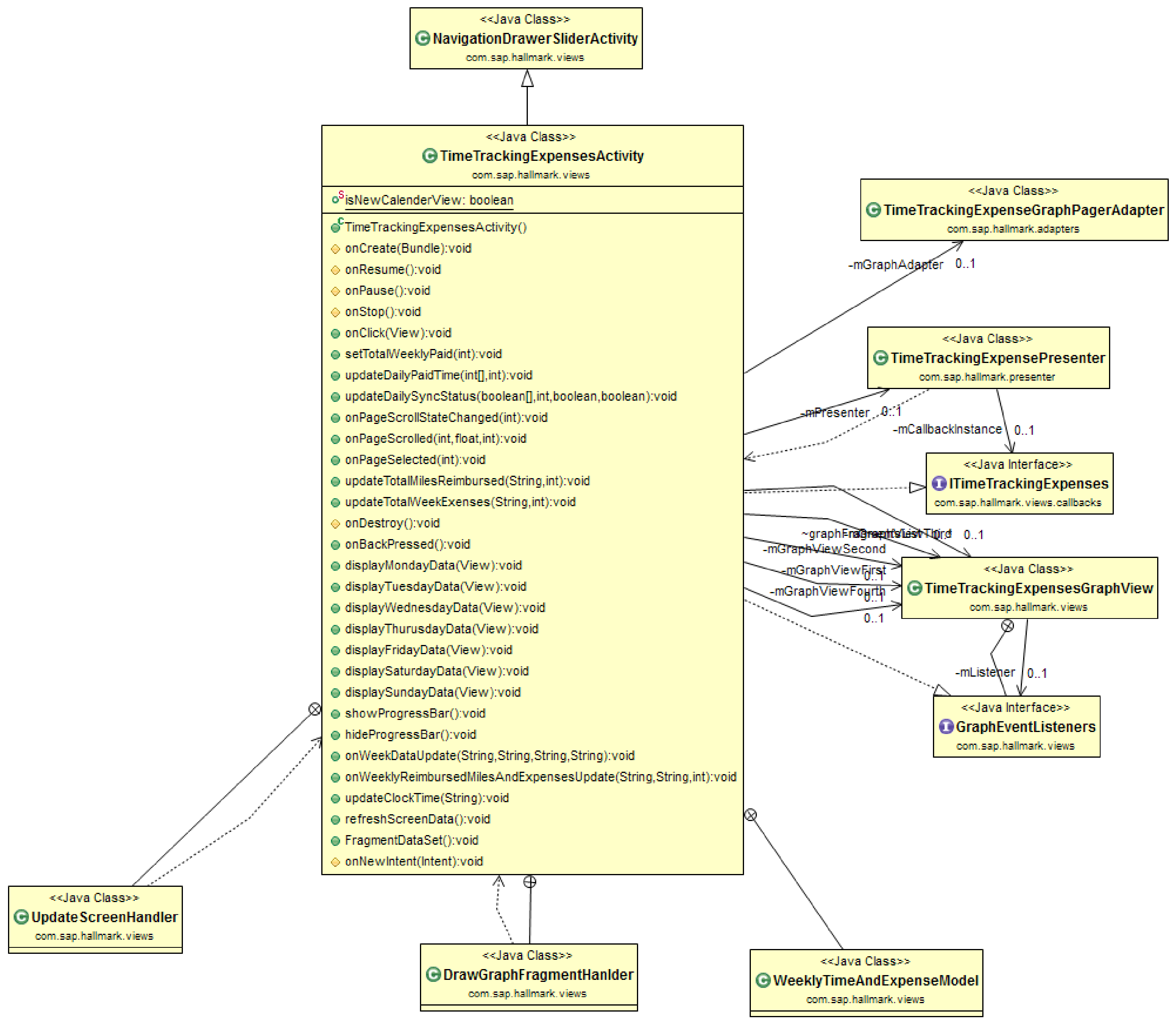
Time Tracking

Priyanka S N
SEP 16, 2015

Agenda

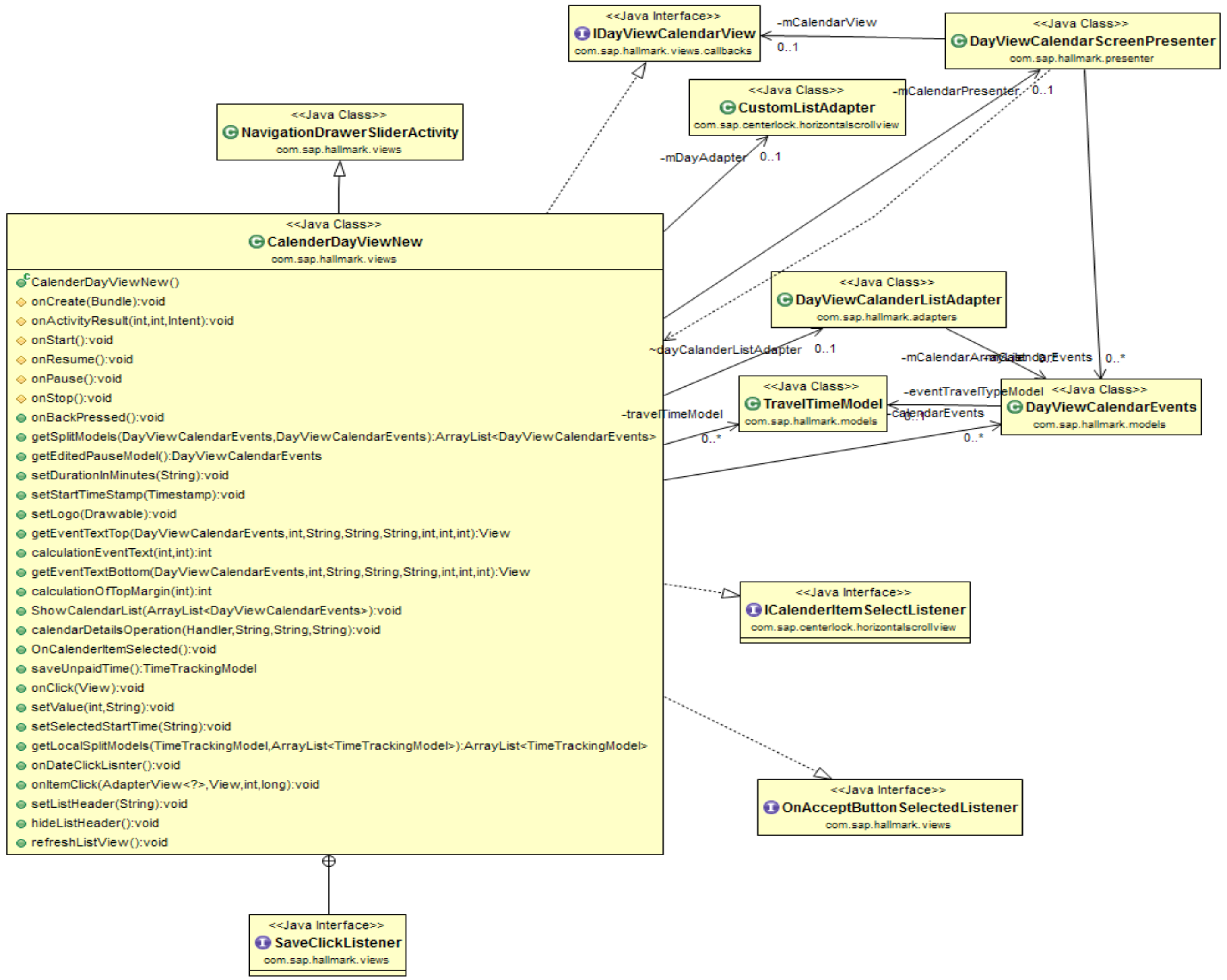
- General Program Structure
- Packages
- Interfaces and Classes
- Special Population
- Time Tracking Automatic
- Weekly Day View
- Day View

