteInformationView

.descriptionSetView

.bottomAnchor

.constraint(

equalTo: trainImageView.bottomAnchor)

.isActive = true

trainImageView.widthAnchor.constraint(equalTo: trainImageView.heightAnchor, multiplier: 6).isActive = true

trainImageView

.setContentCompressionResistancePriority(

.init(rawValue: 249), for: .vertical)

trainImageView

.setContentCompressionResistancePriority(

.init(rawValue: 249), for: .horizontal)

}

weak var seatMapSelectionVC: SeatMapSelectionViewController?

func obtainData() {

SeatMap.fetchData { [weak self] result in

if case .success(let seatMap) = result {

DispatchQueue.main.async {

self?.seatMap = seatMap

if let vc = self?.seatMapSelectionVC,

let indexPath = self?.mainTableView?.indexPathForSelectedRow,

let seatClass = self?.seatClasses[indexPath.row],

let selectedEntity = seatMap.seatClassEntities.first(where: {

$0.seatClass == seatClass.seatClass

}){

vc.seatClass = seatClass

vc.seatClassEntity = selectedEntity

vc.seatMap = seatMap

}

}

}

}

let decoder = JSONDecoder()

if let data = NSDataAsset(name: "ModelData")?.data,

let modelData = try? decoder.decode([ModelData].self, from: data) {

NodeFactory.shared =

NodeFactory(modelData: modelData)

}else{

fatalError("There is some errors of trying to phrase JSON, so please check ModelData.json in Assets.xcassets")

}

}

func setupStaticContent() {

trainImageView.image = trainImage

}

override func setupInteraction() {

super.setupInteraction()

backButton.addTarget(self,

action: #selector(backButtonDidTouch(\_:)),

for: .touchUpInside)

}

private func setupTableView() {

mainTableView.dataSource = self

mainTableView.delegate = self

mainTableView.register(SeatClassTableViewCell.self, forCellReuseIdentifier: "SeatClassTableViewCell")

}

@objc func backButtonDidTouch(\_ sender: Button) {

//trainTypeContainerView.isHidden = false

navigationController?.popViewController(animated: true)

}

}

extension SeatClassSelectionViewController: UITableViewDataSource {

func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

return seatClasses.count

}

func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

guard let cell = tableView.dequeueReusableCell(withIdentifier: "SeatClassTableViewCell",

for: indexPath) as? SeatClassTableViewCell else { return UITableViewCell() }

let seatClass = seatClasses[indexPath.row]

cell.setupValue(seatClassType: seatClass.seatClass,

seatClassName: seatClass.seatClass.name,

price: "\(seatClass.price/50)",

description: seatClass.description,

seatImage: seatClass.seatClass.image())

return cell

}

func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

guard !(navigationController?.viewControllers.last is SeatMapSelectionViewController?) else {

return print("Seat map is loading please wait")

}

//

selectedIndexPath = indexPath

let seatClass = seatClasses[indexPath.row]

let selectedEntity = seatMap?.seatClassEntities.first(where: {

$0.seatClass == seatClass.seatClass

})

let seatMapSelectionVC = SeatMapSelectionViewController()

seatMapSelectionVC.seatClass = seatClass

seatMapSelectionVC.seatClassEntity = selectedEntity

seatMapSelectionVC.seatMap = seatMap

seatMapSelectionVC.headerInformation = headerInformation

seatMapSelectionVC.headerInformation?.carNumber = selectedEntity?.carNumber

seatMapSelectionVC.headerInformation?.className = seatClass.name

seatMapSelectionVC.headerInformation?.price = "\(seatClass.price/50)"

if seatMap == nil {

self.seatMapSelectionVC = seatMapSelectionVC

}

seatMapSelectionVC.thistrainSchedule = thistrainSchedule

navigationController?.pushViewController(seatMapSelectionVC, animated: true)

}

}//

// SeatMapSelectionViewController.swift

import UIKit

class SeatMapSelectionViewController: BookingViewController {

var thistrainSchedule:TrainSchedule?

var mainCardView: CardControl!

var seatMapSceneView: SeatMapSceneView!

var selectedSeatID: Int?

var seatClass: SeatClass?

var sceneDidSetup = false

var seatMap: SeatMap? {

didSet {

setupScene()

}

}

var seatClassEntity: SeatClassEntity?

private var isTransitionPerforming: Bool = true

private var selectedEntity: ReservableEntity? {

didSet {

if headerInformation?.carNumber != selectedEntity?.carNumber {

headerInformation?.carNumber = selectedEntity?.carNumber

headerRouteInformationView.descriptionSetView.carNumberSetView.alpha = 0

}

mainCallToActionButton.isEnabled = selectedEntity != nil

//let buttonTitle = "Continue with seat \(selectedEntity?.name ?? "\*")"

let buttonTitle = "Continue with seat"

if mainCallToActionButton.titleLabel?.text != buttonTitle {

mainCallToActionButton.setTitle(buttonTitle)

mainCallToActionButton.titleLabel?.alpha = 0

}

UIView.animate(withDuration: 0.35,

animations: {

self.headerRouteInformationView

.descriptionSetView

.carNumberSetView.alpha = 1

self.mainCallToActionButton

.titleLabel?.alpha = 1

})

}

}

deinit {

mainCardView.contentView.motionEffects.removeAll()

seatMapSceneView.removeFromSuperview()

}

override func viewDidLoad() {

super.viewDidLoad()

setupStaticContent()

//seatClass?.price = (seatClass?.price ?? 50)/50

//seatMapSceneView.isHidden = true

}

override func viewDidAppear(\_ animated: Bool) {

super.viewDidAppear(animated)

isTransitionPerforming = false

}

override func setupView() {

super.setupView()

mainViewType = .view

mainCardView = CardControl(type: .large)

mainContentView.addSubview(mainCardView,

withConstaintEquals: .edges,

insetsConstant: .init(bottom: -mainCardView.layer.cornerRadius))

seatMapSceneView = SeatMapSceneView()

seatMapSceneView.seatMapDelegate = self

mainCardView.contentView.addSubview(seatMapSceneView,

withConstaintEquals: .edges)

mainCardView.contentView.isUserInteractionEnabled = true

mainCallToActionButton.isEnabled = false

setupScene()

}

override func setupInteraction() {

super.setupInteraction()

mainCallToActionButton.addTarget(self,

action: #selector(mainCallToActionButtonDidTouch(\_:)),

for: .touchUpInside)

backButton.addTarget(self,

action: #selector(backButtonDidTouch(\_:)),

for: .touchUpInside)

}

private func setupStaticContent() {

mainCallToActionButton.setTitle("Please Select a Seat")

}

private func setupScene() {

guard let seatMap = seatMap,

let seatMapSceneView = seatMapSceneView,

let seatClassEntity = seatClassEntity, !sceneDidSetup else {

return

}

seatMapSceneView.setupContent(seatMap: seatMap,

currentEntity: seatClassEntity,

fromStation: headerInformation?.fromStation,

toStation: headerInformation?.toStation)

sceneDidSetup = true

}

func verticalRubberBandEffect(byVerticalContentOffset contentOffsetY: CGFloat) {

guard contentOffsetY < 0 else {

mainCardView.transform.ty = 0

headerRouteInformationView.verticalRubberBandEffect(byVerticalContentOffset: 0)

backButton.shapeView.transform.tx = 0

return

}

mainCardView.transform.ty = -contentOffsetY

headerRouteInformationView.verticalRubberBandEffect(byVerticalContentOffset: contentOffsetY)

}

@objc func mainCallToActionButtonDidTouch(\_ sender: Button) {

guard let selectedEntity = selectedEntity else {

return

}

isTransitionPerforming = true

let bookingConfirmationVC = BookingConfirmationViewController()

bookingConfirmationVC.seatClassType = seatClassEntity?.seatClass

bookingConfirmationVC.headerInformation = headerInformation

bookingConfirmationVC.headerInformation?.seatNumber = selectedEntity.name

bookingConfirmationVC.headerInformation?.price = "\((seatClass?.price ?? 50)/50)"

bookingConfirmationVC.thistrainSchedule = thistrainSchedule

bookingConfirmationVC.seatClassEntity = seatClassEntity

navigationController?.pushViewController(bookingConfirmationVC, animated: true)

}

@objc func backButtonDidTouch(\_ sender: Button) {

isTransitionPerforming = true

navigationController?.popViewController(animated: true)

}

}

extension SeatMapSelectionViewController: SeatMapSceneViewDelegate {

func sceneViewDidPanFurtherUpperBoundLimit(by offset: CGPoint) {

if !isPopPerforming {

// Perform interaction

if !isTransitionPerforming {

DispatchQueue.main.async { [weak self] in

self?.verticalRubberBandEffect(byVerticalContentOffset: offset.y)

}

}

}

}

func sceneView(sceneView: SeatMapSceneView, didSelected reservableEntity: ReservableEntity) {

selectedEntity = reservableEntity

}

}////

//// StationSelectTableViewController.swift

//

//import UIKit

//

//class StationSelectTableViewController: UITableViewController {

//

//

// var station = [

// "臺北市","新北市","桃園市","臺中市","臺南市","高雄市",

// "基隆市",

// "新竹市",

// "嘉義市"

//

// ]

// override func viewDidLoad() {

// super.viewDidLoad()

// tableView.register(UITableViewCell.self, forCellReuseIdentifier: "StationCell")

// // Uncomment the following line to preserve selection between presentations

// // self.clearsSelectionOnViewWillAppear = false

//

// // Uncomment the following line to display an Edit button in the navigation bar for this view controller.

