

Team10

D0617735 霍湛軒

D0616688 馮潤輝

1. 內容 (what- function)

這是一個員工在遠距離上班時所需要的App，包括打卡、行事歷、工資查詢的功能。

2. 設計 (how- design)

在疫情影響下，大部分的公司都會安排員工在家工作，從而減少人群聚集並降低感染的機會。因此需要做出一個具有打卡功能來記錄員工的工作時數，並可以瀏覽目前時數對應的工資，另外行事曆可以讓員工自己增加綫上開會的時間提醒開會。

3. 程式模組說明 (how- code)

打卡：

利用stopwath去記錄員工的打上班時間，用結束的時間減去開始的時間，並把員工一天的上班時間記錄到資料庫。

行事曆：

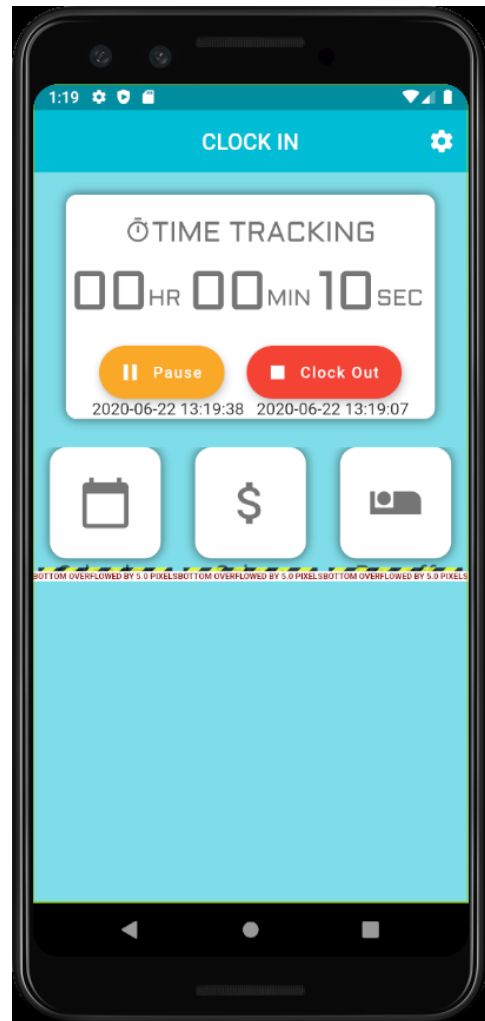
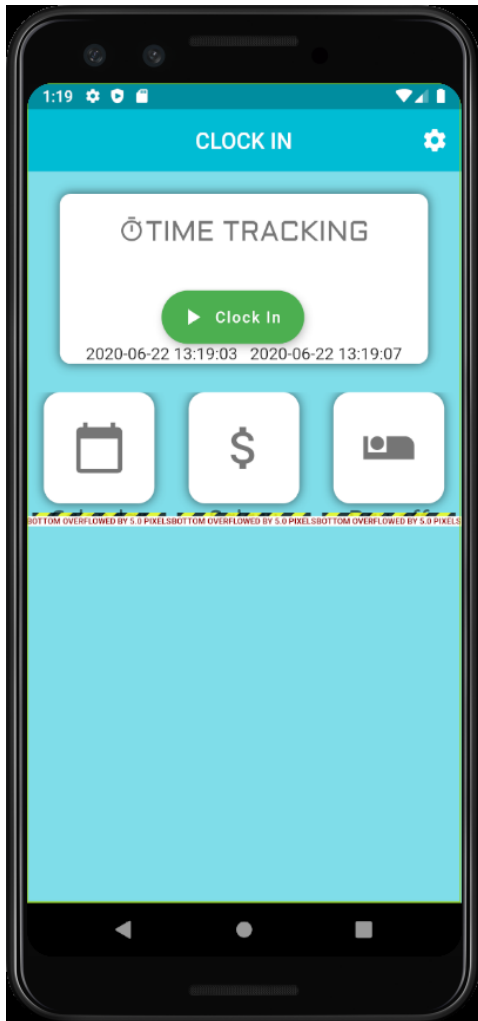
讓使用者可以新增和查詢開會事件。當中包含標題，開會時間和詳細。