General Program Structure



<<Java Class>>
NavigationDrawerSliderActivity
com.sap.hallmark.views
<<Java Class>>
TimeTrackingExpensesActivity
com.sap.hallmark.views
isNewCalenderView: boolean
TimeTrackingExpensesActivity()
onCreate(Bundle):void
onResume():void
onPause():void
onStop():void
onClick(View):void
setTotalWeeklyPaid(int):void
updateDailyPaidTime(int[],int):void
updateDailySyncStatus(boolean[],int,boolean,boolean):void
onPageScrollStateChanged(int):void
onPageScrolled(int,float,int):void
onPageSelected(int):void
updateTotalMilesReimbursed(String,int):void
updateTotalWeekExenses(String,int):void
onDestroy():void
onBackPressed():void
displayMondayData(View):void
displayTuesdayData(View):void
displayWednesdayData(View):void
displayThurusdayData(View):void
displayFridayData(View):void
displaySaturdayData(View):void
displaySundayData(View):void
showProgressBar():void
hideProgressBar():void
onWeekDataUpdate(String,String,String,String):void
onWeeklyReimbursedMilesAndExpensesUpdate(String,String,int):void
updateClockTime(String):void
refreshScreenData():void
FragmentDataSet():void
onNewIntent(Intent):void
<<Java Class>>
TimeTrackingExpenseGraphPagerAdapter
com.sap.hallmark.adapters
-mGraphAdapter 0..1
<<Java Class>>
TimeTrackingExpensePresenter
com.sap.hallmark.presenter
-mPresenter 0..1
-mCallbackInstance 0..1
<<Java Interface>>
ITimeTrackingExpenses
com.sap.hallmark.views.callbacks
~graphFragmentsList 0..1
-mGraphViewThird
-mGraphViewSecond
-mGraphViewFirst 0..1
-mGraphViewFourth 0..1
0..1
<<Java Class>>
TimeTrackingExpensesGraphView
com.sap.hallmark.views
-mListener 0..1
<<Java Interface>>
GraphEventListeners
com.sap.hallmark.views
<<Java Class>>
UpdateScreenHandler
com.sap.hallmark.views
<<Java Class>>
DrawGraphFragmentHanlder
com.sap.hallmark.views
<<Java Class>>
WeeklyTimeAndExpenseModel
com.sap.hallmark.views

