

A woman with dark hair, wearing a blue button-down shirt, is looking upwards and to the right with a slight smile. She is positioned on the right side of the frame. Overlaid on the left side is a large, stylized cloud shape filled with numerous small, circular icons representing various business and technology concepts like a car, camera, handshake, globe, and clock. In the bottom left corner, the SAP logo is visible. The background is a blurred city street scene with yellow Chinese characters on a building facade.

Priyanka S N
SEP 18, 2015



Agenda

- Packages
- Time drawer
- Meal Break
- Time Tracking Manual

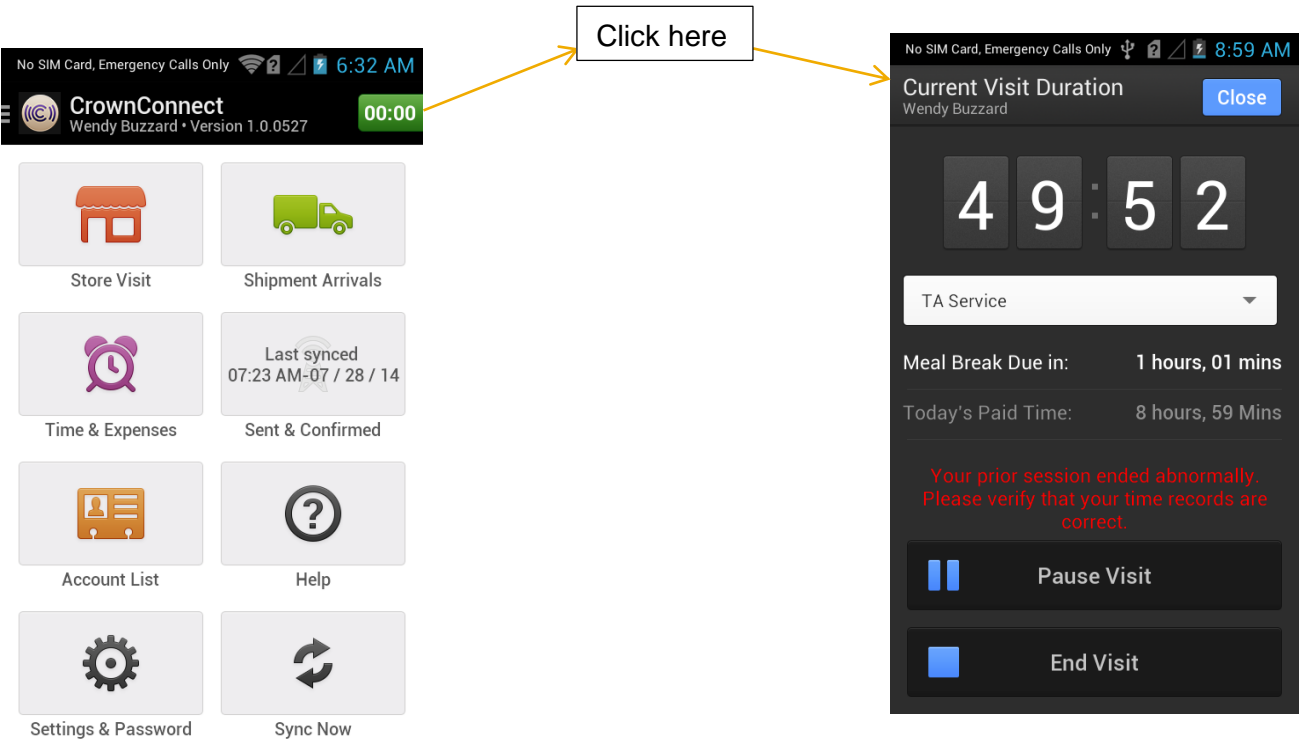
Packages

The following packages are created to support this functionality:

- `com.sap.hallmark.views`
- `com.sap.centerlock.horizontalscrollview`
- `com.sap.hallmark.adapters`
- `com.sap.hallmark.abo`
- `com.mdc.sap.fso.mbo`
- `com.sap.hallmark.framework`

