Ng Shi Wei - Project Portfolio

PROJECT: TagLine

About the project

My team of 5 software engineering students were tasked with enhancing a basic command line interface desktop AddressBook application for our Software Engineering project. We chose to morph it into a note management application called **TagLine**. This enhanced application enables students to store their notes in a centralised application; organise notes related to contact, topic or project group; and filter and find their notes with ease.

This is what our project looks like:

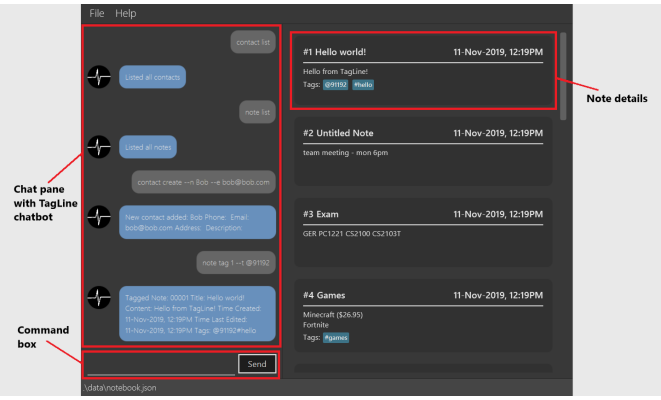


Figure 1. The graphical user interface for TagLine

My role was to design and write the codes for the note commands, including basic add, delete, edit command as well as a filter feature. The following sections illustrate these enhancements in more detail, as well as the relevant documentation I have added to the user and developer guides in relation to these enhancements.

Note the following formatting used in this document:

- `note list cs` - A grey highlight mark-up (OR red text in PDF) with spaced text indicates that this is a command that can be input into the command line and executed by the application.
- `ListNoteParser` - A mark-up (OR red text in PDF) in *PascalCase* indicates the class used.
- `ListNoteParser#parse()` - The *camelCase* text after the '#' indicates the method called in the class.

Summary of contributions

This section shows a summary of my coding, documentation, and other helpful contributions to the team project.

Enhancement added: I added the ability to store and filter notes

- What it does:
  - The `note create` command creates and stores a note to TagLine.
  - The `note delete` command deletes the note from TagLine.
  - The `note edit` command edits the note in TagLine.
  - The `note list` command filters and list notes according to the user input provided.
- Justification: In the case that the notes in TagLine grows, it will be difficult to find notes. The `note list` command allows users to easily find and filter the notes that they require.
- Highlight
  - Note filtering work with both keyword filtering as well as tag filtering. A user can specify a keyword, if the keyword is found in note content or title, the note will be displayed. Users can also tag their notes to organise them, as implemented by other members. The `note list` command can filter the notes using tags.
  - An in-depth analysis of design alternatives was necessary for the code style and design to be consistent with other features. Also, future changes and additions should be easy to be implemented if need be.
  - The implementation was challenging as TagLine has 3 different tags – Hashtags, Contact as tag, and Group name as tag. The note list command has to be implemented such that the filtering applies generally to all 3 tags.
  - Credit: The architecture and code of note create/delete/edit commands were reused and morphed from AddressBook commands.

Code contributed: Please click these links to see a sample of my code: [ListNoteParser] [ListNoteCommand] [ListNoteParserTest] [ListNoteCommandTest]

Other contributions:

- Documentation:
  - Add the details for basic note commands in user guide: NoteCommand
  - Add the logic of note commands in developer guide after their addition to TagLine: NoteLogic
- Community:
  - Reviewed and offered suggestions for Developer Guide of other teams in class (examples: SecureIT#22)

Contributions to the User Guide

We had to update the original AddressBook User Guide with instructions for the enhancements that we had added. The following is an excerpt from our **TagLine User Guide**, showing additions that I have made for the `note` commands.

List notes: `list`

Lists all notes in the application. Filters can be applied to show only notes related to certain keywords, hashtags, users or groups.

Format:

`note list [# / @ / %][FILTER]`

Example:

Format	Example	Outcome
`note list`	`note list`	Lists all notes.
`note list KEYWORD`	`note list meeting`	Lists all notes which contain the phrase "*meeting*".
`note list #HASHTAG`	`note list #cs2100`	Lists all notes with the hashtag '*#cs2100*'.
`note list @CONTACTID`	`note list @12345`	Lists all notes tagged with the contact of ID '*12345*'.
`note list %GROUPNAME`	`note list %cs2103t`	Lists all notes tagged with the group '*cs2103 team*'.