Day View



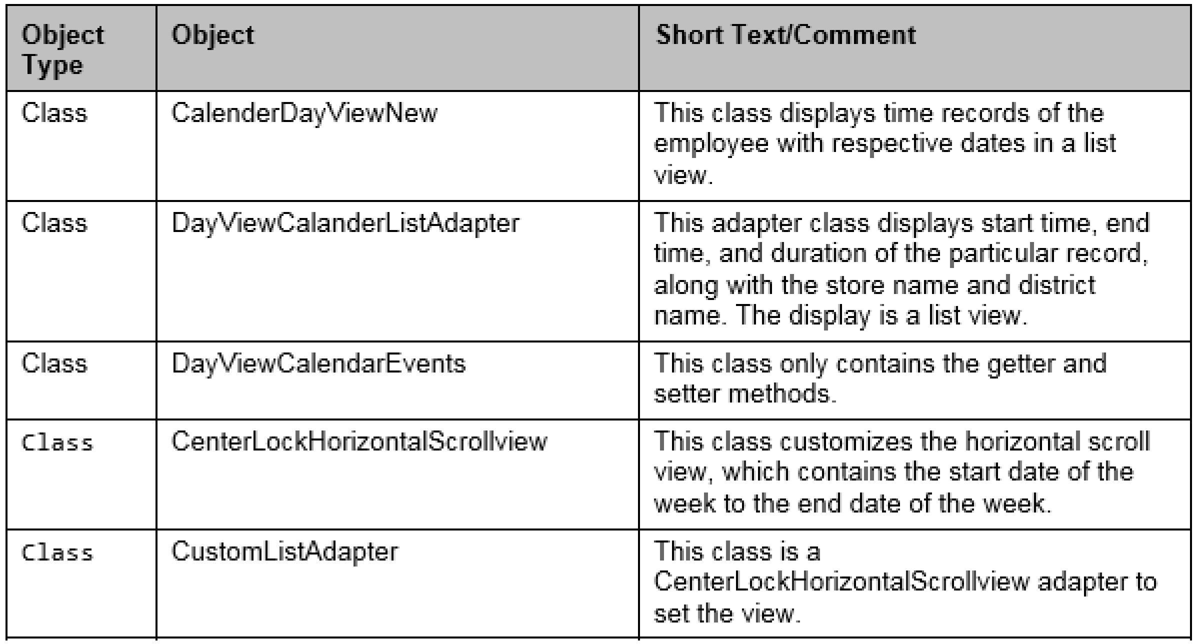
<<Java Interface>>
IDayViewCalendarView
com.sap.hallmark.views.callbacks
-mCalendarView
0..1
<<Java Class>>
DayViewCalendarScreenPresenter
com.sap.hallmark.presenter
<<Java Class>>
CustomListAdapter
com.sap.centerlock.horizontalscrollview
-mCalendarPresenter
0..1
<<Java Class>>
NavigationDrawerSliderActivity
com.sap.hallmark.views
-mDayAdapter
0..1
<<Java Class>>
CalenderDayViewNew
com.sap.hallmark.views
CalenderDayViewNew()
onCreate(Bundle):void
onActivityResult(int,int,Intent):void
onStart():void
onResume():void
onPause():void
onStop():void
onBackPressed():void
getSplitModels(DayViewCalendarEvents,DayViewCalendarEvents):ArrayList<DayViewCalendarEvents>
getEditedPauseModel():DayViewCalendarEvents
setDurationInMinutes(String):void
setStartTimeStamp(Timestamp):void
setLogo(Drawable):void
getEventTextTop(DayViewCalendarEvents,int,String,String,String,int,int,int):View
calculationEventText(int,int):int
getEventTextBottom(DayViewCalendarEvents,int,String,String,String,int,int,int):View
calculationOfTopMargin(int):int
ShowCalendarList(ArrayList<DayViewCalendarEvents>):void
calendarDetailsOperation(Handler,String,String,String):void
OnCalenderItemSelected():void
saveUnpaidTime():TimeTrackingModel
onClick(View):void
setValue(int,String):void
setSelectedStartTime(String):void
getLocalSplitModels(TimeTrackingModel,ArrayList<TimeTrackingModel>):ArrayList<TimeTrackingModel>
onDateClickLisnter():void
onItemClick(AdapterView<?>,View,int,long):void
setListHeader(String):void
hideListHeader():void
refreshListView():void
<<Java Class>>
DayViewCalanderListAdapter
com.sap.hallmark.adapters
~dayCalanderListAdapter
0..1
-mCalendarArrayList
-mCalendarEvents
0..*
<<Java Class>>
TravelTimeModel
com.sap.hallmark.models
-eventTravelTypeModel
0..1
<<Java Class>>
DayViewCalendarEvents
com.sap.hallmark.models
-travelTimeModel
0..*
-calendarEvents
0..*
<<Java Interface>>
ICalenderItemSelectListener
com.sap.centerlock.horizontalscrollview
<<Java Interface>>
OnAcceptButtonSelectedListener
com.sap.hallmark.views
<<Java Interface>>
SaveClickListener
com.sap.hallmark.views

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

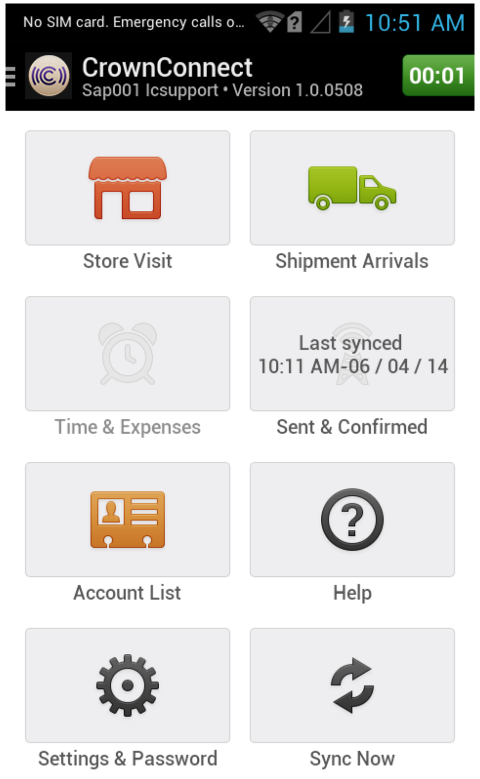
Interfaces, Classes

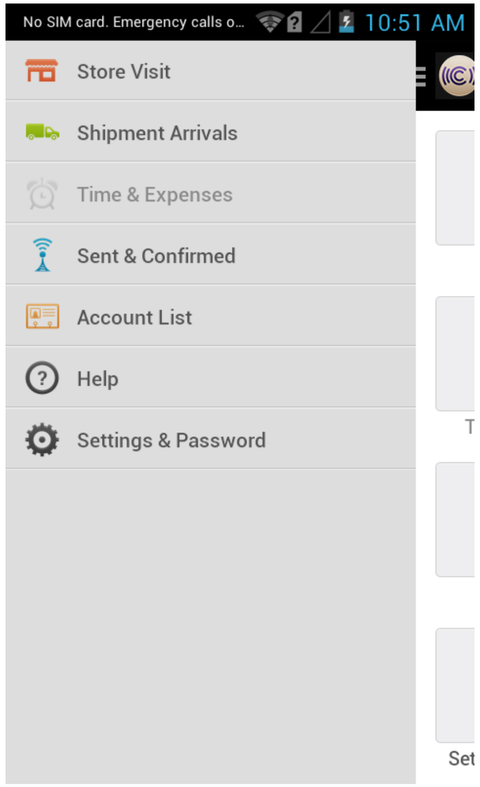


Object Type	Object	Short Text/Comment
Class	CalenderDayViewNew	This class displays time records of the employee with respective dates in a list view.
Class	DayViewCalanderListAdapter	This adapter class displays start time, end time, and duration of the particular record, along with the store name and district name. The display is a list view.
Class	DayViewCalendarEvents	This class only contains the getter and setter methods.
Class	CenterLockHorizontalScrollview	This class customizes the horizontal scroll view, which contains the start date of the week to the end date of the week.
Class	CustomListAdapter	This class is a CenterLockHorizontalScrollview adapter to set the view.