// // self.navigationItem.rightBarButtonItem = self.editButtonItem

// }

//

// // MARK: - Table view data source

//

// override func numberOfSections(in tableView: UITableView) -> Int {

// // #warning Incomplete implementation, return the number of sections

// return station.count

// }

//

// override func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

// // #warning Incomplete implementation, return the number of rows

// return 1

// }

//

//

// override func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

// let cell = tableView.dequeueReusableCell(withIdentifier: "StationCell", for: indexPath) as! StationCell

// //let cell = tableView.dequeueReusableCellWithIdentifier(cellIdentifier,forIndexPath: indexPath) as! StationCell

// cell.textLabel?.text = station[indexPath.row]

// // Configure the cell...

//

// return cell

// }

//

//

// /\*

// // Override to support conditional editing of the table view.

// override func tableView(\_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Bool {

// // Return false if you do not want the specified item to be editable.

// return true

// }

// \*/

//

// /\*

// // Override to support editing the table view.

// override func tableView(\_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {

// if editingStyle == .delete {

// // Delete the row from the data source

// tableView.deleteRows(at: [indexPath], with: .fade)

// } else if editingStyle == .insert {

// // Create a new instance of the appropriate class, insert it into the array, and add a new row to the table view

// }

// }

// \*/

//

// /\*

// // Override to support rearranging the table view.

// override func tableView(\_ tableView: UITableView, moveRowAt fromIndexPath: IndexPath, to: IndexPath) {

//

// }

// \*/

//

// /\*

// // Override to support conditional rearranging of the table view.

// override func tableView(\_ tableView: UITableView, canMoveRowAt indexPath: IndexPath) -> Bool {

// // Return false if you do not want the item to be re-orderable.

// return true

// }

// \*/

//

// /\*

// // MARK: - Navigation

//

// // In a storyboard-based application, you will often want to do a little preparation before navigation

// override func prepare(for segue: UIStoryboardSegue, sender: Any?) {

// // Get the new view controller using segue.destination.

// // Pass the selected object to the new view controller.

// }

// \*/

//

//}

import UIKit

class StationSelectTableViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

var type = 0

var tableView = UITableView()

//var tableData = ["Beach", "Clubs", "Chill", "Dance"]

var tableData = [

"臺北市","新北市","桃園市","臺中市","臺南市","高雄市",

"基隆市",

"新竹市",

"嘉義市"

]

override func viewDidLoad() {

super.viewDidLoad()

// Do any additional setup after loading the view, typically from a nib.

tableView = UITableView(frame: self.view.bounds, style: UITableView.Style.plain)

tableView.dataSource = self

tableView.delegate = self

//tableView.backgroundColor = UIColor.white

tableView.register(UITableViewCell.self, forCellReuseIdentifier: "my")

tableView.contentInset.top = 20

let contentSize = self.tableView.contentSize

let footer = UIView(frame: CGRect(x: self.tableView.frame.origin.x,

y: self.tableView.frame.origin.y + contentSize.height,

width: self.tableView.frame.size.width,

height: self.tableView.frame.height - self.tableView.contentSize.height))

self.tableView.tableFooterView = footer

self.tableView.backgroundColor = .black

//self.tableView.setupTheme()

view.addSubview(tableView)

print("type=-------\(type)")

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

// Dispose of any resources that can be recreated.

}

func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

let cell = tableView.dequeueReusableCell(withIdentifier: "my", for: indexPath)

cell.textLabel?.text = " \(tableData[indexPath.row])"

cell.backgroundColor = .black

cell.textLabel?.textColor = .white

return cell

}

func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

return tableData.count

}

// var fromStation = ""

// var destinationStation = ""

var mainvc: BookingCriteriaViewController?

func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

if(type==0){

print(tableData[indexPath.row])

mainvc?.fromStation = tableData[indexPath.row]

mainvc?.fromStationCardControl.setupValue(stationNameJP: tableData[indexPath.row], stationName: tableData[indexPath.row])

}

else{

mainvc?.destinationStation = tableData[indexPath.row]

mainvc?.destinationStationCardControl.setupValue(stationNameJP: tableData[indexPath.row], stationName: tableData[indexPath.row])

}

dismiss(animated: true, completion: nil)

}

}//

// TrainSelectionViewController.swift

import UIKit

import Kumi

class TrainSelectionViewController: BookingViewController {

var traintype = "自強號"

var didFirstLoad: Bool = false

var selectedIndexPath: IndexPath?

var loadingActivityIndicatorView: UIActivityIndicatorView!

var trainCriteria: TrainCriteria? {

didSet {

let now = Date()

trainSchedules = (trainCriteria?.trainSchedules ?? []).filter {

let component = Calendar.current.dateComponents(in: Date.JPCalendar.timeZone, from: $0.fromTime)

let date = Date(byHourOf: component.hour, minute: component.minute, second: component.second).addingTimeInterval(dateOffset + timeOffset)

return now < date

}

mainTableView.reloadData()

}

}

private var trainSchedules: [TrainSchedule] = []

var dateOffset: TimeInterval = 0

var timeOffset: TimeInterval = 0

override func viewDidLoad() {

super.viewDidLoad()

setupTableView()

//selectTraintype()

}

override func viewDidAppear(\_ animated: Bool) {

super.viewDidAppear(animated)

if !didFirstLoad {

TrainCriteria.fetchData { [weak self] result in

if case .success(let trainCriteria) = result {

DispatchQueue.main.async {

[weak self] in

self?.trainCriteria = trainCriteria

self?.mainTableView.visibleCells.enumerated().forEach { (index, cell) in

guard let cell = cell as? TrainScheduleTableViewCell else { return }

cell.preparePropertiesForAnimation()

cell.transform.ty = 24 \* CGFloat(index)

var animationStyle = UIViewAnimationStyle.transitionAnimationStyle

animationStyle.duration = 0.05 \* TimeInterval(index) + 0.5

UIView.animate(withStyle: animationStyle, animations: {

cell.setPropertiesToIdentity()

cell.transform.ty = 0

})

}

self?.didFirstLoad = true

self?.mainTableView.isUserInteractionEnabled = true

self?.loadingActivityIndicatorView.stopAnimating()

/// Check if there is no result

if self?.mainTableView.visibleCells.count == 0 {

let label = Label()

label.numberOfLines = 0

label.text = "No train scheduled in the time range you selected"

label.textStyle = textStyle.caption1()

label.textColor = currentColorTheme.componentColor.secondaryText

label.textAlignment = .center

self?.mainTableView

.addSubview(label,

withConstaintEquals: [.center, .trailingMargin, .leadingMargin],

insetsConstant: .init(bottom: 24))

}

}

}

}

//selectTraintype()

}

}

override func setupView() {

super.setupView()

loadingActivityIndicatorView = UIActivityIndicatorView(style: .whiteLarge)

loadingActivityIndicatorView.color = currentColorTheme.componentColor.secondaryText

loadingActivityIndicatorView.startAnimating()

mainViewType = .tableView

mainTableView.isUserInteractionEnabled = false

mainTableView.addSubview(loadingActivityIndicatorView,

withConstaintEquals: .centerSafeArea)

}

override func setupInteraction() {

super.setupInteraction()

backButton.addTarget(self,

action: #selector(backButtonDidTouch(\_:)),

for: .touchUpInside)

}

private func setupTableView() {

mainTableView.dataSource = self

mainTableView.delegate = self

mainTableView.register(TrainScheduleTableViewCell.self, forCellReuseIdentifier: "TrainScheduleTableViewCell")

}

var mainvc: BookingCriteriaViewController?

