Final Submission for Vimigo Technical Assessment

Yeoh Hui Jia (Elle) for Flutter Developer Engineer

This PDF serves to accompany the POC application submitted. However, not all information in this PDF may appear with the created application and will only serve to provide an idea on a more theoretical standpoint.

As understood from the brief, the task is to come up with a **scalable proof-of-concept (POC) application** that allows **proper management of online and offline modes** of the end user (employee)'s application, allowing them to maintain **consistency** and **completeness** of data related to their daily tasks.

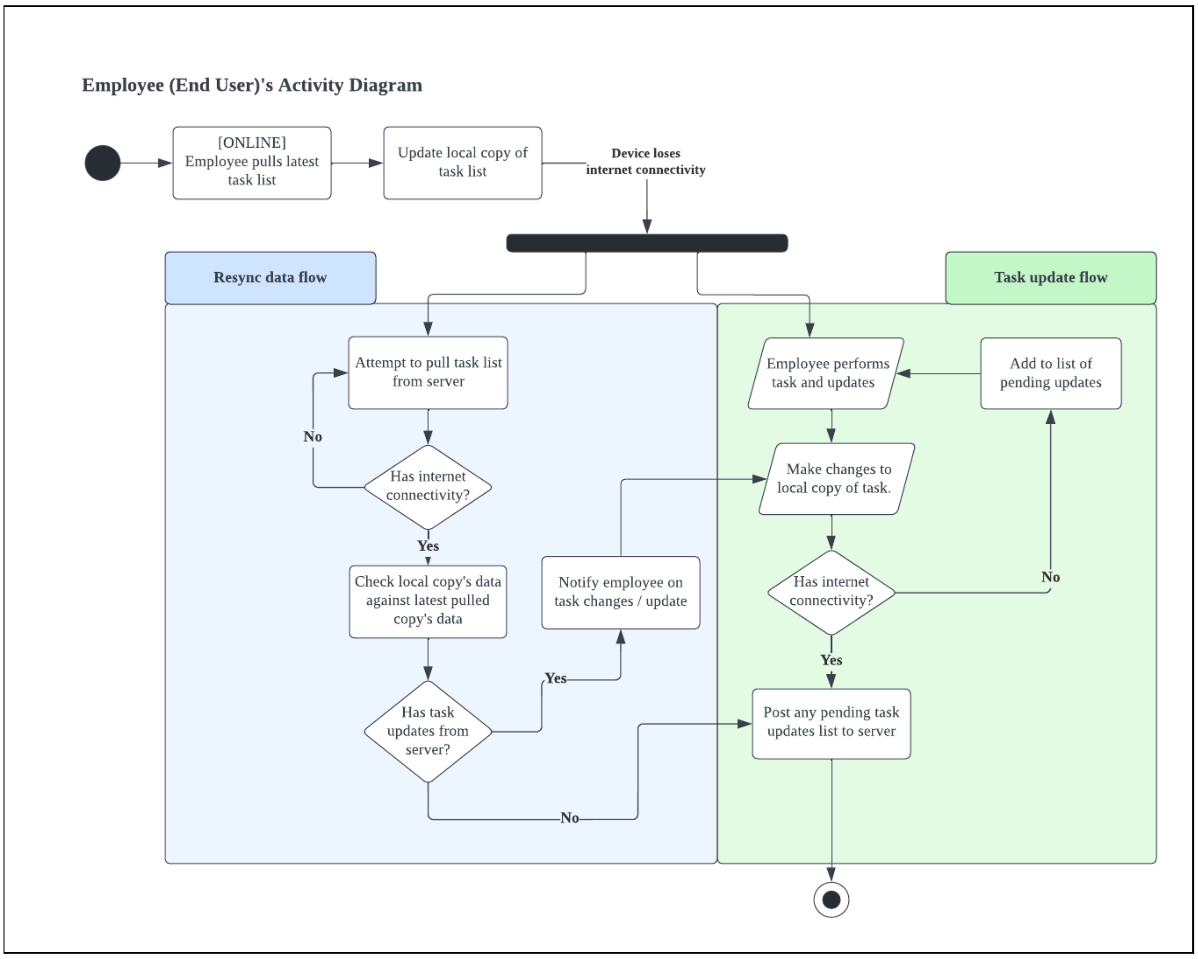
The following is the breakdown / idea of the proof-of-concept that has been devised with regards to the problem statement:

Problem Statement:
Employees who have lost network connectivity on their mobile devices are unable to obtain any updates in real time nor be informed about it, thus potentially leading to inconsistency and incompleteness in their daily task list.