Special Population

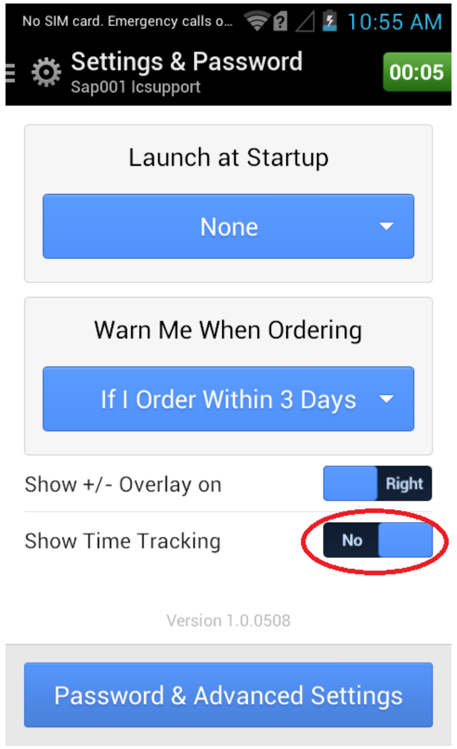
If user is a special population user, then **TimeTrackingExpense** icon is disabled in HomeScreen i.e. user cannot see time records. All the records are submitted to server as a RM service**.**





Special Population

There will be no pause button for SP user. A special population user can see the MealBreak due in time in TimeDrawer screen, but cannot receive meal alerts.



No SIM card. Emergency calls o... 10:55 AM

Settings & Password
Sap001 Icsupport
00:05

Launch at Startup: None

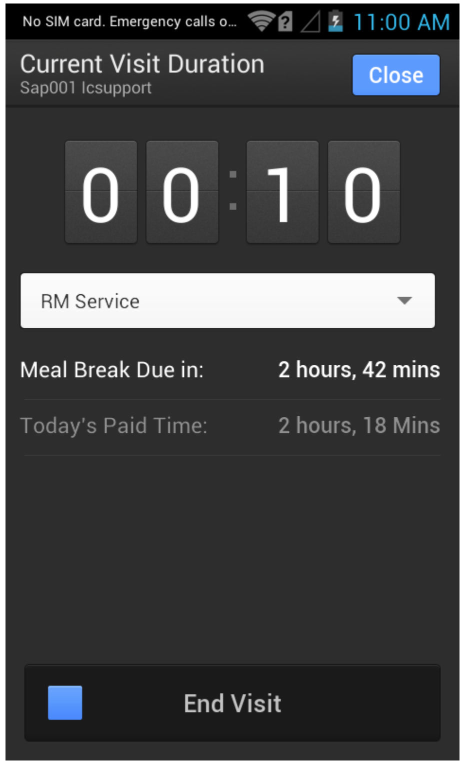
Warn Me When Ordering: If I Order Within 3 Days

Show +/- Overlay on: Right

Show Time Tracking: No

Version 1.0.0508

Password & Advanced Settings



No SIM card. Emergency calls o... 11:00 AM

Current Visit Duration
Sap001 Icsupport
Close

00:10

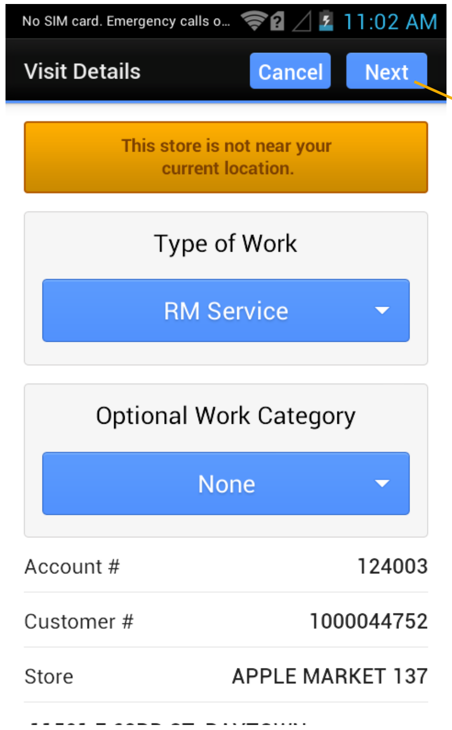
RM Service

Meal Break Due in:	2 hours, 42 mins
Today's Paid Time:	2 hours, 18 Mins

End Visit

Special Population

There is a difference in the store check-in process for SP users.



No SIM card. Emergency calls o... 11:02 AM

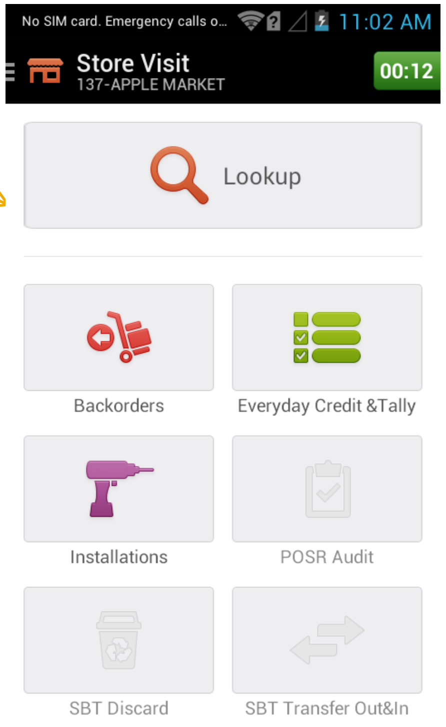
Visit Details — Cancel — Next

This store is not near your current location.