@objc func backButtonDidTouch(\_ sender: Button) {

mainvc?.trainTypeContainerView.isHidden = false

navigationController?.popViewController(animated: true)

}

}

extension TrainSelectionViewController: UITableViewDataSource {

func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

selectTraintype()

return trainSchedules.count

}

func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

guard let cell = tableView.dequeueReusableCell(withIdentifier: "TrainScheduleTableViewCell",

for: indexPath) as? TrainScheduleTableViewCell else { return UITableViewCell() }

//selectTraintype()

let trainSchedule = trainSchedules[indexPath.row]

let granClassObject = trainSchedule.seatClasses.first(where: {

$0.seatClass == .granClass

})

let greenObject = trainSchedule.seatClasses.first(where: {

$0.seatClass == .green

})

let ordinaryObject = trainSchedule.seatClasses.first(where: {

$0.seatClass == .ordinary

})

let availableObjects = [granClassObject, greenObject, ordinaryObject].compactMap({$0})

let cheapestPrice = availableObjects.sorted(by: { (classL, classR) -> Bool in

return classL.price < classR.price

}).first?.price

// MARK: Offset of time is only for a sake of mock data

let fromTimeString = trainSchedule.fromTime.addingTimeInterval(timeOffset).time

let toTimeString = trainSchedule.toTime.addingTimeInterval(timeOffset).time

cell.setupValue(time: "\(fromTimeString) – \(toTimeString)",

amountOfTime: trainSchedule.toTime.offset(from: trainSchedule.fromTime),

trainNumber: trainSchedule.trainNumber,

trainName: trainSchedule.trainName,

showGranClassIcon: granClassObject != nil,

isGranClassAvailable: granClassObject?.isAvailable ?? false,

showGreenIcon: greenObject != nil,

isGreenAvailable: greenObject?.isAvailable ?? false,

showOrdinaryIcon: ordinaryObject != nil,

isOrdinaryAvailable: ordinaryObject?.isAvailable ?? false,

price: "from \((cheapestPrice ?? 50)/50 ?? 0)",

trainImage: UIImage(named: trainSchedule.trainImageName))

cell.contentView.alpha = didFirstLoad ? 1 : 0

//self.traintype

// let sessionId = "this is a test"

// let index = sessionId.index(sessionId.startIndex, offsetBy: 2)

// let prefix = sessionId.substring(to: index)

return cell

}

func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

//

selectedIndexPath = indexPath

let trainSchedule = trainSchedules[indexPath.row]

let seatClasses = trainSchedule.seatClasses

let seatClassSelectionVC = SeatClassSelectionViewController()

seatClassSelectionVC.seatClasses = seatClasses

seatClassSelectionVC.trainImage = UIImage(named: trainSchedule.trainImageName)

seatClassSelectionVC.headerInformation = headerInformation

seatClassSelectionVC.headerInformation?.fromTime = trainSchedule.fromTime.addingTimeInterval(timeOffset).time

seatClassSelectionVC.headerInformation?.toTime = trainSchedule.toTime.addingTimeInterval(timeOffset).time

seatClassSelectionVC.headerInformation?.trainNumber = trainSchedule.trainNumber

seatClassSelectionVC.headerInformation?.trainName = trainSchedule.trainName

seatClassSelectionVC.thistrainSchedule = trainSchedule

navigationController?.pushViewController(seatClassSelectionVC, animated: true)

}

func selectTraintype(){

var unremovelist = [Int]()

var buffer:[TrainSchedule] = []

if(self.trainSchedules.count>=1){

print("first")

for i in 0...(self.trainSchedules.count-1){

let x = self.trainSchedules[i].trainNumber

let y = self.traintype

let index = x.index(x.startIndex , offsetBy: 1)

let index2 = y.index(y.startIndex, offsetBy: 1)

let a = x.substring(to: index)

let b = y.substring(to: index2)

if( a == b){

// print(x)

// print(y)

// print(a)

// print(b)

// print("equal")

unremovelist.append(i)

//print(self.trainSchedules[i].trainNumber)

buffer.append(self.trainSchedules[i])

//self.trainSchedules[i].id = -1

//cell.removeFromSuperview()

}

}

}

// print(unremovelist.count)

// print(self.trainSchedules.count)

// print(unremovelist)

// if(unremovelist.count>=1){

// for i in 0...(unremovelist.count-1){

// print(self.trainSchedules[i].trainNumber)

// buffer.append(self.trainSchedules[i])

// //print(self.trainSchedules[i])

// //self.trainSchedules.remove(at: i)

// }

// }

self.trainSchedules = buffer

}

}//

// ViewControllerAnimatedTransitioning.swift

import UIKit

import Kumi

class ViewControllerAnimatedTransitioning: NSObject, UIViewControllerAnimatedTransitioning {

var isPresenting : Bool

lazy var animationStyle: UIViewAnimationStyle = UIViewAnimationStyle.transitionAnimationStyle

lazy var CAAnimationStyle: CABasicAnimationStyle = CABasicAnimationStyle.transitionAnimationStyle

init(isPresenting : Bool) {

self.isPresenting = isPresenting

}

func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) -> TimeInterval {

return animationStyle.duration

}

func animateTransition(using transitionContext: UIViewControllerContextTransitioning) {

let fromViewController = transitionContext.viewController(forKey: UITransitionContextViewControllerKey.from)

let toViewController = transitionContext.viewController(forKey: UITransitionContextViewControllerKey.to)

let container = transitionContext.containerView

guard let fromView = transitionContext.view(forKey: UITransitionContextViewKey.from) else { return }

guard let toView = transitionContext.view(forKey: UITransitionContextViewKey.to) else { return }

isPresenting ? container.addSubview(toView) : container.insertSubview(toView, belowSubview: fromView)

let parentVC = isPresenting ? fromViewController : toViewController

let childVC = isPresenting ? toViewController : fromViewController

// MARK: Set all the views in place and have been processed thru Autolayout

parentVC?.view.layoutIfNeeded()

childVC?.view.layoutIfNeeded()

// MARK: Transition Between BookingViewController

if let fromBookingVC = fromViewController as? BookingViewController,

let toBookingVC = toViewController as? BookingViewController {

// Reset verticalRubberBandEffect

fromBookingVC.headerRouteInformationView.alpha = isPresenting ? 0 : 1

toBookingVC.headerRouteInformationView.alpha = !isPresenting ? 0 : 1

UIView.animate(withStyle: animationStyle, animations: {

fromBookingVC.headerRouteInformationView.verticalRubberBandEffect(byVerticalContentOffset: 0)

fromBookingVC.interactivePopOverlayView.overlayAlpha = 0

}, completion: { \_ in

fromBookingVC.headerRouteInformationView.alpha = toBookingVC is BookingCriteriaViewController ? 0 : 1

toBookingVC.headerRouteInformationView.alpha = 1

})

struct AnimateObject {

let fromVC: BookingViewController

let toVC: BookingViewController

func \_animate(fromView: UIView, fromParentView: UIView,

toView: UIView, toParentView: UIView,

basedHorizontalAnimationOffset: CGFloat = 0,

basedVerticalAnimationOffset: CGFloat = 0,

percentageEndPoint: TimeInterval = 1) {

// Sets all animated views to orginal position before them get calculated

fromView.transform = .identity

toView.transform = .identity

var displacement: CGPoint = .zero

var isTransitioningHiddenView = false

if fromView.isHidden || toView.isHidden {

isTransitioningHiddenView = true

// MARK: Hidden views will be slide up

if fromView.isHidden {

displacement.x = (-basedHorizontalAnimationOffset).systemSizeMuliplier()

displacement.y = (-basedVerticalAnimationOffset).systemSizeMuliplier()

}

if toView.isHidden {

displacement.x = basedHorizontalAnimationOffset.systemSizeMuliplier()

displacement.y = basedVerticalAnimationOffset.systemSizeMuliplier()

}

} else {

let fromActualFrame = fromView.frame(in: fromParentView)

let toActualFrame = toView.frame(in: toParentView)

displacement = CGPoint(x: toActualFrame.midX - fromActualFrame.midX,

y: toActualFrame.minY - fromActualFrame.minY)

}

toView.transform.tx = -displacement.x

toView.transform.ty = -displacement.y

// MARK: In case one view is hidden

if fromView.isHidden {

toView.alpha = 0

}

if toView.isHidden {

fromView.alpha = 1

}

// mutate

let animationStyle = UIViewAnimationStyle.transitionAnimationStyle

var mutatedAnimationStyle = animationStyle

mutatedAnimationStyle.duration = animationStyle.duration \* (fromView.isHidden ? 1 - percentageEndPoint : percentageEndPoint)

mutatedAnimationStyle.delay = animationStyle.duration - mutatedAnimationStyle.duration

UIView.animate(withStyle: isTransitioningHiddenView ? mutatedAnimationStyle : animationStyle,

delay: fromView.isHidden ? mutatedAnimationStyle.delay : 0,

animations: {

fromView.transform.tx = displacement.x

fromView.transform.ty = displacement.y

toView.transform = .identity

// MARK: Animation for one that is hidden

if fromView.isHidden {

toView.alpha = 1

}

if toView.isHidden {

fromView.alpha = 0

}

})

}

func animate(view: ((BookingViewController) -> UIView),

parentView: ((BookingViewController) -> UIView)? = nil,

basedVerticalAnimationOffset: CGFloat,

percentageEndPoint: TimeInterval = 0.4

) {

\_animate(fromView: view(fromVC),

fromParentView: (parentView != nil) ? parentView!(fromVC) : (view(fromVC).superview ?? UIView()),

toView: view(toVC),

toParentView: (parentView != nil) ? parentView!(toVC) : (view(toVC).superview ?? UIView()),

basedVerticalAnimationOffset: basedVerticalAnimationOffset,

percentageEndPoint: 0.4)

}

}

let animateObject = AnimateObject(fromVC: fromBookingVC, toVC: toBookingVC)

animateObject.animate(view: {$0.dateLabel},

parentView: {$0.view},

basedVerticalAnimationOffset: 18)

animateObject.animate(view: {$0.headerRouteInformationView},

parentView: {$0.view},

basedVerticalAnimationOffset: 72)

animateObject.animate(view: {$0.headerRouteInformationView

.stationPairView

.fromStationHeadlineView

.subtitleLabel},

basedVerticalAnimationOffset: 6)

animateObject.animate(view: {$0.headerRouteInformationView

.stationPairView

.toStationHeadlineView

.subtitleLabel},

basedVerticalAnimationOffset: 6)

animateObject.animate(view: {$0.headerRouteInformationView

.descriptionSetView},

basedVerticalAnimationOffset: 18)

animateObject.animate(view: {$0.headerRouteInformationView

.descriptionSetView.carNumberSetView},

basedVerticalAnimationOffset: 12)

animateObject.animate(view: {$0.headerRouteInformationView

.descriptionSetView

.seatNumberSetView},

basedVerticalAnimationOffset: 12)

animateObject.animate(view: {$0.headerRouteInformationView

.descriptionSetView

.priceSetView},

basedVerticalAnimationOffset: 12)