1. When you would like to see all the notes you have in TagLine, you can enter the command `note list`.

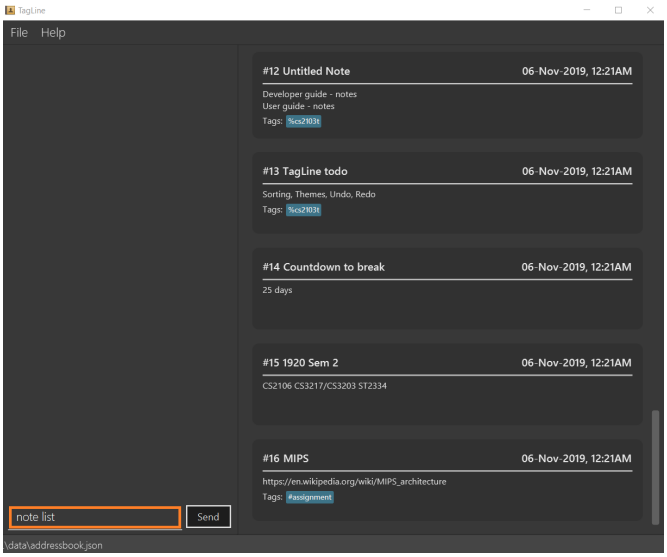


Figure 2. Entering `note list` command

2. All notes are displayed.

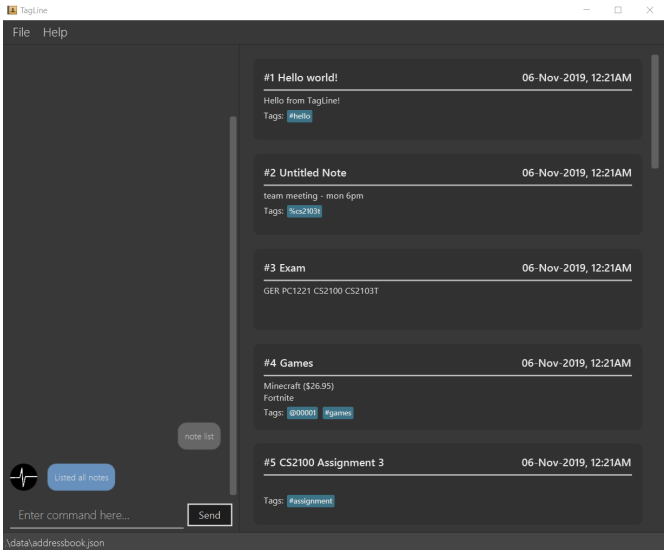


Figure 3. All notes displayed

3. When you would like to find the notes containing the keyword "cs", you can enter the command `note list cs`.

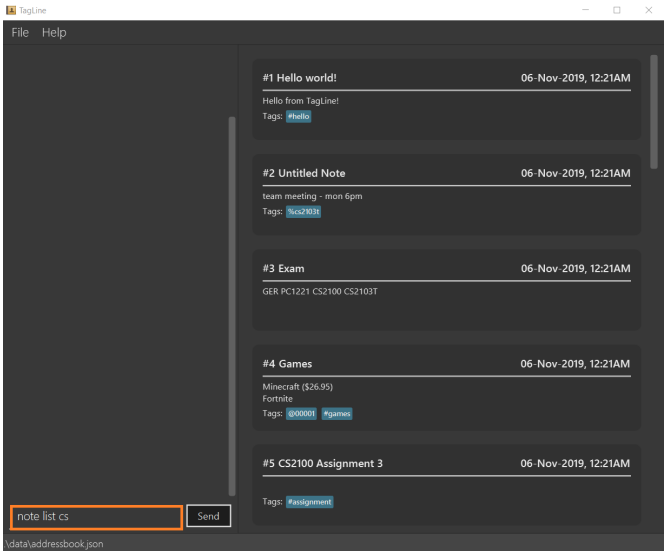


Figure 4. Entering `note list` command with keyword

4. Notes with the keyword "cs" found in the title or content are displayed.

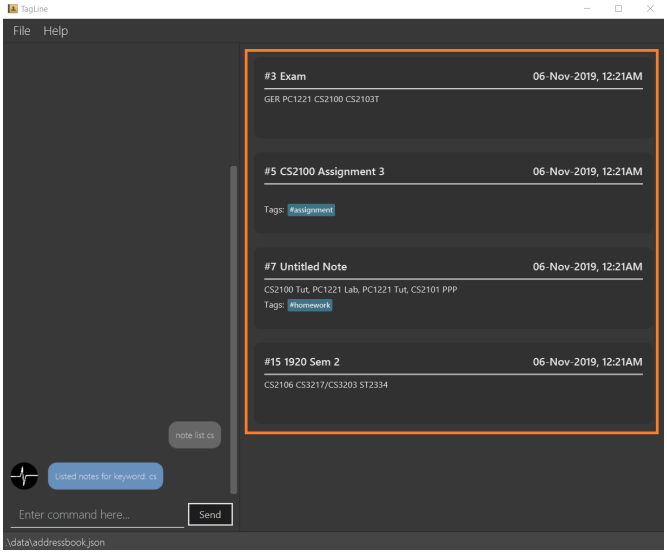


Figure 5. Notes containing keyword displayed

5. When you would like to see the notes tagged with the hashtag "assignment", you can enter the command `note list #assignment`.