Type of Work: RM Service

Optional Work Category: None

Account #	124003
Customer #	1000044752
Store	APPLE MARKET 137



No SIM card. Emergency calls o... 11:02 AM

Store Visit
137-APPLE MARKET
00:12

Special Population

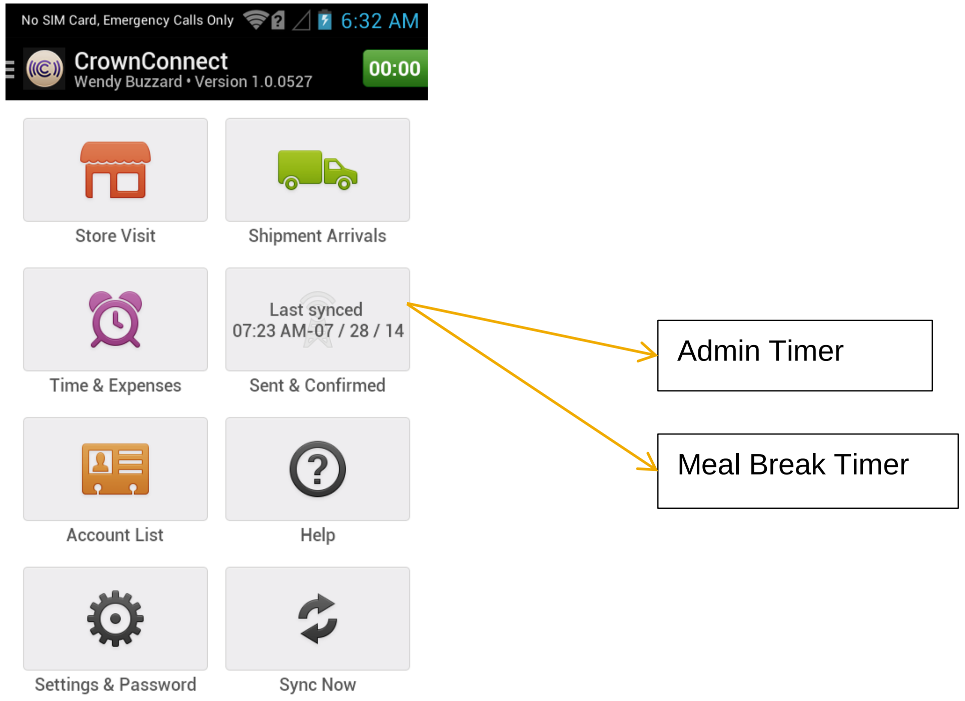
To know if your user is special population user or not, there is an API in AppUtility class :**isSpecialPopulationUser()** → if true, then your user is special population user.If false, then your user is normal user.

```
if(!AppUtility.getAppUtilityInstance().isSpecialPopulationUser() ||
AppConfig.getAppConfigInstance(this).isTimerEnabledForSPUser()) {

                super.startServiceTimer(IAppConstants.TimerManagerConstants.TIMER
_UTILITY_ADMIN_TIME);
}else {

                super.startServiceTimer(IAppConstants.TimerManagerConstants.TIMER
_UTILITY_RM_SERVICE_TIMER_OBJECT);
}
```

Time Tracking

When application first launches (Home Screen), application starts the below TimeCounter instances :

Admin Timer and Meal Break Timer.   (**method :** super.startMealBreakTimer();) these timers are started at the time of activity creation:



Time Tracking

- **Rules to submit time records to DB**

Before submitting any records into DB, application should follow a 30 secs rule which is mentioned below :

Application will submit any time records into database only if :

a) Recorded time is more than 30 secs.

b) There is a change in mins in recorded time .

Time Tracking

10 mins rule for Admin time recording on automatic recording :

- If Admin Time is more than 10 mins and user clicks on **“Begin”** button then at that point of time

  Admin record will submit into database and Service timer will start.

  The service will start from 00:00 hrs. Also the header timer will change to 00:00 hrs.

Time Tracking

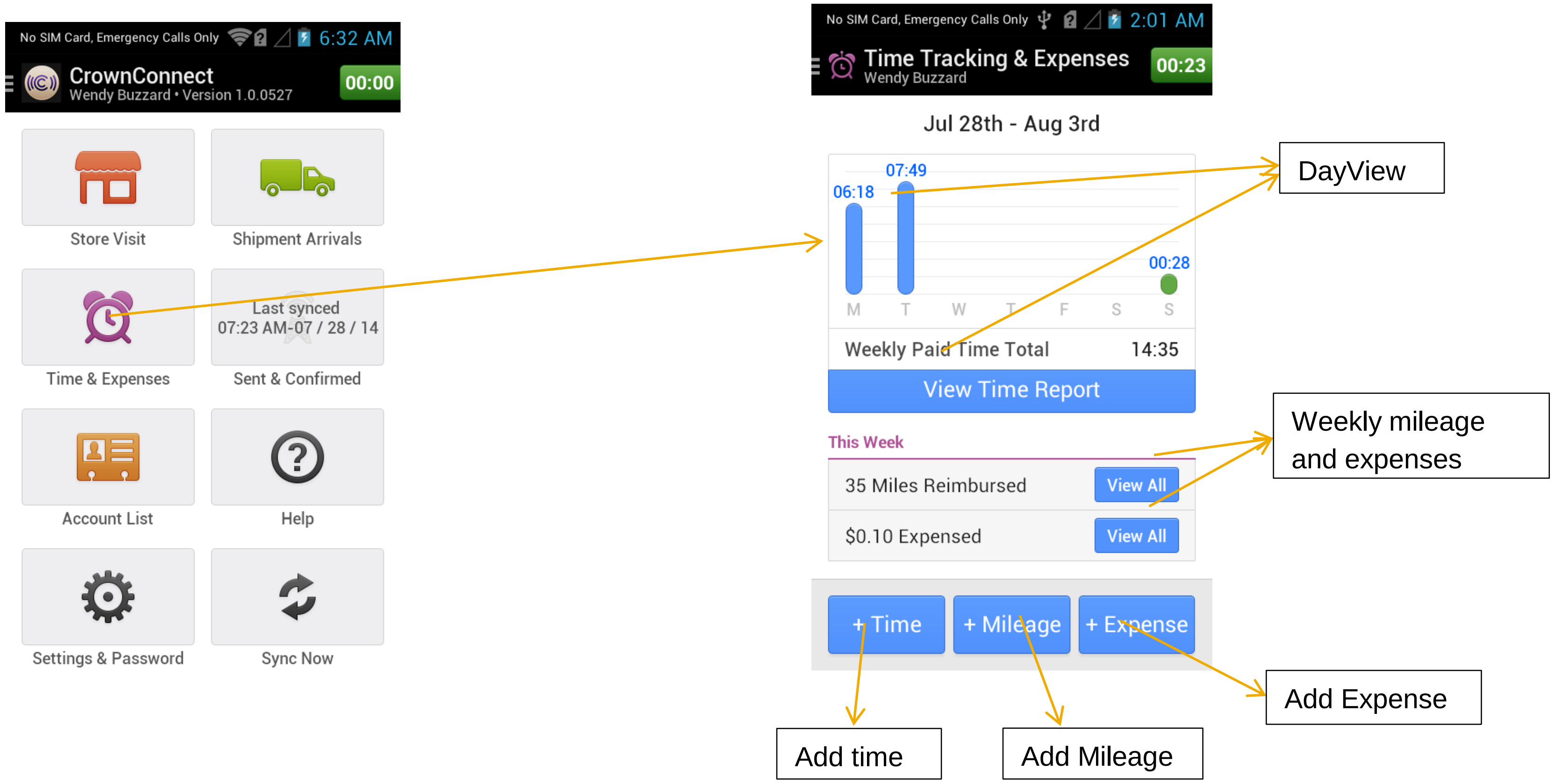
```
if(!AppUtility.getAppUtilityInstance().isSpecialPopulationUser() ||
AppConfig.getAppConfigInstance(this).isTimerEnabledForSPUser()) {
                                        TimeCounter tc =
TimerUtility.getTimerUtilityInstance().getTimerObjectFromContainer(IAppConstants.T
imerManagerConstants.TIMER_UTILITY_ADMIN_TIME);
                    if(tc == null)
                    super.startServiceTimer(IAppConstants.TimerManagerConstants.TIMER
_UTILITY_ADMIN_TIME);}
else {                                                  TimeCounter tc =
TimerUtility.getTimerUtilityInstance().getTimerObjectFromContainer(IAppConstants.T
imerManagerConstants.TIMER_UTILITY_RM_SERVICE_TIMER_OBJECT);
                                                        if(tc == null)
                    super.startServiceTimer(IAppConstants.TimerManagerConstants.TIMER
_UTILITY_RM_SERVICE_TIMER_OBJECT);
                                        }
```

