Advanced Software Engineering
Note Taking Application
Second Iteration Demo
Team: Code Phoenix
Project Mentor - Fujunku Chen

Team Members:
ag4015 (Aviral Gupta)
ag4020 (Arjun Gupta)
smk2256 (Sachin Kelkar)

Date and Time of Demo:
December 4, Thursday between 3:15 - 3:45 PM

The demo went smoothly and there were no challenges as such.

User Stories and Use Cases Demonstrated during Demo:
All the use cases shown in the demo were in addition to those during the first iteration, no first iteration features were changed in this iteration.

User Story 1 -

As a user, I want a note-taking mechanism that allows me to take notes right in the shell which I am using for development/general use. This avoids the overhead of using a separate application and saves time. Along with notes, I want the app to handle my events, to do lists, reminders, etc. My conditions of satisfaction are -

A. Should have an interface to quickly launch the app.
B. Should be able to create new notes quickly and store to a local storage on the device running the application.
C. Should be able to show previously saved notes taken on the application.
D. Should be able to edit previously saved notes taken on the application.
E. Should support multiple types of notes (Note, Event etc).
F. Before starting any note, I should be asked to select the type.

A, B, C and D have already been implemented. E and F are part of the wishlist and will be implemented as a part of the final iteration.

Use Case -

Title - Launch the application from the command line and create a new note.

Actor - User of the application

Description - The user launches the application and chooses one of the available options shown to him on the main menu.

Basic Flow -

1. The user navigates to the directory with the application and types python notes.py in the terminal which opens the main menu with list of options.
2. The user can create a new note on the application by pressing ‘a’ followed by enter.
3. The user is then asked to input the title of the note and the contents of the note.
4. The user is asked whether he would like to save the note or not.
5. If the user chooses to save the note, he is asked to input a password to encrypt the content of the note before storing the note.
6. The user is also asked whether he wants to sync the note to Google Drive.
7. After this, the user is redirected to the home screen for the next action.

Alternate Flow -

Flow 1 -

1. The user can view previously stored notes by pressing ‘v’ followed by enter.
2. The user is guided to a screen with paginated entries showing a fixed number of entries.
3. The user can select a number, to open the entry corresponding to that entry.
4. The user can go to the next page by pressing ‘n’ or the previous page by pressing ‘p’.
5. The user can go to the previous menu by pressing ‘q’.

Flow 2 -

1. The user can quit the application by pressing ‘q’ followed by enter.

User Story 3 -

3. As a user who takes notes on multiple platforms, the notes which I explicitly select should be available to me on any device through syncing with a cloud service of my choice. My conditions of satisfaction are -

    A. The app should support selective sync to cloud services of my choice(Google Drive, MS OneDrive, etc.)

    B. I should have an option to select which notes to sync.

    C. The notes that are already synced with some cloud service should sync automatically on updates.

D. Conflicts(for example, if I modify a note directly on the Google Drive and also on my local machine so that sync cannot be performed automatically), if any, should not be resolved automatically, but only after asking me.

We changed A to support only Google Drive for now. Multiple-cloud sync support will be supported if time permits.
B, C, D have been implemented.
Use Case

Title - Sync notes to Google Drive

Actor - User of the application

Description - The user should have an option to backup his note to a cloud provider(in this case Google Drive) for backup purposes. The user might also edit the notes on Google Drive and thus these changes should be reflected in the corresponding note in the note taking application. The conflicts, if any, should be resolved after asking the user.

Basic Flow -

1. When the user saves a new note, they are given a prompt whether they want to sync with Google Drive.
2. If the user selects “Yes”, a *sync* flag is stored as True in DB against the note and the note is then uploaded to the Drive of the user and they exit to main menu.
3. Next time when the user opens this note, the note is checked against the copy on Google Drive. If there is a mismatch/updates, the user is given option to update local copy.
4. If the user selects “Yes”, the local note updates and otherwise it doesn’t.
5. Now when the user saves this note, the file is automatically synced with Google Drive.

Alternate Flow -

When user decides to not sync the note

1. Instead of choosing to sync the note if the user selects no sync, the file is saved normally.
2. Next time when the file is opened, the local copy loads as this file is not being synced.

When there is no Internet connection and sync is enabled

1. When the file is being synced and the user tries to save the file or open it, an error throws up and the Note Application continues as normal by pressing return.
2. The local copy of note is still created on save.

User Story 4 -

1. As a user who will modify the same notes multiple times, I want the app to keep a version history of my notes. My conditions of satisfaction are -
    A. Keep versioned copies of the notes
    B. It should store upto last 10 versions
    C. Versioning should be automatic
    D. Should be able to get the difference between two versions of a note

Use Case -

Title - Notes Versioning

Actor - User of the application

Description - Maintain and View previous versions of a note and get difference between any 2 versions of the note.

Basic Flow -

1. When the user saves a new note, a versioned copy is created and stored in local storage.
2. When the user chooses to edit a note, after inputting the correct password, he is taken to a menu where he can decide to view the note, edit the note, view previous versions or return to previous menu.
3. On pressing ‘v’, the user is taken to a menu with a list of previously stored versions for the selected note.
4. He can input the index corresponding to a version entry to view the contents of that version.

Alternate Flow -

1. Instead of inputting the index of an entry, the user can press ‘d’ to get a diff between two versions.
2. The user is prompted to enter the index of the first version for diff.
3. The user is prompted to enter the index of the second version for diff.
4. If both versions inputted are valid, the user is shown the diff, otherwise he gets a prompt which tells him the input is invalid and asks him to press enter key to continue.

CI Mechanism:

We used pytest as our unit testing tool, pylint as our static analysis and bug finding tool, and coverage plugin for pytest to check coverage.
We ran the above tools in the pre-commit and also on TravisCI after every pull request request. TravisCI also uploads the coverage reports to the gh-pages branch which has also been shown.

GitHub Repository:

https://github.com/s4chin/coms_4156

Version Shown during Demo:

https://github.com/s4chin/coms_4156/releases/tag/v2.0