工資查詢：

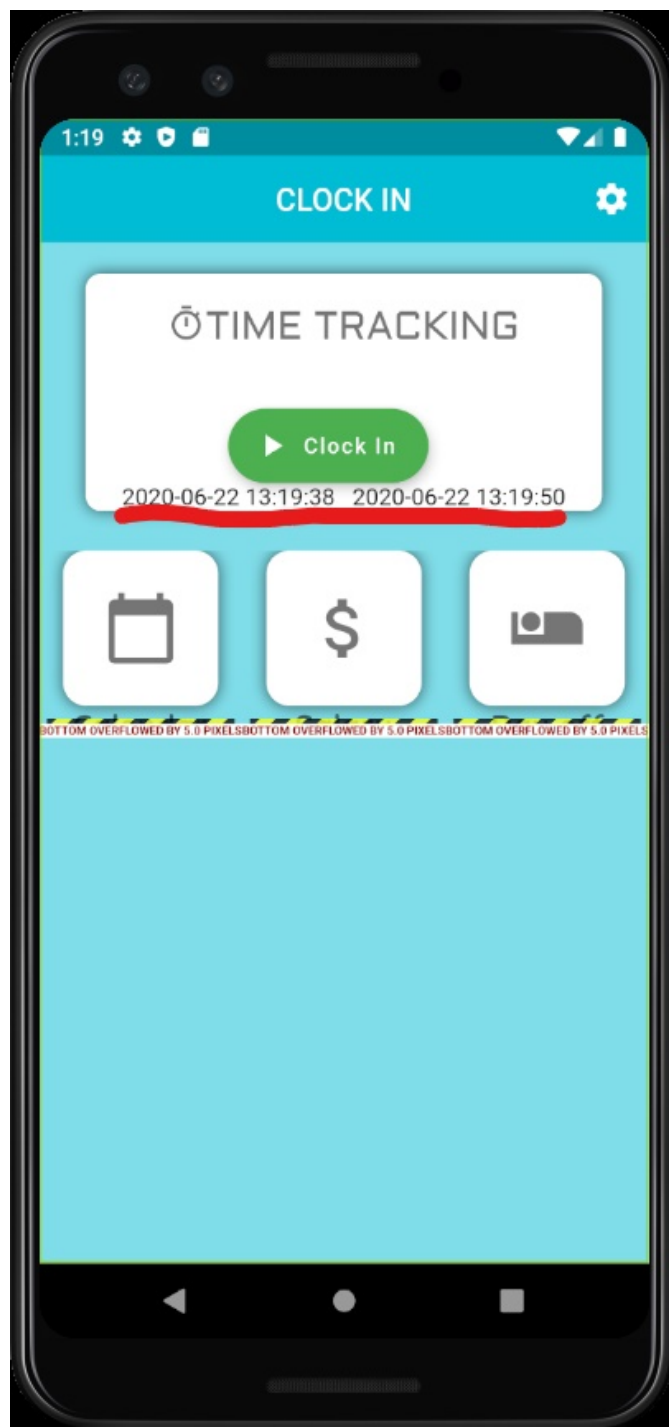
利用打卡的時間算出的工作時數並記錄下來，並將總時數乘上時薪。如果總時數小數點的部分小於0.5則向下取。反之向上取。