Time Tracking

```
public void startSystemTimer() {
if(!isTimerStarted) {
isTimerStarted = true;
if(mTimer == null) {
mTimer = new Timer();
}

if(mTimerTask == null) {
mTimerTask = new TimerTask() {
@Override
public void run() {
updateCount();
updateMealBreak();
}
};
}
mTimer.scheduleAtFixedRate(mTimerTask, 0, 1000);
}
}
```

Weekly View

When the user clicks on Time & Expenses icon on the HomeScreen or NavigationDrawer, the WeeklyView screen will open :



Weekly View

TimeTrackingExpenseGraphView.java

Every graph bar is a 9-patch image and we are adjusting the bar height based on the recorded time in mins during that day. Bar height can be a minimum of 25px(4 mins) and a max of 140 px (8 hours). The formula to convert the recorded mins to bar height is :

(data[i] * 7) / 24   (data[i] in mins).

Where data[i] is the nth number recorded data in mins starts from Monday.

Weekly View

TimeTrackingExpenseGraphView.java

```
private int getUpdatedTimeinMins() {
                TimeCounter currentServiceTC =
AppUtility.getAppUtilityInstance().getCurrentRunningServiceTimeCounter();
        if(currentServiceTC != null) {
                        long currentTimeinMillis =
currentServiceTC.getCurrentSystemTime()+currentServiceTC.getAccumulatedPaidTim
e();
                        long currentTimeinMins =
currentTimeinMillis/(1000*60);
                        return (int)currentTimeinMins;
}else return 0;
```

Weekly View

TimeTrackingExpenseGraphView.java

The method **drawGraph()** will sets the graph color codes (blue for submitted, green for current running time and red for unsubmitted data).

In this class method updateGraphData() is called from OnResume and for each minute from TimeTrackingExpenseActivity class.

This method refreshes and sets the updated value of data into the graph.

Weekly View

TimeTrackingExpenseGraphView.java

The method :

```
public boolean showSubmitButton() {
				return isUnsubmittedTime;
		}
```

Returns a Boolean value that whether this fragment needs to show **“Please Submit”** button (if there is any unsubmitted data / un-approved time) in this fragment or not.

```
		}
```

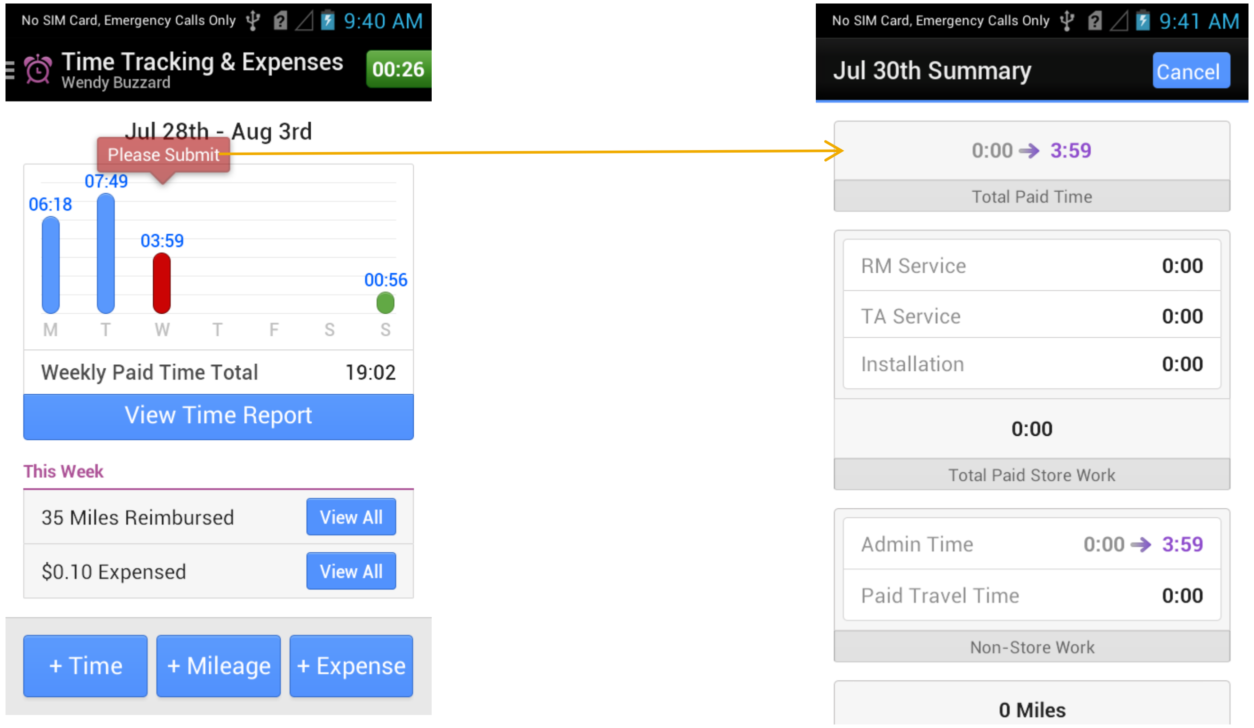
Weekly View

<u>**TimeTrackingExpenseGraphView.java**</u>

If Please submit needs to show then the day number of “Please Submit” button is given by below method :

```
public int getUnsubmittedDay() {
                        return unsubmittedDay;
            }
```