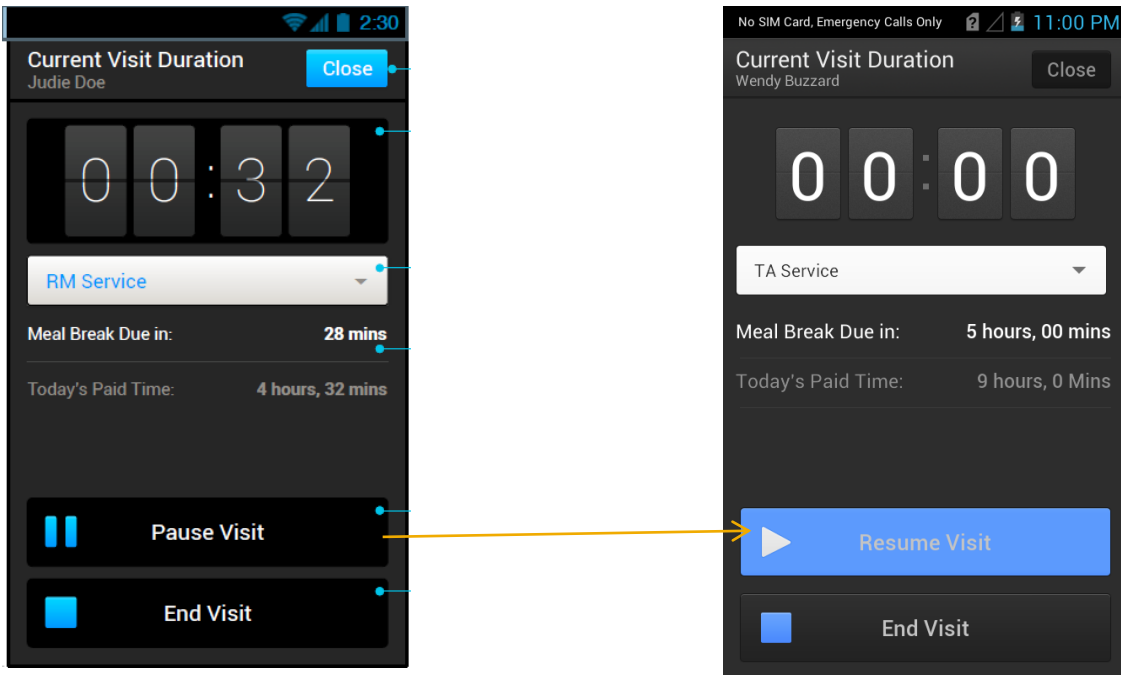
Time Drawer

If the user clicks on the green color header icon (header timer icon), then the application opens the TimeDrawer screen,



Time Drawer

If the user clicks on the pause button then all the running timers are paused on that instance and the screen UI will be as seen below :



Time Drawer

Pause Code (TimeDrawerPresenter.java)

```
public void pauseSystemTimer() {  
  
    pauseSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_HEADER_TIMER_OBJECT);  
        pauseSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_ADMIN_TIME);  
  
    pauseSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_INSTALLATION_SERVICE_TIMER_OBJ  
ECT);  
  
    pauseSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_MAA_SERVICE_TIMER_OBJECT);  
  
    pauseSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_RM_SERVICE_TIMER_OBJECT);  
  
    pauseSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_TA_SERVICE_TIMER_OBJECT);  
  
    pauseSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_MEALBREAK_TIME);  
        startPauseTimer();  
}
```

Time Drawer

```
private void pauseSystemTimer(String key) {  
    TimeCounter timeCounter = TimerUtility.getTimerUtilityInstance()  
                                           .getTimerObjectFromContainer(key);  
    if (timeCounter != null) {  
        timeCounter.pauseSystemTimer();  
        //timeCounter.unregisterTimerCounterCallback();  
    }  
}
```

Time Drawer

Resume Code (TimeDrawerPresenter.java):

```
public void resumeSystemTimer() {  
    resumeSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_HEADER_TIMER_OBJECT);  
  
    resumeSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_ADMIN_TIMER_OBJECT);  
  
    resumeSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_MEALBREAK_TIMER_OBJECT);  
  
    resumeSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_INSTALLATION_SERVICE_TIMER_OBJECT);  
  
    resumeSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_MAA_SERVICE_TIMER_OBJECT);  
  
    resumeSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_RM_SERVICE_TIMER_OBJECT);  
  
    resumeSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_TA_SERVICE_TIMER_OBJECT);  
}
```