Understandings & Assumptions:

- The employer (system user) always has a reliable internet connection and can perform updates to the employees' tasks.
- Each employee's mobile device has allowed notifications to be turned on for the application.
- Each employee has a working phone number that they are contactable with.
- Employees are able to perform their tasks and updates offline, but regardless will still need to resync their updates to the server at least by the end of the day. This puts the assumption that the employees will at least need to have internet connectivity twice a day.

Breakdown:
The main idea behind this proof-of-concept includes the use of local storage, caching, periodic polling / websockets and background processing capabilities of a mobile device.


Employee (End User)'s Activity Diagram
[ONLINE] Employee pulls latest task list
Update local copy of task list
Device loses internet connectivity
Resync data flow
Task update flow
Attempt to pull task list from server
Has internet connectivity?
No
Yes
Check local copy's data against latest pulled copy's data
Has task updates from server?
Yes
No
Notify employee on task changes / update
Employee performs task and updates
Add to list of pending updates
Make changes to local copy of task.
Has internet connectivity?
No
Yes
Post any pending task updates list to server

Resync Task List Flow:

After losing internet connectivity, the application will be unable to communicate with the server. In the proposed flow, the application will periodically attempt to *(GET)* sync the task list with the server via polling approach. A local copy of the tasks will be kept to ensure a seamless user experience, allowing the employee to view the last updated copy of their task list without the need for internet connection.

During each poll cycle, the application will check if the device is connected to the internet / network. If it is not, it will skip that poll cycle and await the next cycle after the interval to attempt the same process.

If the device is pushed to the background, a background process will run to maintain this same checking, ensuring that even though the application is not active, the task list will still be updated.

Task Update Flow:

All tasks performed by the user in the application will have its changes made to the employee's local copy first.
In the event of no internet connection, the device will cache a list / dictionary of unsynced changes locally until internet connection is restored.
Upon the next sync cycle of tasks or next update of a task, the device will also then check if there are other pending changes to be synced and will update (*POST*) the corresponding changes to the server.

How do Employees Get Informed?

In terms of informing the employee on task updates, the application may take up notifying via Push / Local notifications when the device is online and via SMS if the device is offline.

Push / Local notifications
Using push notifications will allow employees to be aware of any updates coming from the server. For example, when the polling method completes its comparison and finds that there are updates to existing tasks, the application can schedule a local notification to inform the user of this. Alternatively, the server could opt to use push notifications to notify users of any updates made to their tasks.

SMS
In my opinion, using SMS to notify users who are offline would serve as a plausible solution. If the employee loses internet connection, he/she will not be able to receive any form of notifications via the app but may still be able to receive it via SMS. The SMS could contain a simple message informing them of task updates and prompting them to regain network connectivity at their soonest convenience to sync.

Scalability

On the subject of scalability, this proposed concept will not incur too heavy of a load when it attempts to resync the task list as the interval of syncing can be adjusted to prevent possible overloading on both the mobile and the backend. As there is also a step to check if the device is connected to the internet, there is no creation of resources to submit a HTTP request that would definitely fail. As the background fetching also runs on a 15 minute interval minimum, there is also minimal stress upon the user's device.

This concludes my submission for the technical assessment, my take and elaboration on the management of online and offline modes for the employee's mobile application. Thank you.

Appendix:

This is the structure of the JSON used in the API calls the POC application connects to:

JSON structure:

id : String
title : String
details : String
status: Int
isCompleted : Bool
assigneeId : String
createdAt : DateTime
assigneeUpdatedOn : DateTime
employerUpdatedOn : DateTime

API Details:

Endpoint: https://626170fd73499e9af90c5978.mockapi.io

Methods:

1. **[GET]** Obtain task list: ***{{endpoint}}/tasks***
2. **[GET]** Obtain single task by ID: ***{{endpoint}}/tasks/{{id}}***
3. **[POST]** Create a new task: ***{{endpoint}}/tasks***
4. **[PUT]** Update task by ID: ***{{endpoint}}/tasks/{{id}}***
5. **[DELETE]** Remove task by ID: ***{{endpoint}}/tasks/{{id}}***

Sample PUT & POST body:

```
{
  "createdAt": 1650553420,
  "details": "App Y update to News screen title and description",
  "assigneeUpdatedOn": 1650553420,
  "assigneeId": 1,
  "status": 1,
  "isCompleted": false,
  "title": "[App Y] Update News screen title",
  "employerUpdatedOn": 1650553420
}
```