Weekly View


No SIM Card, Emergency Calls Only 9:40 AM

Time Tracking & Expenses
Wendy Buzzard
00:26

Jul 28th - Aug 3rd

Please Submit

Weekly Paid Time Total	19:02

View Time Report

This Week

35 Miles Reimbursed	View All
$0.10 Expensed	View All

+ Time | + Mileage | + Expense

No SIM Card, Emergency Calls Only 9:41 AM

Jul 30th Summary — Cancel

0:00 → 3:59
Total Paid Time

RM Service	0:00
TA Service	0:00
Installation	0:00

0:00
Total Paid Store Work

Admin Time	0:00 → 3:59
Paid Travel Time	0:00

Non-Store Work

0 Miles

If user clicks on please submit then the application moves the user to **“DayData”** which is used to approve the recorded time of that day.

Weekly View

TimeTrackingExpenseActivity.java

This Activity collects all the 4 weeks data using **ThreadPoolExecutor** and displays the current week data in this screen.

To display all fours weeks graph data, this screen contains 4 different integer arrays that contains all 4 weeks graph data's which are :

```
private static int[] first_week_data = {0, 0, 0, 0, 0, 0, 0};

private static int[] second_week_data = {0, 0, 0, 0, 0, 0, 0};

private static int[] third_week_data = {0, 0, 0, 0, 0, 0, 0};

private static int[] fourth_week_data = {0, 0, 0, 0, 0, 0, 0};
```

Weekly View

<u>TimeTrackingExpenseActivity.java</u>

4 boolean arrays that contains the status whether data are synched or not :

```
private static boolean[] first_week_sync_status = {true, true, true, true, true, true, true};

private static boolean[] second_week_sync_status = {true, true, true, true, true, true, true};

private static boolean[] third_week_sync_status = {true, true, true, true, true, true, true};

private static boolean[] fourth_week_sync_status = {true, true, true, true, true, true, true};
```

Weekly View

TimeTrackingExpenseActivity.java

Two different arrays of 4 strings contains week miles and expenses reimbursed of all 4 weeks :

```
private static String[] totalMilesReimbursed = {IAppConstants.EMPTY_STRING, IAppConstants.EMPTY_STRING,
IAppConstants.EMPTY_STRING, IAppConstants.EMPTY_STRING};

private static String[] totalExpensesReimbursed = {IAppConstants.EMPTY_STRING, IAppConstants.EMPTY_STRING,
IAppConstants.EMPTY_STRING, IAppConstants.EMPTY_STRING};
```

Weekly View

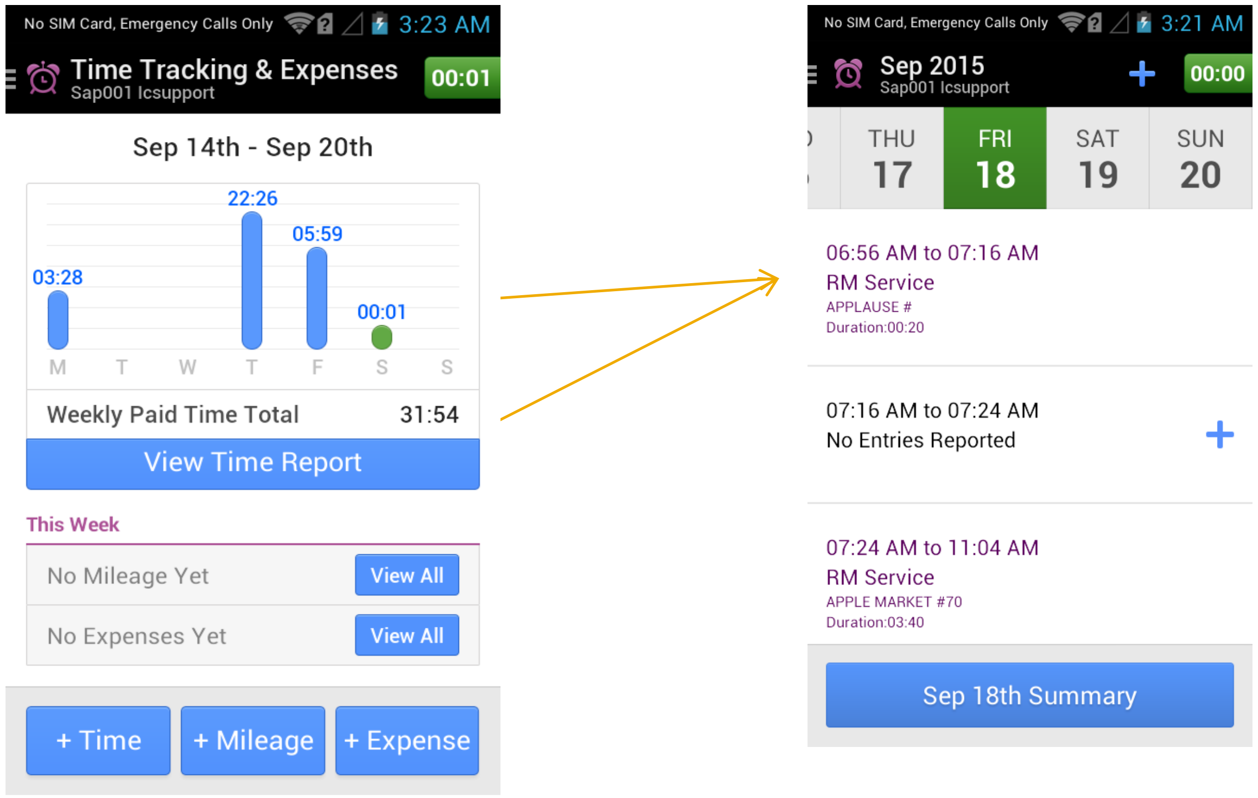
TimeTrackingExpenseActivity.java

- When this activity starts, the application runs a background **ThreadPoolExecutor** that collects the whole month’s data in the background and returns it to the above array variables. This sends the data to the selected graph and the weekly graph fragment.