// Animate Back Button

toBookingVC.backButton.transform.tx = isPresenting ? -44 : 44

fromBookingVC.backButton.transform.tx = 0

toBookingVC.backButton.alpha = 0

toBookingVC.backButton.setPath(to: self.isPresenting ? .pullBack : .pushedForward)

toBookingVC.backButton.playAnimation(to: .original, withDuration: 0.35)

fromBookingVC.backButton.playAnimation(to: self.isPresenting ? .pullBack : .pushedForward, withDuration: 0.35)

UIView.animate(withStyle: .halfTransitionAnimationStyle, animations: {

fromBookingVC.backButton.shapeView.transform = .identity

toBookingVC.backButton.transform.tx = 0

fromBookingVC.backButton.transform.tx = self.isPresenting ? 24 : -24

toBookingVC.backButton.alpha = 1

fromBookingVC.backButton.alpha = 0

})

}

let percentageEndPoint: TimeInterval = 0.4

// MARK: Transition in and out views in BookingCriteriaViewController

if let bookingCriterialVC = fromViewController as? BookingCriteriaViewController {

let transitionDirection: UIView.TransitionalDirection = .transitionOut

bookingCriterialVC.headerStackView.translateAndFade(as: transitionDirection,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -72))

bookingCriterialVC.horizontalStationStackView.translateAndFade(as: transitionDirection,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -48))

bookingCriterialVC.dateSegmentedContainerView.translateAndFade(as: transitionDirection,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -32))

bookingCriterialVC.timeSegmentedContainerView.translateAndFade(as: transitionDirection,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -16))

bookingCriterialVC.mainCallToActionButton.translateAndFade(as: transitionDirection,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -16))

} else if let bookingCriterialVC = toViewController as? BookingCriteriaViewController {

let transitionDirection: UIView.TransitionalDirection = .transitionIn

bookingCriterialVC.headerStackView.translateAndFade(as: transitionDirection,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -72))

bookingCriterialVC.horizontalStationStackView.translateAndFade(as: transitionDirection,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -48))

bookingCriterialVC.dateSegmentedContainerView.translateAndFade(as: transitionDirection,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -32))

bookingCriterialVC.timeSegmentedContainerView.translateAndFade(as: transitionDirection,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -16))

bookingCriterialVC.mainCallToActionButton.translateAndFade(as: transitionDirection,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -16))

}

// MARK: Transition in and out views in SeatMapSelectionViewController

if let seatMapSelectionVC = toViewController as? SeatMapSelectionViewController {

seatMapSelectionVC.mainCallToActionButton.translateAndFade(as: .transitionIn,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: seatMapSelectionVC.mainCallToActionButton.bounds.height \* 0.5))

} else if let seatMapSelectionVC = fromViewController as? SeatMapSelectionViewController {

seatMapSelectionVC.mainCallToActionButton.translateAndFade(as: .transitionOut,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: seatMapSelectionVC.mainCallToActionButton.bounds.height \* 0.5))

}

// MARK: Transition in and out views in BookingConfirmationViewController

if let bookingConfirmationVC = toViewController as? BookingConfirmationViewController {

bookingConfirmationVC.mainCallToActionButton.translateAndFade(as: .transitionIn,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -bookingConfirmationVC.mainCallToActionButton.bounds.height \* 0.25))

} else if let bookingConfirmationVC = fromViewController as? BookingConfirmationViewController {

bookingConfirmationVC.mainCallToActionButton.translateAndFade(as: .transitionOut,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -bookingConfirmationVC.mainCallToActionButton.bounds.height \* 0.25))

}

// MARK: Transition In views in TrainSelectionViewController

if let trainSelectionVC = toViewController as? TrainSelectionViewController,

fromViewController is BookingCriteriaViewController {

trainSelectionVC.mainTableView.translateAndFade(as: .transitionIn,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: 128))

}

// MARK: Transition out views in TrainSelectionViewController

if let trainSelectionVC = fromViewController as? TrainSelectionViewController,

toViewController is BookingCriteriaViewController {

trainSelectionVC.mainTableView.translateAndFade(as: .transitionOut,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: 72))

trainSelectionVC.mainTableView.visibleCells.enumerated().forEach {

(index, cell) in

cell.translateAndFade(as: .transitionOut,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: CGFloat(index) \* 16))

}

}

// MARK: Transition views in SeatMapSelectionViewController to BookingConfirmationViewController

if let seatMapSelectionVC = fromViewController as? SeatMapSelectionViewController,

let bookingConfirmationVC = toViewController as? BookingConfirmationViewController {

// seatMapSelectionVC.seatMapSceneView.contentNode.removeAction(forKey: "panDrift")

seatMapSelectionVC.mainCardView.translateAndFade(as: .transitionOut,

animationStyle: animationStyle,

percentageEndPoint: 1,

translate: .init(x: 0, y: seatMapSelectionVC.mainCardView.bounds.height))

bookingConfirmationVC.mainCardView.translateAndFade(as: .transitionIn,

animationStyle: animationStyle,

percentageEndPoint: 0,

translate: .init(x: 0, y: -bookingConfirmationVC.mainCardView.bounds.height))

// bookingConfirmationVC.dateLabel.translateAndFade(as: .transitionIn,

// animationStyle: animationStyle,

// percentageEndPoint: 0,

// translate: .init(x: 0, y: -bookingConfirmationVC.mainCardView.bounds.height + bookingConfirmationVC.dateLabel.bounds.height))

} else if let seatMapSelectionVC = toViewController as? SeatMapSelectionViewController,

let bookingConfirmationVC = fromViewController as? BookingConfirmationViewController {

seatMapSelectionVC.mainCardView.translateAndFade(as: .transitionIn,

animationStyle: animationStyle,

percentageEndPoint: 0,

translate: .init(x: 0, y: seatMapSelectionVC.mainCardView.bounds.height))

bookingConfirmationVC.mainCardView.translateAndFade(as: .transitionOut,

animationStyle: animationStyle,

percentageEndPoint: 1,

translate: .init(x: 0, y: -bookingConfirmationVC.mainCardView.bounds.height))

}

// MARK: Transition views in TrainSelectionViewController to SeatClassSelectionViewController

if let trainSelectionVC = fromViewController as? TrainSelectionViewController,

let seatClassSelectionVC = toViewController as? SeatClassSelectionViewController {

// MARK: Transition Train Image View

seatClassSelectionVC

.trainImageView

.translateAndFade(

as: .transitionIn,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: 18))

// Demo

if let selectedIndexPath = trainSelectionVC.selectedIndexPath {

guard let cell = trainSelectionVC.mainTableView.cellForRow(at: selectedIndexPath) as? TrainScheduleTableViewCell else { return }

// MARK: Fade all other cells

trainSelectionVC.mainTableView.visibleCells.forEach { (otherCell) in

if otherCell != cell {

otherCell.fadeAnimation(as: .transitionOut,

animationStyle: animationStyle,

percentageEndPoint: 0.3)

}

}

// MARK: Fade all other cells

seatClassSelectionVC.mainTableView.visibleCells.forEach { (otherCell) in

otherCell.fadeAnimation(as: .transitionIn,

animationStyle: animationStyle,

percentageEndPoint: 0.55)

}

//

let originRectInOriginView = cell.cardView.frame(in: cell.contentView)

seatClassSelectionVC.mainTableView.visibleCells.forEach { (destinationCell) in

guard let destinationCell = destinationCell as? SeatClassTableViewCell else { return }

let destinationRectInOriginView = destinationCell.cardView.frame(in: cell.contentView)

let dummyView = UIView()

cell.contentView.addSubview(dummyView)

dummyView.layer.setLayer(cell.cardView.layer.layerStyle)

dummyView.layer.withNoShadow()

cell.contentView.layer.setShadow(cell.cardView.layer.shadowStyle)

cell.cardView.layer.removeAllAnimations()

cell.cardView.contentView.layer.removeAllAnimations()

cell.contentView.sendSubviewToBack(dummyView)

if destinationCell == seatClassSelectionVC.mainTableView.visibleCells.last {

cell.cardView.layer.setLayer(LayerStyle())

cell.cardView.contentView.backgroundColor = .clear

cell.cardView.contentView.fadeAnimation(as: .transitionOut, animationStyle: animationStyle, percentageEndPoint: 0.3)

}

dummyView.frame = originRectInOriginView

var muatatedAnimationStyle = animationStyle

muatatedAnimationStyle.duration = animationStyle.duration \* 0.35

UIView.animate(withStyle: muatatedAnimationStyle,

delay: 0.65 \* animationStyle.duration,

animations: {

dummyView.alpha = 0

})

