Task:

1. \_ID
2. taskType (serviceCall, calibration, maintenance, contract, inventory)
3. itemID (if type is calibration, maintenance, contract, inventory) or serviceCallID (if type is serviceCall)
4. itemName
5. Status (Open, Complete)
6. AssignedTo (string)
7. AssignedToContactInfo (string)
8. dueDateTimeStamp(integer)
9. CompletedTimeStamp (integer)
10. CompletionComments (string)

FTSTask:

1. itemName
2. AssignedTo
3. taskType (in string format)

Task UI:

Same as item.

For Task List UI, the list is searchable for the following attributes:

1. itemName
2. AssignedTo
3. taskType
4. dueDate

The following should be shown for each task in the list:

1. itemName
2. taskType
3. AssignedTo
4. Due date (applicable for Calibration, Maintenance, Contract)

For Task Detail UI, use the following views depending on the taskType:

1. ServiceCallView
2. CalibrationView
3. MaintenanceView
4. ContractView
5. InventoryView

Have a “Completed” button in the action bar. When “completed” button is clicked:

1. Pop up a completion dialog:
   1. Completion Comments
2. Update task entity with:
   1. CompletionComments
   2. CompletedTimestamp
3. Delete the corresponding entry in the FTSTaskTable

Have a “Assign To” button in the action bar. When “Assign To” button is clicked:

1. Pop up a Assign To dialog:
   1. Assign To – should be linked to Android Contacts Manager for suggestions.
   2. Contact Info
2. Update task entity with:
   1. Assign To
   2. Contact Info
3. Update the corresponding entry in the FTSTaskTable

For populating the task table:

1. Have action bar button on TaskList view called “Query for new tasks”. This will show a dialog box telling about delay due to intensive computing to be done in the background.
2. A service that runs that 1:00 am to computeTasks().
3. Should use the JobInfo API for Lollipop (and higher). If older OS is found, use the AlarmManager (implement this in scheduleJob()).
4. A settings UI to change the 1:00 am default time for computeTasks().
5. ComputeTask():
   1. There can only be one task per service call and for the tuple (itemID, taskType)
   2. Collect all currentTasks in list1.
   3. Compute new tasks and put them in list2
   4. Subtract list1 from list2
   5. Add remaining list2 items to the Tasks Table and the FTSTaskTable.
   6. All new tasks have assignedTo set to “None”.
   7. The message to show is “The new tasks are being computed. This may take a while. Please check back after a few minutes.”

Item Attributes

1. Type (Instrument, Consummable)
2. For Instrument:
   1. Reminders
      1. Service Call
      2. Maintenance
         1. Frequency (every x days)
         2. Last Done date
         3. Instructions
      3. Calibration
         1. Frequency (every x days)
         2. Last Done date
         3. Instructions
      4. Contract
         1. Frequency (every x months)
         2. Last renewed date
         3. Instructions
3. For Consummable:
   1. Minimum Required Quantit
   2. currentQuantity
   3. Need UI for consumable checkin/checkout
   4. have “+” and “-“ action bar buttons in the ItemDetail if the item is a consumable

Service Call Entity:

* + - 1. \_ID
      2. itemID
      3. Description
      4. Priority
      5. Status (Open, Closed)
      6. OpenTimeStamp
      7. ClosedTimeStamp

Misc:

1. Validations!
   1. Implement item validations inside updateItemFromUI.
   2. Ditto for task
   3. Service Call Dialog
2. Overhaul icons.
3. Send SMS/email on task assignment
4. Overdue tasks should be highlighted somehow
5. Settings UI for:
   1. When computeNewTasks() will run
6. Add backup and Restore
7. Add location field to item and task
   1. Add location to sms messages when assigning tasks
8. Populate contact no. in the task entity when assigned and use that for sms during task allocation
9. Store images as blobs in db (instead of files). This makes the following easy:
   1. Backup and restore
   2. Migration to mysql
10. One time code for alarm manager
11. How to reset the alarm to the new time if a user changes the refresh time
12. Add image support to items
    1. Delete the image file when deleting the item
13. Need UI for service call
    1. Have a “service call” action bar button in ItemDetail
       1. Enabled only for instruments
    2. Pops up a dialog for filling out Service Call entity attributes

Enterprise Features:

1. Add barcode support to items
2. MySQL support
3. NO userids – everybody gets the same app with the same data