Time Drawer

```
        TimeCounter pauseTimeCounter =
TimerUtility.getTimerUtilityInstance().getTimerObjectFromContainer(IAppConstants.TimerMa
nagerConstants.TIMER_UTILITY_PAUSE_ACTIVITY_TIMER_OBJECT);
        if(pauseTimeCounter != null) {
            long pauseTime =
pauseTimeCounter.getTimeLapsed();
            if(pauseTime < 60) {
                //flush pause timer - no affect

                addPauseSecondsToTimer(IAppConstants.TimerManagerConstants.TIMER_UTI
LITY_HEADER_TIMER_OBJECT, pauseTime);

                addPauseSecondsToTimer(IAppConstants.TimerManagerConstants.TIMER_UTI
LITY_ADMIN_TIME, pauseTime);

                addPauseSecondsToTimer(IAppConstants.TimerManagerConstants.TIMER_UTI
LITY_MEALBREAK_TIME, pauseTime);

                addPauseSecondsToTimer(IAppConstants.TimerManagerConstants.TIMER_UTI
LITY INSTALLATION SERVICE TIMER OBJECT, pauseTime);
```

Time Drawer

```
addPauseSecondsToTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_MAA_SERVICE_TIMER_OBJECT, pauseTime);

        addPauseSecondsToTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_RM_SERVICE_TIMER_OBJECT, pauseTime);

        addPauseSecondsToTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_TA_SERVICE_TIMER_OBJECT, pauseTime);

        pauseTimeCounter.stopSystemTimer();

        pauseTimeCounter.flushSystemTimer();

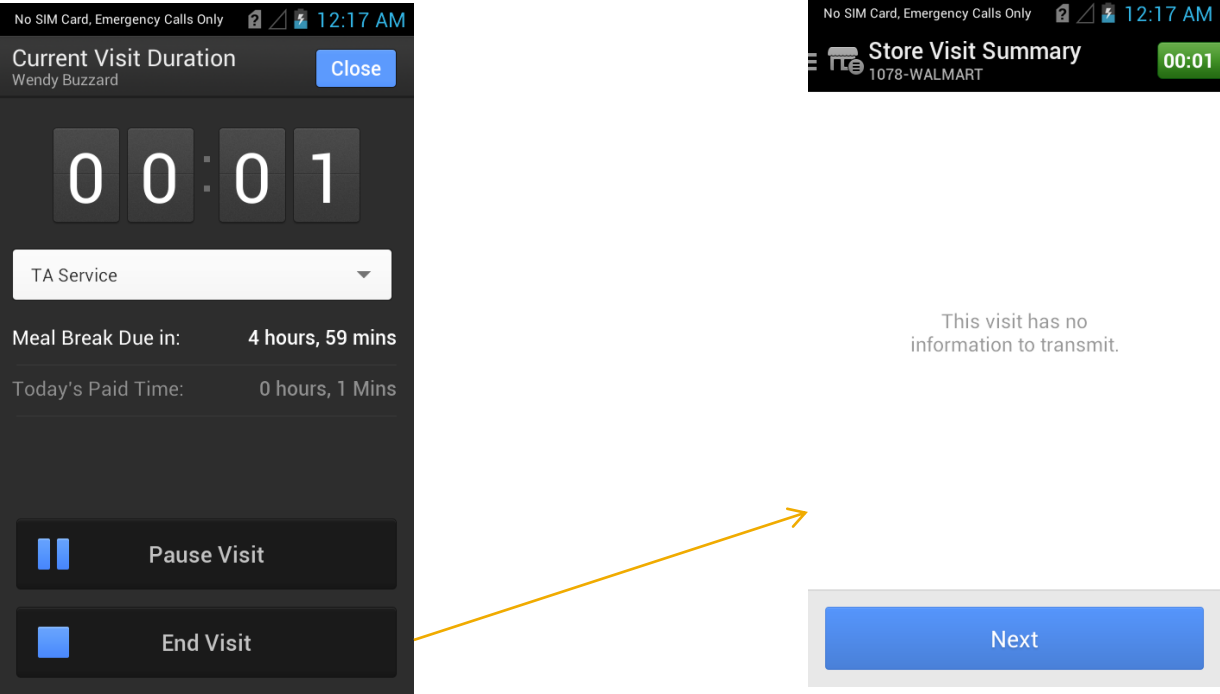
        TimerUtility.getTimerUtilityInstance().removeTimerObjectFromContainer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_PAUSE_ACTIVITY_TIMER_OBJECT);
    }
```