4. 成果 (what- result), 包含畫面

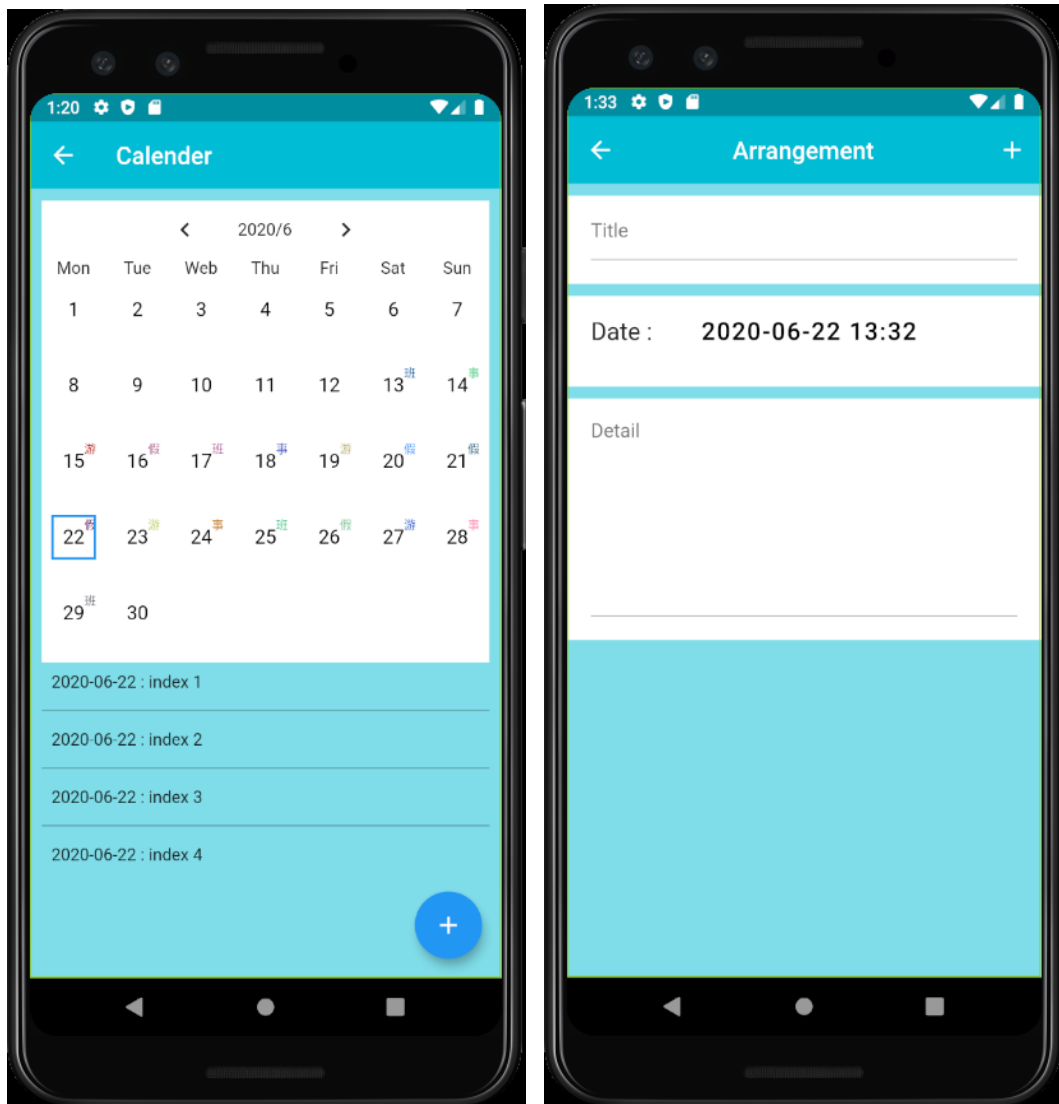
打卡功能：Clock In 則是代表開始打卡時間，Clock Out 則是結束打卡時間。



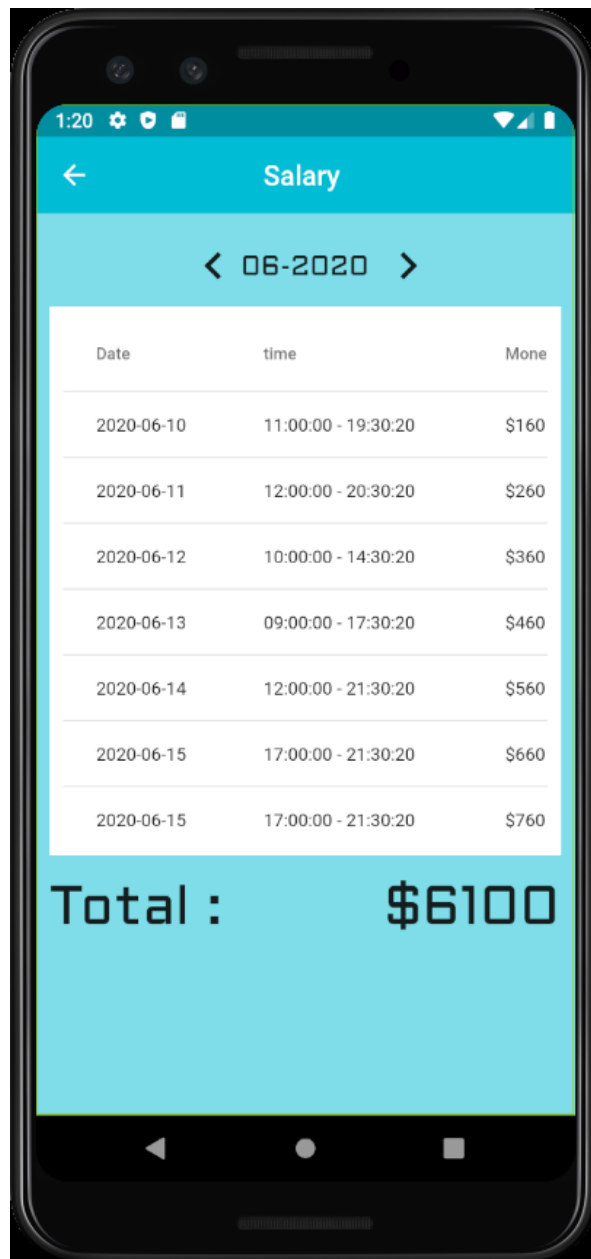
完成打卡的時間會顯示在下面。打卡時間會利用 calTime 計算工時並加進資料庫。



