# Build a desktop note taking application (like Evernote/OneNote, etc.)

# Requirements

1. A note is comprised of a title and a body of text.
2. Users can store multiple notes.
3. Users can select a note for editing. The title and the body can be edited.
4. Copies of notes are stored on the cloud as well, so that they can be edited by web versions of the application.
5. Changes that were made on the cloud are also synced back to the desktop application.
6. The desktop application can work offline. As a result, the changes made to notes on the cloud and desktop are synced when it is back online.
7. In the case of a conflict when the same note is modified both online and offline, the desktop copy of the note is retained and the copy of the note on the cloud is discarded and replaced with the desktop copy.
8. When notes are being synced, user experience must not be hampered by blocking user's interaction with the app.
9. Also, users value the ability to open a note and start editing it immediately. So, syncing a note during open will not result in a good user experience.
10. A note can be also be shared over email, using the default mail client on the desktop.

Note: The scope of the case study is limited to building the desktop application. While the web application is outside the scope of this exercise, there should be a means to simulate changes made to the cloud copy of the notes, so that the sync can be validated.

Please feel free to use any Microsoft technology to build the app.

HTTP must be used for communication between the desktop and the cloud.

The use of automated unit/integration tests and suitable mocks/stubs is highly encouraged.

Apart from a working prototype, please be ready to discuss the design (design patterns, classes, interfaces, etc.) of the application and its testability.

# High Level Design:

## Considerations:

Data storage can be database or file

**Database:**

If database is used table format should be

|  |  |
| --- | --- |
| Table : **Notes** : to store notes | |
| ID | (Ex D/C\_DateTime) while creating note, mention whether it is created via DesktopApp or cloud. This will help in generating unique id. Primary key. |
| Title | Title of note: string of restricted length |
| Note | Notes: stirngs of max length |
| LastUpdateTIme | Last Update Time: this should be used to consider which copy(desktop/cloud) is latest. Sync and save the same. |

|  |  |
| --- | --- |
| Table : **TrackChanges** : to keep track of changes: rather than comparing entire **Notes** table, this will simplify the sync process | |
| ID | (Ex D/C\_DateTime) while creating note, mention whether it is created via DesktopApp or cloud. This will help in generating unique id. This should be unique key and part of **Notes** table. |
| Operation | New/updated/delete |
| Time | In case of delete , row doesn’t exists in notes table to compare |

**File:**

XML format

Node to store notes:

<MyNote>

<Notes>

<D/C\_DateTime>

<Title>Note1</Title>

<Body>Here is my note</Body>

<LastUpdated>DateTime</LastUpdated>

</ D/C\_DateTime >

<D/C\_DateTime2>

<Title>Note2</Title>

<Body>Here is my note</Body>

<LastUpdated> DateTime </LastUpdated>

</ D/C\_DateTime 2>

</Notes>

<TrackChanges>

<D8\_28\_2016\_11\_49\_30\_PM Operation="Add" Time="8/28/2016 11:49:29 PM" />

<D8\_29\_2016\_12\_00\_44\_AM Operation="Delete" Time="8/29/2016 12:01:35 AM" />

<D8\_29\_2016\_12\_02\_14\_AM Operation="Delete" Time="8/29/2016 12:03:18 AM" />

<D8\_29\_2016\_1\_32\_46\_AM Operation="Add" Time="8/29/2016 1:32:46 AM" />

<D8\_29\_2016\_1\_33\_05\_AM Operation="Add" Time="8/29/2016 1:33:05 AM" />

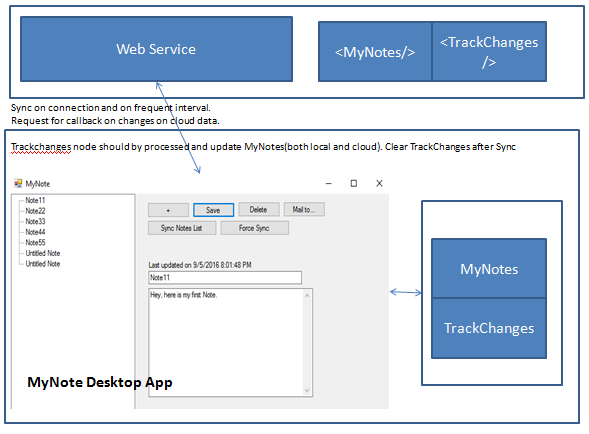
</TrackChanges>

<SyncFile>SyncFileNote.xml</SyncFile>

<SyncInterval>120</SyncInterval>

</MyNote>

**High Level Architecture:**



**Classes:**

**XML Interaction**

*Variables:*

Dictionary – to save requested changes initially

XMLNode – to communicate with XML and make changes

*Methods:*

*GetNotes()*

Public PushChanges() – push data dictionary

Private PushToXML() – this method run on different thread to update XML.

Public SyncWithCloud() – If sync has detected online changes this will run in new thread and update the local and cloud data.

AddNote()

UpdateNote()

RemoveNote()

**Sync:**

Communicate with cloud app and wait for call back if any changes happened on cloud.

CheckOnlineChanges()

**MyNote: G**UI Class -

*Variable*

XMLInteractionObject – to retrieve note details and push changes carried via GUI.

Methods:

CreateNewNote()

GetNoteInfo()

UpdateNote()

RemoveNote()

MailNote()