Time Drawer

```
else if(pauseTime >= 60 && pauseTime < 30*60) {  
  
    // save pause timer to DB  
    TimestampUtility.getTimeStampInstance(mContext).submitPauseTimeStamp();  
    pauseTimeCounter.stopSystemTimer();  
    pauseTimeCounter.flushSystemTimer();  
  
    TimerUtility.getTimerUtilityInstance().removeTimerObjectFromContainer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_PAUSE_ACTIVITY_TIMER  
_OBJECT);  
  
    }else if(pauseTime >= 30*60) {  
        //save pause timer to DB and reset meal break timer to 0.  
        TimestampUtility.getTimeStampInstance(mContext).submitPauseTimeStamp();  
        Timestamp endTimeStamp =  
  
        AppUtility.getAppUtilityInstance().getCurrentTimeStampWithoutSecs();  
  
        resetSystemTimer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_MEALBREAK_TIME,  
true);  
  
        pauseTimeCounter.stopSystemTimer();  
        pauseTimeCounter.flushSystemTimer();  
  
        TimerUtility.getTimerUtilityInstance().removeTimerObjectFromContainer(IAppConstants.TimerManagerConstants.TIMER_UTILITY_PAUSE_ACTIVITY_TIMER  
_OBJECT);  
  
        }  
        //Delete all local pause records.  
        WriteLocalDataUtility localDataUtil = new WriteLocalDataUtility(mContext);  
        localDataUtil.deleteAllPauseRecords();  
    }  
}
```

Time Drawer

If the user enters into the store and clicks on the end visit button.



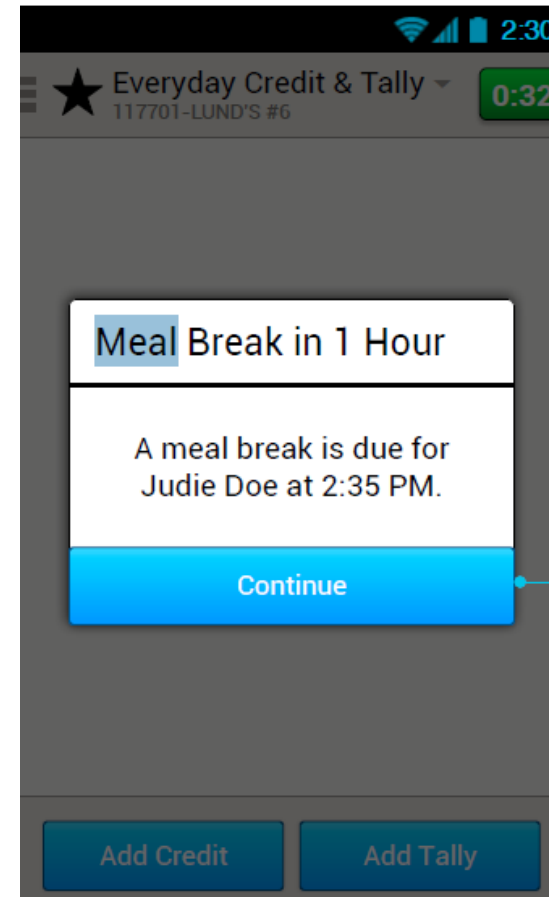
Meal Break

The employee is required to take a meal break for every five hours worked. Each meal break has two warnings.

The first warning is given an hour before the required meal break with the title "Meal Break in 1 Hour".

The second warning is given fifteen minutes before the required meal break with the title "Meal Break in 15 Mins".

The backend needs to record that the warnings are shown and that the employee acknowledges them via tapping Continue.



Meal Break

Meal break due in time will reset based on the below conditions:

1 – If there is a timer pause 30mins or more than 30 mins.

2 - After every 5 hours.

3 – At midnight.

Time Tracking

The user is able to choose between Work, Travel and Administrative time reports.

Account details are automatically populated with the current checked in account. If the user is not checked in, only the "Choose Account" button is present

Optional work categories values comes from the OptionalWorkCategory MBO.