UIView.animate(withStyle: animationStyle, animations: {

dummyView.frame = destinationRectInOriginView

}, completion: {

\_ in

cell.contentView.layer.withNoShadow()

cell.cardView.setupTheme()

dummyView.removeFromSuperview()

})

let originRectInDestinationView = cell.cardView.frame(in: destinationCell.contentView)

let destinationRectInDestinationView = destinationCell.cardView.frame(in: destinationCell.contentView)

destinationCell.cardView.frame = originRectInDestinationView

UIView.animate(withStyle: animationStyle, animations: {

destinationCell.cardView.frame = destinationRectInDestinationView

})

}

}

}

// MARK: Transition views in SeatClassSelectionViewController to TrainSelectionViewController (Backward)

if let trainSelectionVC = toViewController as? TrainSelectionViewController,

let seatClassSelectionVC = fromViewController as? SeatClassSelectionViewController {

// MARK: Transition Train Image View

seatClassSelectionVC

.trainImageView

.translateAndFade(

as: .transitionOut,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: 18))

// Demo

if let selectedIndexPath = trainSelectionVC.selectedIndexPath {

guard let cell = trainSelectionVC.mainTableView.cellForRow(at: selectedIndexPath) as? TrainScheduleTableViewCell else { return }

// MARK: Fade all other cells

trainSelectionVC.mainTableView.visibleCells.forEach { (otherCell) in

if otherCell != cell {

otherCell.fadeAnimation(as: .transitionIn,

animationStyle: animationStyle,

percentageEndPoint: 0.5)

}

}

// MARK: Fade all other cells

seatClassSelectionVC.mainTableView.visibleCells.forEach { (otherCell) in

otherCell.fadeAnimation(as: .transitionOut,

animationStyle: animationStyle,

percentageEndPoint: 0.15)

}

//

let originRectInOriginView = cell.cardView.frame(in: cell.contentView)

seatClassSelectionVC.mainTableView.visibleCells.forEach { (destinationCell) in

guard let destinationCell = destinationCell as? SeatClassTableViewCell else { return }

let destinationRectInOriginView = destinationCell.cardView.frame(in: cell.contentView)

//

cell.cardView.layer.removeAllAnimations()

cell.cardView.contentView.layer.removeAllAnimations()

cell.contentView.layer.setShadow(cell.cardView.layer.shadowStyle)

let dummyView = UIView()

cell.contentView.addSubview(dummyView)

cell.contentView.sendSubviewToBack(dummyView)

dummyView.layer.setLayer(cell.cardView.layer.layerStyle)

dummyView.layer.withNoShadow()

dummyView.frame = destinationRectInOriginView

dummyView.alpha = 0

// Set cell to have no appearance if the cell loops to the last cell

if destinationCell == seatClassSelectionVC.mainTableView.visibleCells.last {

cell.cardView.layer.setLayer(LayerStyle())

cell.cardView.contentView.backgroundColor = .clear

cell.cardView.contentView.fadeAnimation(as: .transitionIn, animationStyle: animationStyle, percentageEndPoint: 0.3)

}

var muatatedAnimationStyle = animationStyle

muatatedAnimationStyle.duration = animationStyle.duration \* 0.05

UIView.animate(withStyle: muatatedAnimationStyle,

animations: {

dummyView.alpha = 1

})

// Animate dummyView's frame

UIView.animate(withStyle: animationStyle, animations: {

dummyView.frame = originRectInOriginView

}, completion: {

\_ in

cell.contentView.layer.withNoShadow()

cell.cardView.setupTheme()

dummyView.removeFromSuperview()

})

//Animation the Destination Cell

let originRectInDestinationView = cell.cardView.frame(in: destinationCell.contentView)

let destinationRectInDestinationView = destinationCell.cardView.frame(in: destinationCell.contentView)

destinationCell.cardView.frame = destinationRectInDestinationView

UIView.animate(withStyle: animationStyle, animations: {

destinationCell.cardView.frame = originRectInDestinationView

})

}

}

}

// MARK: Transition views in SeatClassSelectionViewController to SeatMapSelectionViewController

if let seatClassSelectionVC = fromViewController as? SeatClassSelectionViewController,

let seatMapSelectionVC = toViewController as? SeatMapSelectionViewController {

// MARK: Transition Train Image View

seatClassSelectionVC

.trainImageView

.translateAndFade(

as: .transitionOut,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -18))

seatClassSelectionVC.mainTableView.clipsToBounds = false

// Demo

if let selectedIndexPath = seatClassSelectionVC.selectedIndexPath,

let selectedCell = seatClassSelectionVC.mainTableView.cellForRow(at: selectedIndexPath) as? SeatClassTableViewCell {

seatClassSelectionVC.mainTableView.visibleCells.forEach { (otherCell) in

if otherCell != selectedCell {

otherCell.translateAndFade(as: .transitionOut, animationStyle: animationStyle, percentageEndPoint: 0.3, translate: .zero)

}

}

let originRectInMainContentView = selectedCell.cardView.frame(in: seatMapSelectionVC.mainContentView)

let destinationRectInMainContentView = seatMapSelectionVC.mainCardView.frame(in: seatMapSelectionVC.mainContentView)

let destinationRectInOriginView = seatMapSelectionVC.mainCardView.frame(in: selectedCell.contentView)

if let cardViewInDestinationView = seatMapSelectionVC.mainCardView {

// let cardViewInSelectedCell = selectedCell.cardView

selectedCell.cardView.contentView.layer.removeAllAnimations()

selectedCell.cardView.layer.removeAllAnimations()

// let originalContentView = selectedCell.cardView

let dummyView = UIView()

seatMapSelectionVC.mainContentView.addSubview(dummyView)

seatMapSelectionVC.mainContentView.sendSubviewToBack(dummyView)

dummyView.layer.setLayer(selectedCell.cardView.layer.layerStyle)

dummyView.layer.setShadow(selectedCell.cardView.layer.shadowStyle)

dummyView.frame = originRectInMainContentView

dummyView.alpha = 0

let dummyViewInCell = UIView()

selectedCell.contentView.addSubview(dummyViewInCell)

selectedCell.contentView.sendSubviewToBack(dummyViewInCell)

dummyViewInCell.layer.setLayer(selectedCell.cardView.layer.layerStyle)

dummyViewInCell.layer.setShadow(selectedCell.cardView.layer.shadowStyle)

dummyViewInCell.frame = selectedCell.cardView.frame

dummyViewInCell.alpha = 1

selectedCell.cardView.layer.setLayer(LayerStyle())

selectedCell.cardView.contentView.backgroundColor = .clear

selectedCell.cardView.contentView.fadeAnimation(as: .transitionOut, animationStyle: animationStyle, percentageEndPoint: 0.3)

cardViewInDestinationView.layer.cornerRadius = dummyView.layer.cornerRadius

cardViewInDestinationView.contentView.layer.cornerRadius = dummyView.layer.cornerRadius

cardViewInDestinationView.alpha = 0

cardViewInDestinationView.frame = originRectInMainContentView

cardViewInDestinationView.contentView.frame = cardViewInDestinationView.bounds

cardViewInDestinationView.fadeAnimation(as: .transitionIn, animationStyle: animationStyle, percentageEndPoint: 0.55)

var muatatedAnimationStyle = animationStyle

muatatedAnimationStyle.duration = animationStyle.duration \* 0.3

UIView.animate(withStyle: muatatedAnimationStyle,

delay: 0.3 \* animationStyle.duration,

animations: {

dummyView.alpha = 1

}, completion: {

\_ in

UIView.animate(withStyle: muatatedAnimationStyle,

animations: {

dummyView.alpha = 0

dummyViewInCell.alpha = 0

})

})

// Animate subview in cardView.content of Destination

cardViewInDestinationView.contentView.subviews.forEach { (view) in

view.transform.tx = cardViewInDestinationView.bounds.midX - destinationRectInMainContentView.midX

view.transform.ty = cardViewInDestinationView.bounds.midY - destinationRectInMainContentView.midY

UIView.animate(withStyle: animationStyle, animations: {

view.transform = .identity

})

}

UIView.animate(withStyle: animationStyle, animations: {

cardViewInDestinationView.frame = destinationRectInMainContentView

cardViewInDestinationView.contentView.frame = cardViewInDestinationView.bounds

cardViewInDestinationView.layer.cornerRadius = layerStyle.largeCard.normal().cornerRadius

cardViewInDestinationView.contentView.layer.cornerRadius = layerStyle.largeCard.normal().cornerRadius

dummyView.layer.cornerRadius = layerStyle.largeCard.normal().cornerRadius

dummyViewInCell.layer.cornerRadius = layerStyle.largeCard.normal().cornerRadius

dummyView.frame = destinationRectInMainContentView

dummyViewInCell.frame = destinationRectInOriginView

}, completion: {

\_ in

dummyView.removeFromSuperview()

dummyViewInCell.removeFromSuperview()

cardViewInDestinationView.setupTheme()

selectedCell.cardView.setupTheme()

seatClassSelectionVC.mainTableView.clipsToBounds = true

})

}

}

}

// MARK: Transition views in SeatClassSelectionViewController to SeatMapSelectionViewController

if let seatMapSelectionVC = fromViewController as? SeatMapSelectionViewController ,

let seatClassSelectionVC = toViewController as? SeatClassSelectionViewController{