- The variable ***private static int unsyncedWeekNumber = -1; indicates*** the value of the unsynced week number when application launches for the first time. If there are any previous unsynced weeks, then the application always starts from the WeeklyView screen upon launch and will display the unsynced week graph.

Day View

Day view is responsible for displaying the Individual days' time recording. Each row of the UI contains the timeslots of the recorded and pause times. This class takes the data of the recorded time records, and adds empty and pause records in between the recorded time slots.



No SIM Card, Emergency Calls Only 3:23 AM

Time Tracking & Expenses
Sap001 Icsupport
00:01

Sep 14th - Sep 20th

Weekly Paid Time Total 31:54

View Time Report

This Week

No Mileage Yet View All

No Expenses Yet View All

+ Time + Mileage + Expense

No SIM Card, Emergency Calls Only 3:21 AM

Sep 2015
Sap001 Icsupport
00:00

THU 17 FRI 18 SAT 19 SUN 20

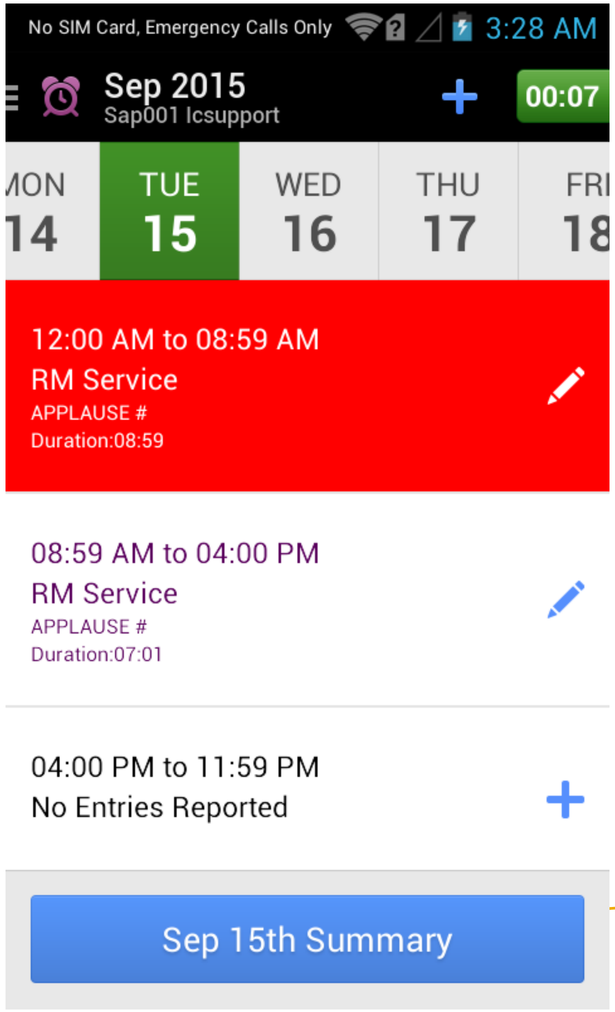
06:56 AM to 07:16 AM
RM Service
APPLAUSE #
Duration:00:20

07:16 AM to 07:24 AM
No Entries Reported

07:24 AM to 11:04 AM
RM Service
APPLE MARKET #70
Duration:03:40

Sep 18th Summary

Day View



No SIM Card, Emergency Calls Only 3:28 AM

Sep 2015
Sap001 Icsupport
00:07

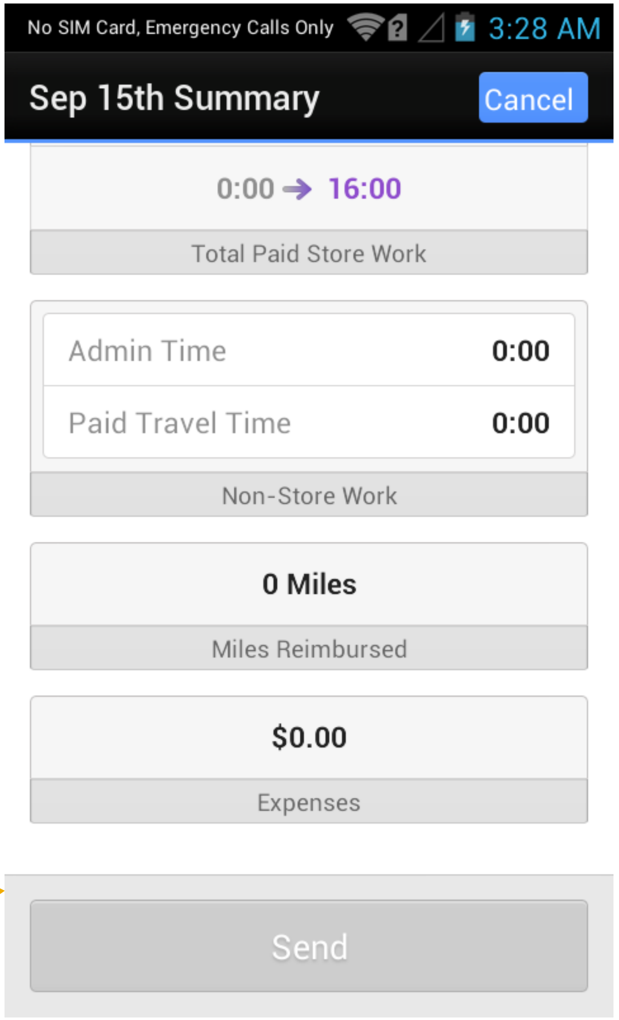
MON	TUE	WED	THU	FRI
14	15	16	17	18

12:00 AM to 08:59 AM
RM Service
APPLAUSE #
Duration:08:59

08:59 AM to 04:00 PM
RM Service
APPLAUSE #
Duration:07:01

04:00 PM to 11:59 PM
No Entries Reported

Sep 15th Summary



No SIM Card, Emergency Calls Only 3:28 AM

Sep 15th Summary
Cancel

0:00 → 16:00
Total Paid Store Work

Admin Time	0:00
Paid Travel Time	0:00

Non-Store Work

0 Miles
Miles Reimbursed

$0.00
Expenses

Send

Admin Time Threshold =6 Hrs.
RM Service/TA Service Threshold Hrs. = 8Hrs
Installation Service Threshold Hrs. = 12 Hrs.

Thank You