The screenshot shows a mobile application interface for adding work time. At the top, there's a status bar with the time 2:30. Below it, a header bar contains the text 'Add Work Time' and two buttons: 'Cancel' and 'Add'. Under the header is a tab bar with three tabs: 'STORE' (which is selected and highlighted in blue), 'TRAVEL', and 'ADMIN'. The main content area is a form with several sections. The first section is 'Type of Store Work' with a blue dropdown menu currently showing 'RM Service'. Below this is a section with two input fields: 'Start Time' showing '09:00 AM' and 'Duration' showing '50 Mins'. The next section is 'Date' showing 'Today, 06 / 03 / 13'. Below that is 'Optional Work Category' with a blue dropdown menu showing 'None'. The final section is 'Split Shift' with a toggle switch currently set to 'No'. At the very bottom of the form is a section labeled 'Account Details'.

Time Tracking

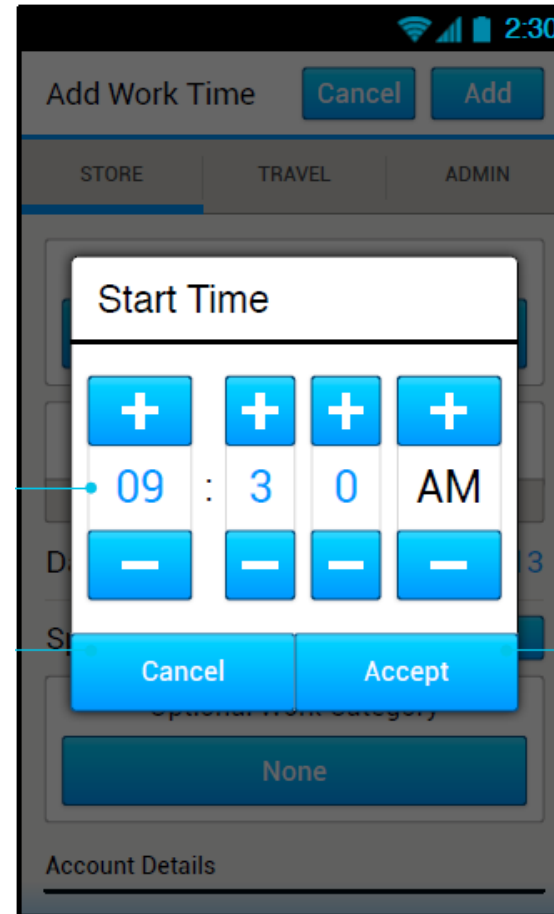
If the user chooses Installation as the type of store work, we reveal a second option for them to choose an installation event.

The screenshot shows a mobile application interface for 'Add Work Time'. At the top, there are status icons (Wi-Fi, signal, battery) and the time '2:30'. Below the title bar, there are 'Cancel' and 'Add' buttons. A tabbed interface shows 'STORE', 'TRAVEL', and 'ADMIN' tabs, with 'STORE' being the active tab. The main content area is titled 'Type of Store Work' and contains a dropdown menu with 'Installation' selected. Below this is a button labeled 'Choose Installation Event'. Further down, there are two input fields: 'Start Time' with the value '09:00 AM' and 'Duration' with the value '50 Mins'. Below these is a 'Date' field with the value 'Today, 06 / 03 / 13'. At the bottom, there is a section titled 'Optional Work Category' with a dropdown menu showing 'None'.

Time Tracking

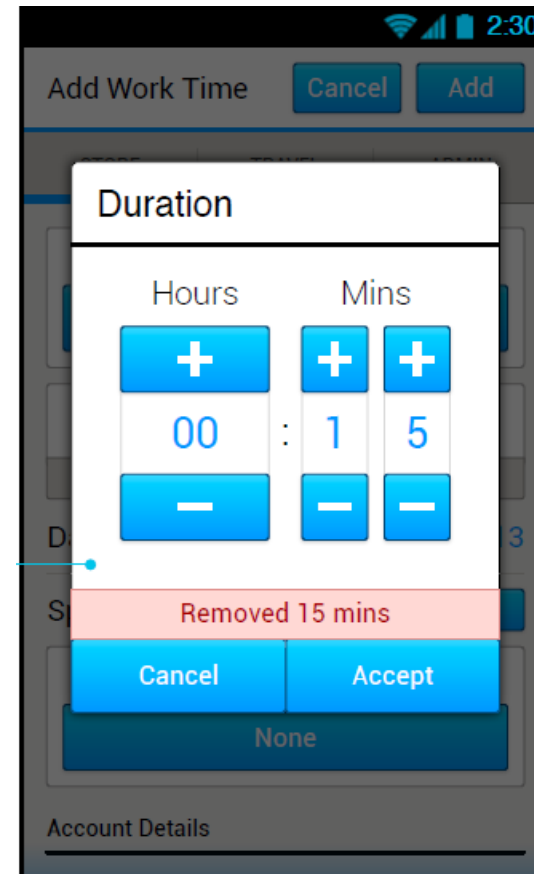
Midnight (12AM) is the baseline start time for any day.