// MARK: Transition Train Image View

seatClassSelectionVC

.trainImageView

.translateAndFade(

as: .transitionIn,

animationStyle: animationStyle,

percentageEndPoint: percentageEndPoint,

translate: .init(x: 0, y: -18))

seatClassSelectionVC.mainTableView.clipsToBounds = false

// Demo

if let selectedIndexPath = seatClassSelectionVC.selectedIndexPath,

let selectedCell = seatClassSelectionVC.mainTableView.cellForRow(at: selectedIndexPath) as? SeatClassTableViewCell {

seatClassSelectionVC.mainTableView.visibleCells.forEach { (otherCell) in

if otherCell != selectedCell {

otherCell.translateAndFade(as: .transitionIn, animationStyle: animationStyle, percentageEndPoint: 0.6, translate: .zero)

}

}

let originRectInMainContentView = selectedCell.cardView.frame(in: seatMapSelectionVC.mainContentView)

let destinationRectInMainContentView = seatMapSelectionVC.mainCardView.frame(in: seatMapSelectionVC.mainContentView)

let destinationRectInOriginView = seatMapSelectionVC.mainCardView.frame(in: selectedCell.contentView)

if let cardViewInDestinationView = seatMapSelectionVC.mainCardView {

selectedCell.cardView.contentView.layer.removeAllAnimations()

selectedCell.cardView.layer.removeAllAnimations()

let dummyView = UIView()

let dummyViewInCell = UIView()

seatMapSelectionVC.mainContentView.addSubview(dummyView)

seatMapSelectionVC.mainContentView.sendSubviewToBack(dummyView)

selectedCell.contentView.addSubview(dummyViewInCell)

selectedCell.contentView.sendSubviewToBack(dummyViewInCell)

dummyView.alpha = 0

dummyViewInCell.alpha = 0

dummyView.layer.setLayer(seatMapSelectionVC.mainCardView.layer.layerStyle)

dummyView.layer.setShadow(seatMapSelectionVC.mainCardView.layer.shadowStyle)

dummyView.frame = destinationRectInMainContentView

dummyViewInCell.layer.setLayer(seatMapSelectionVC.mainCardView.layer.layerStyle)

dummyViewInCell.layer.setShadow(seatMapSelectionVC.mainCardView.layer.shadowStyle)

dummyViewInCell.frame = destinationRectInOriginView

selectedCell.cardView.layer.setLayer(LayerStyle())

selectedCell.cardView.contentView.backgroundColor = .clear

selectedCell.cardView.contentView.fadeAnimation(as: .transitionIn, animationStyle: animationStyle, percentageEndPoint: 0.5)

cardViewInDestinationView.layer.cornerRadius = dummyView.layer.cornerRadius

cardViewInDestinationView.contentView.layer.cornerRadius = dummyView.layer.cornerRadius

cardViewInDestinationView.alpha = 1

cardViewInDestinationView.frame = originRectInMainContentView

cardViewInDestinationView.contentView.frame = cardViewInDestinationView.bounds

cardViewInDestinationView.fadeAnimation(as: .transitionOut, animationStyle: animationStyle, percentageEndPoint: 0.3)

var muatatedAnimationStyle = animationStyle

muatatedAnimationStyle.duration = animationStyle.duration \* 0.3

UIView.animate(withStyle: muatatedAnimationStyle,

delay: 0.3 \* animationStyle.duration,

animations: {

dummyView.alpha = 1

}, completion: {

\_ in

UIView.animate(withStyle: muatatedAnimationStyle,

animations: {

dummyView.alpha = 0

})

})

// Animate subview in cardView.content of Destination

cardViewInDestinationView.contentView.subviews.forEach { (view) in

UIView.animate(withStyle: animationStyle, animations: {

view.transform.tx = originRectInMainContentView.midX - destinationRectInMainContentView.midX

view.transform.ty = originRectInMainContentView.midY - destinationRectInMainContentView.midY

})

}

UIView.animate(withStyle: animationStyle, animations: {

cardViewInDestinationView.frame = originRectInMainContentView

cardViewInDestinationView.contentView.frame = cardViewInDestinationView.bounds

cardViewInDestinationView.layer.cornerRadius = layerStyle.card.normal().cornerRadius

cardViewInDestinationView.contentView.layer.cornerRadius = layerStyle.card.normal().cornerRadius

dummyView.layer.cornerRadius = layerStyle.card.normal().cornerRadius

dummyViewInCell.layer.cornerRadius = layerStyle.card.normal().cornerRadius

dummyView.frame = originRectInMainContentView

dummyViewInCell.frame = selectedCell.cardView.frame

}, completion: {

\_ in

dummyView.removeFromSuperview()

dummyViewInCell.removeFromSuperview()

cardViewInDestinationView.setupTheme()

selectedCell.cardView.setupTheme()

seatClassSelectionVC.mainTableView.clipsToBounds = true

})

}

}

}

container.backgroundColor = toView.backgroundColor

toView.backgroundColor = toView.backgroundColor?.withAlphaComponent(0)

fromView.backgroundColor = fromView.backgroundColor?.withAlphaComponent(0)

// MARK: Perform Animation

UIView

.animate(withStyle: animationStyle,

animations: {

toView.backgroundColor = toView.backgroundColor?.withAlphaComponent(0.0001)

},

completion: {

finished in

fromView.backgroundColor = fromView.backgroundColor?.withAlphaComponent(1)

toView.backgroundColor = toView.backgroundColor?.withAlphaComponent(1)

transitionContext.completeTransition(!transitionContext.transitionWasCancelled)

})

}

}#Storyboard

<?xml version="1.0" encoding="UTF-8"?>

<document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="15505" targetRuntime="iOS.CocoaTouch" propertyAccessControl="none" useAutolayout="YES" useTraitCollections="YES" useSafeAreas="YES" colorMatched="YES">

<device id="retina6\_1" orientation="portrait" appearance="light"/>

<dependencies>

<deployment identifier="iOS"/>

<plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="15510"/>

<capability name="Safe area layout guides" minToolsVersion="9.0"/>

<capability name="documents saved in the Xcode 8 format" minToolsVersion="8.0"/>

</dependencies>

<customFonts key="customFonts">

<array key="Rubik-Medium.ttf">

<string>Rubik-Medium</string>

</array>

</customFonts>

<scenes>

<!--Sign Up View Controller-->

<scene sceneID="R4j-ZB-qw8">

<objects>

<viewController storyboardIdentifier="signupview" id="qkc-Xb-Hkw" userLabel="Sign Up View Controller" customClass="SignUpViewController" customModule="Shinkansen3DSeatBookingPrototype" customModuleProvider="target" sceneMemberID="viewController">

<view key="view" contentMode="scaleToFill" id="C6U-vd-tGW">

<rect key="frame" x="0.0" y="0.0" width="414" height="896"/>

<autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>

<subviews>

<view clipsSubviews="YES" alpha="0.5" contentMode="scaleToFill" restorationIdentifier="signupview" translatesAutoresizingMaskIntoConstraints="NO" id="qXJ-zi-JbV" userLabel="MaskView">

<rect key="frame" x="0.0" y="0.0" width="414" height="896"/>

<color key="backgroundColor" white="0.0" alpha="1" colorSpace="calibratedWhite"/>

</view>

<stackView opaque="NO" contentMode="scaleToFill" axis="vertical" spacing="30" translatesAutoresizingMaskIntoConstraints="NO" id="ZSp-Fx-lcR">

<rect key="frame" x="30" y="84" width="354" height="228"/>

<subviews>

<stackView opaque="NO" contentMode="scaleToFill" axis="vertical" spacing="20" translatesAutoresizingMaskIntoConstraints="NO" id="PpN-SQ-GZZ">

<rect key="frame" x="0.0" y="0.0" width="354" height="148"/>

<subviews>

<stackView opaque="NO" contentMode="scaleToFill" axis="vertical" alignment="top" spacing="10" translatesAutoresizingMaskIntoConstraints="NO" id="sf3-hT-ZPQ">

<rect key="frame" x="0.0" y="0.0" width="354" height="35"/>

<subviews>

<stackView opaque="NO" contentMode="scaleToFill" spacing="10" translatesAutoresizingMaskIntoConstraints="NO" id="Gmr-aN-TF6">

<rect key="frame" x="0.0" y="0.0" width="354" height="24"/>

<subviews>

<button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="WJN-Ic-wKb">

<rect key="frame" x="0.0" y="0.0" width="24" height="24"/>

<color key="tintColor" white="1" alpha="1" colorSpace="calibratedWhite"/>

<state key="normal" image="nametag"/>

</button>

<textField opaque="NO" clipsSubviews="YES" contentMode="scaleToFill" horizontalHuggingPriority="249" contentHorizontalAlignment="left" contentVerticalAlignment="center" placeholder="IDNUMBER" textAlignment="natural" minimumFontSize="15" translatesAutoresizingMaskIntoConstraints="NO" id="Az7-qF-ebx">

<rect key="frame" x="34" y="0.0" width="320" height="24"/>

<color key="textColor" red="1" green="1" blue="1" alpha="1" colorSpace="calibratedRGB"/>