行事曆：左圖為行事曆，使用者可以在下方的 List 查看最近的事件。如果需要增加事件的話可以按右下角的 “+” 進入新增事件功能界面，如右圖所示。



工資查詢：上方可以選擇 年月 ，並在下面的 ListView 顯示的打卡時間資料，並利用資料庫的資料呼叫 calSalary 來計算總工資。



5. 測試報告 (test)

1. 測試方法

非獨立邊界測試，涵蓋度測試，強邊界測試，mockito,

2. 測試設計與案例說明

以下為計算時數與工資的測試案列。calTime 測試採用非獨立邊界測試，由於輸入的時間是2個不同的時間，並計算出2個之間的時間即為工時。所以可直接呼叫函式計算。分別有3種情況：

- 工時為整數(case 1~4)
- 工時為小數(case 5~9)
- 輸入時間不對(Case 10)則會回傳 0 小時

Part Time 的calPTSalary 則是利用 資料庫的總工時來進行工資運算，但由於資料庫尚未實做完成，就先利用假資料來測試。那麼為了預防輸入的時間不對而進行防呆處理。在總工時的部分會以0.5作為區間，如果總工時的浮點數部分有做四捨五入。並回傳工資。測試案列則為2種：

- normal資料(case 1)
- 輸入格式錯誤，需要回傳0的資料 (case 2)

calTimeTest									
calTime									
case	Start DateTime	End DateTime					output(hours)		
1	20/6/2020 11:00	20/6/2020 20:00					3		
2	20/6/2020 1:00	20/6/2020 1:00					9		
3	20/6/2020 1:00	20/6/2020 20:00					19		
4	20/6/2020 18:00	21/6/2020 19:00					25		
5	20/6/2020 17:00	20/6/2020 20:15					3.25		
6	20/6/2020 17:00	20/6/2020 20:30					3.5		
7	20/6/2020 17:00	20/6/2020 20:45					3.75		
8	20/6/2020 17:00	20/6/2020 20:20					3.33		
9	20/6/2020 17:00	20/6/2020 20:40					3.67		
10	20/6/2020 17:00	20/6/2020 15:00					0		
calSalary(PT)									
	hours					Total hours	salary	output	
1	3.75	8.75	9.5	6		28	158	4424	
2	8.75	0	8	8.67	5.5	30.92	180	5580	
calSalary(FT)									
	Date		salary	output					
1	1/2/2020 21:00	min-	25000	0					
2	14/2/2020 21:00	norm-	28000	0					
3	15/1/2020 10:00	min	30000	30000					
4	30/3/2020 21:00	norm	35000	35000					
5	31/5/2020 21:00	max-	63000	63000					

Full Time 的 calFTSalary 則是藉由 Mockito 來替代尚未實作的資料庫中的月薪。而在月薪的部分，由於我們的 APP 是顯示該月的員工工資。所以需要假設發薪日是在 15 號，因此我們需要針對這個去設計測試案例，因此有了以上的測試案例並使用了非獨立邊界測試案例。如果在 15 號之前則無工資，反之有資料庫所登記的工資並輸出。以下為 Mockito 的實作圖。

```
group('OT test', () {
    CalTime cal;
    MockCalTime server;

    setUp(() {
        server = new MockCalTime();
        cal = new CalTime();
        cal.setServer(server);
    });

    test('Is Full Time', () {
        var StartTime = DateTime.parse("2020-02-01 21:00:00");
        when(server.setFTSalary()).thenReturn(25000);
        expect(cal.calFTSalary(StartTime), 0);

        StartTime = DateTime.parse("2020-02-14 21:00:00");
        when(server.setFTSalary()).thenReturn(25000);
        expect(cal.calFTSalary(StartTime), 0);

        StartTime = DateTime.parse("2020-01-15 10:00:00");
        when(server.setFTSalary()).thenReturn(30000);
        expect(cal.calFTSalary(StartTime), 30000);

        StartTime = DateTime.parse("2020-03-30 21:00:00");
        when(server.setFTSalary()).thenReturn(35000);
        expect(cal.calFTSalary(StartTime), 35000);

        StartTime = DateTime.parse("2020-05-31 21:00:00");
        when(server.setFTSalary()).thenReturn(63000);
        expect(cal.calFTSalary(StartTime), 63000);

        verify(server.setFTSalary()).called(5);
    });
});
```