Users cannot enter time that is after the current time.
(i.e. They cannot add time for 4PM that day when it is 2:30PM.)



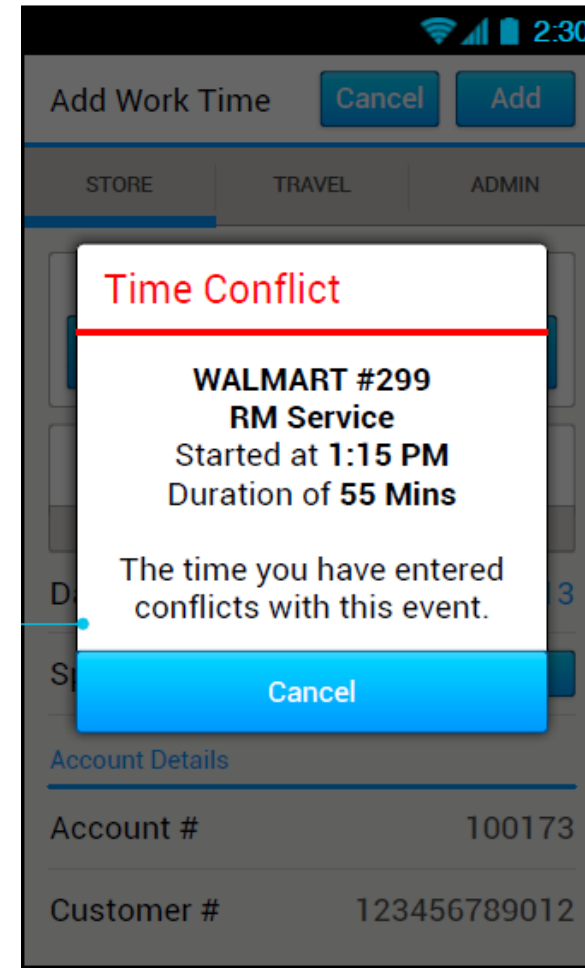
Time Tracking

If the user opens a time picker after initial entry an additional row of information is displayed to show the relative difference between the entered time and modified time; with "Removed" for time removed from the duration and "Added" for time added to the duration. If no change has been made yet, it instead reads "No changes made".



Time Tracking

Multiple alerts for Time Tracking must be supported.
Hallmark will provide a list of triggers and resulting messages.



Time Tracking

Type of Travel is not defaulted to either option, the user must select one of the choices before adding the travel time.

The screenshot shows a mobile application interface for adding travel time. At the top, there's a status bar with signal, battery, and time (2:30). Below it, a header bar contains the text 'Add Travel Time' and two buttons: 'Cancel' and 'Add'. A tab bar below the header has three tabs: 'STORE', 'TRAVEL' (which is selected and highlighted with a blue underline), and 'ADMIN'. The main content area is divided into several sections. The first section is titled 'Type of Travel' and contains two buttons: 'Commute' (highlighted in dark blue) and 'Account to Account' (highlighted in light blue). Below this, there are two input fields: 'Start Time' with the value '09:00 AM' and 'Duration' with the value '50 Mins'. The next section is 'Miles Traveled' with the value '40'. Below that is the 'Date' field with the value 'Today, 06 / 03 / 13'. A summary box states: 'You will be reimbursed for 25 miles and 30 minutes.' The bottom section is titled 'Account Details' and contains the 'Account #' field with the value '100173'.