<fontDescription key="fontDescription" name="Rubik-Medium" family="Rubik" pointSize="20"/>

<textInputTraits key="textInputTraits" autocorrectionType="no" spellCheckingType="no" returnKeyType="next"/>

<userDefinedRuntimeAttributes>

<userDefinedRuntimeAttribute type="color" keyPath="placeHolderColor">

<color key="value" white="1" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

</userDefinedRuntimeAttribute>

</userDefinedRuntimeAttributes>

</textField>

</subviews>

</stackView>

<view contentMode="scaleToFill" translatesAutoresizingMaskIntoConstraints="NO" id="25K-9t-o6U" userLabel="Divider">

<rect key="frame" x="0.0" y="34" width="354" height="1"/>

<color key="backgroundColor" white="1" alpha="1" colorSpace="calibratedWhite"/>

<constraints>

<constraint firstAttribute="height" constant="1" id="uGQ-Bh-lrG"/>

</constraints>

</view>

</subviews>

<constraints>

<constraint firstAttribute="trailing" secondItem="25K-9t-o6U" secondAttribute="trailing" id="PbO-rJ-gqT"/>

<constraint firstAttribute="trailing" secondItem="Gmr-aN-TF6" secondAttribute="trailing" id="zm1-a6-UXQ"/>

</constraints>

</stackView>

<stackView opaque="NO" contentMode="scaleToFill" axis="vertical" alignment="top" spacing="10" translatesAutoresizingMaskIntoConstraints="NO" id="T2A-C2-hRV">

<rect key="frame" x="0.0" y="55" width="354" height="35"/>

<subviews>

<stackView opaque="NO" contentMode="scaleToFill" spacing="10" translatesAutoresizingMaskIntoConstraints="NO" id="izD-BE-sDd">

<rect key="frame" x="0.0" y="0.0" width="354" height="24"/>

<subviews>

<button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="Cql-0B-Wkm">

<rect key="frame" x="0.0" y="0.0" width="24" height="24"/>

<color key="tintColor" white="1" alpha="1" colorSpace="calibratedWhite"/>

<state key="normal" image="email"/>

</button>

<textField opaque="NO" clipsSubviews="YES" contentMode="scaleToFill" horizontalHuggingPriority="249" contentHorizontalAlignment="left" contentVerticalAlignment="center" placeholder="EMAIL" textAlignment="natural" minimumFontSize="15" translatesAutoresizingMaskIntoConstraints="NO" id="OH2-0V-Pwp">

<rect key="frame" x="34" y="0.0" width="320" height="24"/>

<color key="textColor" white="1" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

<fontDescription key="fontDescription" name="Rubik-Medium" family="Rubik" pointSize="20"/>

<textInputTraits key="textInputTraits" autocorrectionType="no" spellCheckingType="no" returnKeyType="next"/>

<userDefinedRuntimeAttributes>

<userDefinedRuntimeAttribute type="color" keyPath="placeHolderColor">

<color key="value" red="0.84705882349999995" green="0.84705882349999995" blue="0.84705882349999995" alpha="1" colorSpace="calibratedRGB"/>

</userDefinedRuntimeAttribute>

</userDefinedRuntimeAttributes>

</textField>

</subviews>

</stackView>

<view contentMode="scaleToFill" translatesAutoresizingMaskIntoConstraints="NO" id="0yV-d1-cve" userLabel="Divider">

<rect key="frame" x="0.0" y="34" width="354" height="1"/>

<color key="backgroundColor" white="1" alpha="1" colorSpace="calibratedWhite"/>

<constraints>

<constraint firstAttribute="height" constant="1" id="oNS-N9-O8S"/>

</constraints>

</view>

</subviews>

<constraints>

<constraint firstAttribute="trailing" secondItem="0yV-d1-cve" secondAttribute="trailing" id="8xh-Br-Dou"/>

<constraint firstAttribute="trailing" secondItem="izD-BE-sDd" secondAttribute="trailing" id="rgm-Uh-CMc"/>

</constraints>

</stackView>

<stackView opaque="NO" contentMode="scaleToFill" axis="vertical" alignment="top" spacing="10" translatesAutoresizingMaskIntoConstraints="NO" id="fHd-QA-YMU">

<rect key="frame" x="0.0" y="110" width="354" height="38"/>

<subviews>

<stackView opaque="NO" contentMode="scaleToFill" spacing="14" translatesAutoresizingMaskIntoConstraints="NO" id="udZ-Zp-Wta">

<rect key="frame" x="0.0" y="0.0" width="354" height="27"/>

<subviews>

<button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="nr1-qo-Z0F">

<rect key="frame" x="0.0" y="0.0" width="20" height="27"/>

<color key="tintColor" white="1" alpha="1" colorSpace="calibratedWhite"/>

<state key="normal" image="lock"/>

</button>

<textField opaque="NO" clipsSubviews="YES" contentMode="scaleToFill" horizontalHuggingPriority="249" contentHorizontalAlignment="left" contentVerticalAlignment="center" placeholder="PASSWORD" textAlignment="natural" minimumFontSize="15" translatesAutoresizingMaskIntoConstraints="NO" id="LMx-bY-0qs">

<rect key="frame" x="34" y="0.0" width="320" height="27"/>

<color key="textColor" red="1" green="1" blue="1" alpha="1" colorSpace="calibratedRGB"/>

<fontDescription key="fontDescription" name="Rubik-Medium" family="Rubik" pointSize="20"/>

<textInputTraits key="textInputTraits" autocorrectionType="no" spellCheckingType="no" returnKeyType="done" secureTextEntry="YES"/>

<userDefinedRuntimeAttributes>

<userDefinedRuntimeAttribute type="color" keyPath="placeHolderColor">

<color key="value" red="0.84705882349999995" green="0.84705882349999995" blue="0.84705882349999995" alpha="1" colorSpace="calibratedRGB"/>

</userDefinedRuntimeAttribute>

</userDefinedRuntimeAttributes>

</textField>

</subviews>

</stackView>

<view contentMode="scaleToFill" translatesAutoresizingMaskIntoConstraints="NO" id="ocQ-hZ-hYq" userLabel="Divider">

<rect key="frame" x="0.0" y="37" width="354" height="1"/>

<color key="backgroundColor" white="1" alpha="1" colorSpace="calibratedWhite"/>

<constraints>

<constraint firstAttribute="height" constant="1" id="UNj-jr-cr1"/>

</constraints>

</view>

</subviews>

<constraints>

<constraint firstAttribute="trailing" secondItem="udZ-Zp-Wta" secondAttribute="trailing" id="cLw-Fk-g5s"/>

<constraint firstAttribute="trailing" secondItem="ocQ-hZ-hYq" secondAttribute="trailing" id="wjo-hN-BbS"/>

</constraints>

</stackView>

</subviews>

</stackView>

<button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="dZa-rN-OZH" customClass="LineButton" customModule="Shinkansen3DSeatBookingPrototype" customModuleProvider="target">

<rect key="frame" x="0.0" y="178" width="354" height="50"/>

<color key="backgroundColor" white="1" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

<constraints>

<constraint firstAttribute="height" constant="50" id="ZE0-rx-tLV"/>

</constraints>

<fontDescription key="fontDescription" name="Rubik-Medium" family="Rubik" pointSize="17"/>

<color key="tintColor" white="1" alpha="1" colorSpace="calibratedWhite"/>

<state key="normal" title="LOGIN">

<color key="titleColor" white="0.0" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

</state>

<userDefinedRuntimeAttributes>

<userDefinedRuntimeAttribute type="number" keyPath="cornerRadius">

<real key="value" value="25"/>

</userDefinedRuntimeAttribute>

</userDefinedRuntimeAttributes>

<connections>

<action selector="Userlogin:" destination="qkc-Xb-Hkw" eventType="touchUpInside" id="HXR-NL-TDC"/>

<segue destination="Ngb-Yu-4j1" kind="show" id="oN2-5W-BtL"/>

</connections>

</button>

</subviews>

<constraints>

<constraint firstAttribute="height" constant="228" id="q39-rX-1q2"/>

</constraints>

</stackView>

</subviews>

<color key="backgroundColor" white="0.33333333333333331" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

<constraints>

<constraint firstItem="ZSp-Fx-lcR" firstAttribute="top" secondItem="9bF-Um-Qc3" secondAttribute="top" constant="40" id="AhZ-EF-30b"/>

<constraint firstItem="qXJ-zi-JbV" firstAttribute="leading" secondItem="9bF-Um-Qc3" secondAttribute="leading" id="En9-oL-SHp"/>

<constraint firstAttribute="trailingMargin" secondItem="ZSp-Fx-lcR" secondAttribute="trailing" constant="10" id="SEH-MQ-32A"/>

<constraint firstAttribute="bottom" secondItem="qXJ-zi-JbV" secondAttribute="bottom" id="d2t-RN-vPn"/>

<constraint firstAttribute="bottom" secondItem="qXJ-zi-JbV" secondAttribute="bottom" id="fp3-S1-cWI"/>

<constraint firstItem="qXJ-zi-JbV" firstAttribute="top" secondItem="C6U-vd-tGW" secondAttribute="top" id="hBO-S5-enr"/>