transformMilliseconds

把微秒的時間轉換成HH：MM：SS的格式。分別有時、分、秒的等價分割。

時： <0 、 $0 \leq X \leq 24$ 、 >24

分： <0 、 $0 \leq X \leq 60$ 、 >60

秒： <0 、 $0 \leq X \leq 60$ 、 >60

transformMilliSeconds	MilliSecond	Output
1	-1	00:00:00
2	0	00:00:00
3	1	00:00:00
4	1000	00:00:01
5	59999	00:00:59
6	60000	00:01:00
7	61000	00:01:01
8	120000	00:02:00
9	3599999	00:59:59
10	3600000	01:00:00
11	3601000	01:00:01
12	7200000	02:00:00
13	86399999	23:59:59
14	86400000	24:00:00
15	86401000	24:00:01

isOT

用來判斷員工是否有超時工作，首先要到資料庫中找到及統計員工一個月的工作時數，但因資料庫尚未實做完成所以先用Mockito代替計算員工時數的function。

```

group('OT test', () {
  HomePage homepage;
  MockServer server;
  setUp(() {
    server = new MockServer();
    homepage = new HomePage();
    homepage.setServer(server);
  });








  test('OT function test', () {
    when(server.getTotalHoursOfThisMonth()).thenReturn(300);
    expect(homepage.isOT(), true);
    when(server.getTotalHoursOfThisMonth()).thenReturn(200);
    expect(homepage.isOT(), false);
    verify(server.getTotalHoursOfThisMonth()).called(2);
  });
});

```

3. 測試結果

Widget test

Widget test 是Flutter所提供的測試工具，可以用程式的方式去測試UI元件在事件之間的正確性，而產生UI的程式碼亦被納入涵蓋度中。

Filename	Line Coverage ↕		Functions ↕	
CalTime.dart		100.0 %	16 / 16	- 0 / 0
arrangement.dart		95.8 %	68 / 71	- 0 / 0
calendar.dart		58.9 %	66 / 112	- 0 / 0
homepage.dart		76.6 %	151 / 197	- 0 / 0
leave.dart		100.0 %	13 / 13	- 0 / 0
main.dart		88.9 %	8 / 9	- 0 / 0
salary.dart		100.0 %	67 / 67	- 0 / 0

6. 自評

1. 功能的完整性 (485行)

功能的完整性不是很完整，因為這個APP並不是我們自己開發的，所以我們只能盡量把APP的功能加入軟體測試，把想到的功能並加以實作和套用進去，所以導致功能和測試感覺扯不上關係。（馮潤輝）

2. 測試的完整性 (包含測試案例數，涵蓋度等量化指標)

在這次測試，我的部分只有使用非獨立邊界測試，所以較為簡單。只需要針對時間去做測試案例，那我設置了16個測試案例，分別是 工時為整數，工時為浮點數，工時為0或小於0 就回傳 0。另外在calSalary 的測試則是設置了2個測試案例，分別是 normal 和 有小數的情況，在有小數的情況中則會四捨五入。那麼我的測試案例涵蓋度是100%，因為功能較為簡單，因此涵蓋度較高。（馮潤輝）

在這次測試中我負責了transformMilliseconds及isOT這兩個比較簡單的功能。在transformMilliseconds中我使用了強邊邊界測及等價分割的方式去測試功能的正確性及完整性；而isOT只是一個簡單的判斷，但這個功能的重點是從資料庫中得到員工這個月的總時數，但因資料庫還沒實做出來，因此用Mockito去模擬資料庫的回傳。

而Widget test 則是利用Flutter去觸發每個UI，而Widget test的結果非常完整除了一些非官方的UI套件因結構複雜未能全部測試。（霍湛軒）

3. 測試的自動化程度（386行）

在這次的測試中，我們只需要設計好測試案例並整理好，然後由Flutter所提供的測試工具即可測試test case及coverage test，而測試後可從Html中觀看介面化的數據及結果。

SonarQupe