Type of Travel	
Commute	Account to Account

Start Time	Duration
09:00 AM	50 Mins

Miles Traveled	
40	

Date	
Today, 06 / 03 / 13	

You will be reimbursed for 25 miles and 30 minutes.

Account Details	
Account #	100173

Time Tracking

In this screen the user has selected Account to Account as their Type of Travel.

The wording for reimbursement changes if the user is eligible for neither.

2:30

Add Travel Time Cancel Add

STORE TRAVEL ADMIN

Type of Travel

Commute Account to Account

09:00 AM 30 Mins

Start Time Duration

Miles Traveled 8

Date Today, 06 / 03 / 13

You will be reimbursed for 8 miles and 30 minutes.

Account Details

Account # 100173

Time Tracking

Districts values comes from the CostCenter MBO. Up to twelve options need to be supported. District will default to the user's current district.

The screenshot shows a mobile application interface for adding admin time. At the top, there is a status bar with a Wi-Fi icon, signal strength bars, battery level, and the time 2:30. Below the status bar is a header bar with the text "Add Admin Time" and two buttons: "Cancel" and "Add". Under the header bar is a tab bar with three tabs: "STORE", "TRAVEL", and "ADMIN". The "ADMIN" tab is selected. Below the tab bar is a form with the following fields:

- Type of Admin Work:** A dropdown menu with "RM Service" selected.
- Start Time:** A text field with "09:00 AM" entered.
- Duration:** A text field with "5 Mins" entered.
- Day:** A text field with "Today, 6 / 3 / 13" entered.
- District:** A dropdown menu with "NAT Southeast" selected.

Time Tracking

When editing an existing time report, the user is presented with the option of deleting the entry.

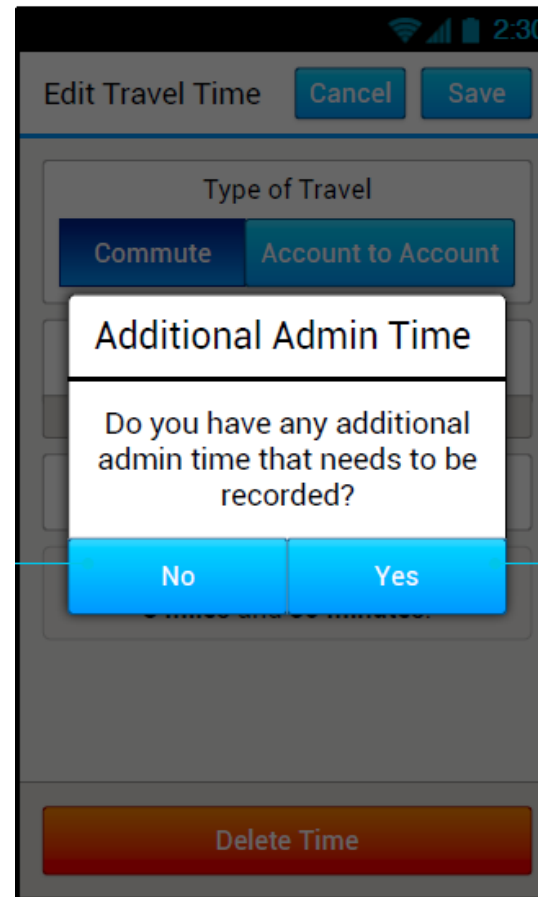
The screenshot shows a mobile application interface for editing work time. At the top, there is a status bar with a Wi-Fi icon, signal strength bars, battery level, and the time 2:30. Below the status bar, the title "Edit Work Time" is displayed, followed by "Cancel" and "Save" buttons. The main form contains the following elements:

- Type of Store Work:** A dropdown menu currently showing "RM Service".
- Start Time:** A field displaying "09:00 AM".
- Duration:** A field displaying "50 Mins".
- Add Unpaid Time:** A large blue button.
- Split Shift:** A toggle switch currently set to "No".
- Account Details:** A section header.
- Account #:** A field displaying "100173".
- Customer #:** A field displaying "123456789012".
- Delete Time:** A large orange button at the bottom.

Time Tracking

When the user has entered the second commute travel time for the day, a dialog requesting if they have any additional admin time to report is shown.

The API **isAdminPopOverRequired** of `TravelTimeABO` class is used to find if the second commute is being captured.



Thank You