<constraint firstItem="ZSp-Fx-lcR" firstAttribute="leading" secondItem="C6U-vd-tGW" secondAttribute="leadingMargin" constant="10" id="kcp-Cq-jI6"/>

<constraint firstItem="9bF-Um-Qc3" firstAttribute="trailing" secondItem="qXJ-zi-JbV" secondAttribute="trailing" id="mc2-pe-yuv"/>

</constraints>

<viewLayoutGuide key="safeArea" id="9bF-Um-Qc3"/>

</view>

<navigationItem key="navigationItem" title="Sign Up" id="c6B-NV-Gxy">

<barButtonItem key="rightBarButtonItem" systemItem="cancel" id="Inb-hU-w3x">

<color key="tintColor" red="1" green="1" blue="1" alpha="1" colorSpace="calibratedRGB"/>

<connections>

<segue destination="3G2-Rh-7NV" kind="unwind" unwindAction="unwindtoWelcomeViewWithSegue:" id="2hc-ME-wqr"/>

</connections>

</barButtonItem>

</navigationItem>

<connections>

<outlet property="EMAIL" destination="OH2-0V-Pwp" id="wZF-ng-y8v"/>

<outlet property="ID" destination="Az7-qF-ebx" id="LIx-c1-DH1"/>

<outlet property="PASSWORD" destination="LMx-bY-0qs" id="aho-sT-ckY"/>

<outlet property="login" destination="dZa-rN-OZH" id="5fn-K1-CK4"/>

</connections>

</viewController>

<placeholder placeholderIdentifier="IBFirstResponder" id="yOA-bU-P0c" userLabel="First Responder" sceneMemberID="firstResponder"/>

<exit id="3G2-Rh-7NV" userLabel="Exit" sceneMemberID="exit"/>

</objects>

<point key="canvasLocation" x="-249" y="-106"/>

</scene>

<!--Sign Up View Controller-->

<scene sceneID="8Wq-vh-qhC">

<objects>

<viewController storyboardIdentifier="selectview" id="Ngb-Yu-4j1" userLabel="Sign Up View Controller" customClass="SelectView" customModule="Shinkansen3DSeatBookingPrototype" customModuleProvider="target" sceneMemberID="viewController">

<view key="view" contentMode="scaleToFill" id="Kif-mm-xsS">

<rect key="frame" x="0.0" y="0.0" width="414" height="842"/>

<autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>

<subviews>

<view clipsSubviews="YES" contentMode="scaleToFill" restorationIdentifier="selectview" translatesAutoresizingMaskIntoConstraints="NO" id="Pl6-mR-HBl" userLabel="MaskView">

<rect key="frame" x="0.0" y="0.0" width="414" height="842"/>

<subviews>

<button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="5is-jN-OEs" customClass="LineButton" customModule="Shinkansen3DSeatBookingPrototype" customModuleProvider="target">

<rect key="frame" x="30" y="450" width="354" height="50"/>

<color key="backgroundColor" white="1" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

<constraints>

<constraint firstAttribute="height" constant="50" id="I14-jb-WpB"/>

</constraints>

<fontDescription key="fontDescription" name="Rubik-Medium" family="Rubik" pointSize="17"/>

<color key="tintColor" white="1" alpha="1" colorSpace="calibratedWhite"/>

<state key="normal" title="BOOKING">

<color key="titleColor" white="0.0" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

</state>

<userDefinedRuntimeAttributes>

<userDefinedRuntimeAttribute type="number" keyPath="cornerRadius">

<real key="value" value="25"/>

</userDefinedRuntimeAttribute>

</userDefinedRuntimeAttributes>

<connections>

<action selector="Booking:" destination="Ngb-Yu-4j1" eventType="touchUpInside" id="oZj-SD-zG6"/>

</connections>

</button>

<button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="middleTruncation" translatesAutoresizingMaskIntoConstraints="NO" id="CWr-Dx-q6j" customClass="LineButton" customModule="Shinkansen3DSeatBookingPrototype" customModuleProvider="target">

<rect key="frame" x="30" y="550" width="354" height="50"/>

<color key="backgroundColor" white="1" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

<constraints>

<constraint firstAttribute="height" constant="50" id="sr9-In-mtd"/>

</constraints>

<fontDescription key="fontDescription" name="Rubik-Medium" family="Rubik" pointSize="17"/>

<color key="tintColor" white="1" alpha="1" colorSpace="calibratedWhite"/>

<state key="normal" title="RECORD">

<color key="titleColor" white="0.0" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

</state>

<userDefinedRuntimeAttributes>

<userDefinedRuntimeAttribute type="number" keyPath="cornerRadius">

<real key="value" value="25"/>

</userDefinedRuntimeAttribute>

</userDefinedRuntimeAttributes>

<connections>

<action selector="Record:" destination="Ngb-Yu-4j1" eventType="touchUpInside" id="hgf-FT-tf3"/>

</connections>

</button>

<label opaque="NO" userInteractionEnabled="NO" contentMode="left" horizontalHuggingPriority="251" verticalHuggingPriority="251" text="RAILYWAY BOOKING" textAlignment="center" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO" translatesAutoresizingMaskIntoConstraints="NO" id="NhP-Ik-UBP">

<rect key="frame" x="20" y="160" width="374" height="36"/>

<fontDescription key="fontDescription" type="system" pointSize="30"/>

<color key="textColor" white="1" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

<nil key="highlightedColor"/>

</label>

</subviews>

<color key="backgroundColor" white="0.33333333333333331" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

<constraints>

<constraint firstItem="CWr-Dx-q6j" firstAttribute="top" secondItem="5is-jN-OEs" secondAttribute="bottom" constant="50" id="4ci-T7-h2k"/>

<constraint firstItem="5is-jN-OEs" firstAttribute="leading" secondItem="Pl6-mR-HBl" secondAttribute="leading" constant="30" id="7U6-Sv-vnm"/>

<constraint firstItem="NhP-Ik-UBP" firstAttribute="top" secondItem="Pl6-mR-HBl" secondAttribute="top" constant="160" id="JB8-IA-d6j"/>

<constraint firstItem="5is-jN-OEs" firstAttribute="top" secondItem="Pl6-mR-HBl" secondAttribute="top" constant="450" id="Wkz-5k-AUO"/>

<constraint firstAttribute="trailing" secondItem="5is-jN-OEs" secondAttribute="trailing" constant="30" id="Y4w-Xn-R2c"/>

<constraint firstAttribute="trailing" secondItem="NhP-Ik-UBP" secondAttribute="trailing" constant="20" id="buC-kv-CgE"/>

<constraint firstItem="NhP-Ik-UBP" firstAttribute="leading" secondItem="Pl6-mR-HBl" secondAttribute="leading" constant="20" id="iG6-Dg-M60"/>

<constraint firstAttribute="trailing" secondItem="CWr-Dx-q6j" secondAttribute="trailing" constant="30" id="p1K-bM-wbK"/>

<constraint firstItem="CWr-Dx-q6j" firstAttribute="leading" secondItem="Pl6-mR-HBl" secondAttribute="leading" constant="30" id="qnN-cG-Kkn"/>

</constraints>

</view>

</subviews>

<color key="backgroundColor" white="0.33333333333333331" alpha="1" colorSpace="custom" customColorSpace="genericGamma22GrayColorSpace"/>

<constraints>

<constraint firstItem="Blk-5B-QMb" firstAttribute="trailing" secondItem="Pl6-mR-HBl" secondAttribute="trailing" id="7g7-s9-k1k"/>

<constraint firstAttribute="bottom" secondItem="Pl6-mR-HBl" secondAttribute="bottom" id="7jf-Gm-jAY"/>

<constraint firstItem="Pl6-mR-HBl" firstAttribute="top" secondItem="Kif-mm-xsS" secondAttribute="top" id="NNO-Fl-ceY"/>

<constraint firstItem="Pl6-mR-HBl" firstAttribute="leading" secondItem="Blk-5B-QMb" secondAttribute="leading" id="hVU-1M-W7a"/>

<constraint firstAttribute="bottom" secondItem="Pl6-mR-HBl" secondAttribute="bottom" id="yPu-uc-yNM"/>

</constraints>

<viewLayoutGuide key="safeArea" id="Blk-5B-QMb"/>

</view>

<navigationItem key="navigationItem" title="Sign Up" id="DnJ-cI-xrN">

<barButtonItem key="rightBarButtonItem" systemItem="cancel" id="hYR-5x-HXZ">

<color key="tintColor" red="1" green="1" blue="1" alpha="1" colorSpace="calibratedRGB"/>

<connections>

<segue destination="CWx-09-TXT" kind="unwind" unwindAction="unwindtoWelcomeViewWithSegue:" id="Mzx-p0-jbo"/>

</connections>

</barButtonItem>

</navigationItem>

</viewController>

<placeholder placeholderIdentifier="IBFirstResponder" id="rHn-aP-C5i" userLabel="First Responder" sceneMemberID="firstResponder"/>

<exit id="CWx-09-TXT" userLabel="Exit" sceneMemberID="exit"/>

</objects>

<point key="canvasLocation" x="458" y="-106"/>

</scene>

</scenes>

<resources>

<image name="email" width="24" height="16"/>

<image name="lock" width="20" height="27"/>

<image name="nametag" width="24" height="20"/>

</resources>

</document>