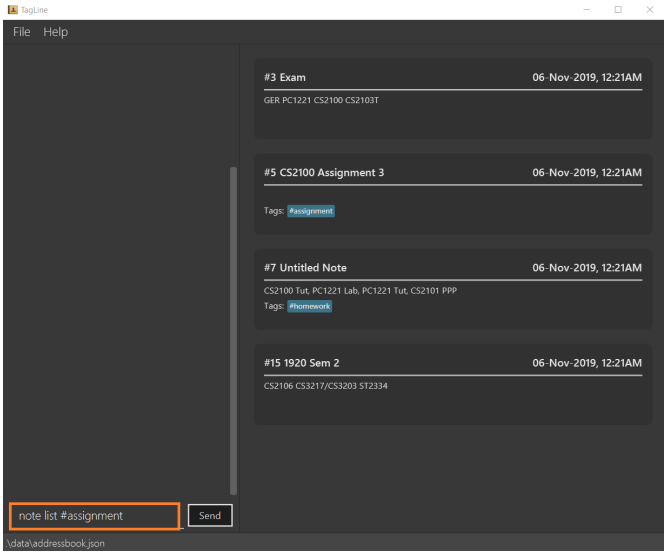


Figure 6. Entering `note list` command with tag filter

6. Notes tagged with "#assignment" are displayed.

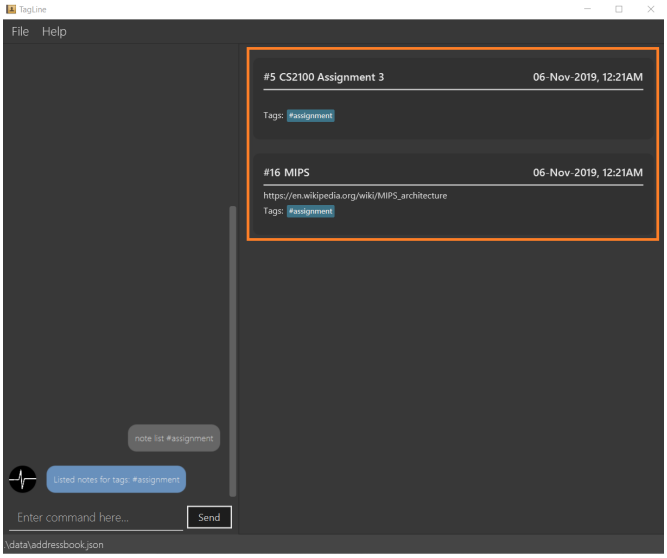


Figure 7. Filtered tagged notes displayed

7. When you would like to filter by multiple tags, you can enter the command `note list @00001 %cs2103t`.

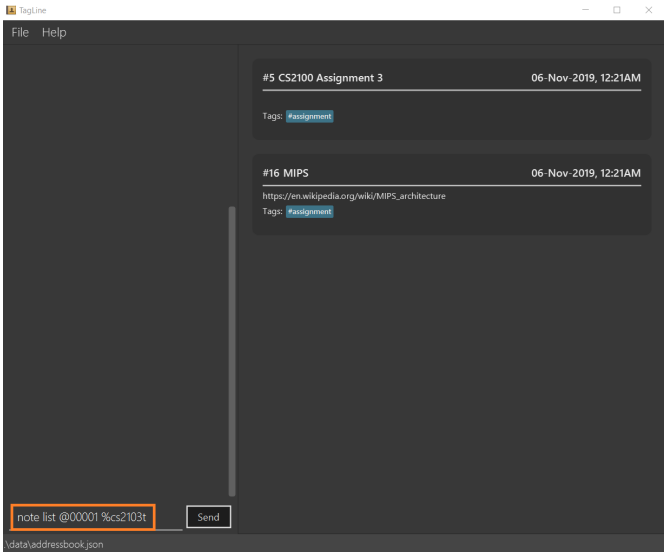


Figure 8. Entering `note list` command with multiple tag filter

8. Notes tagged with contact of contact id "1" or with group with group name "cs2103t" are displayed.

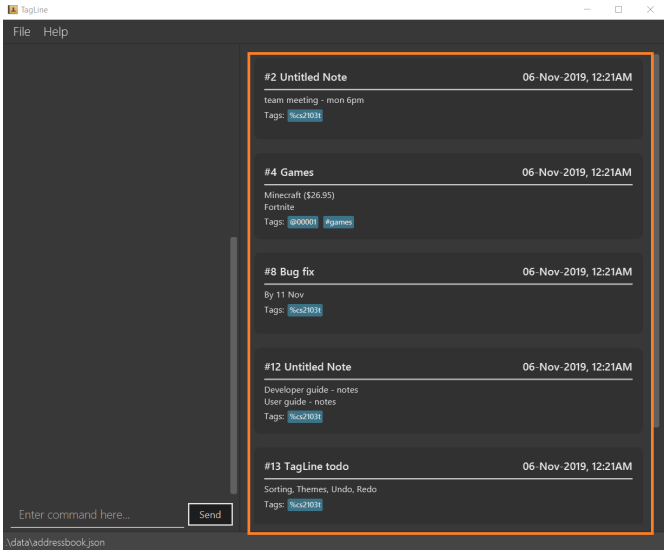


Figure 9. Filtered notes displayed

Contributions to the Developer Guide

The following section shows my additions to the TagLine Developer Guide for the `note` logic and filter feature.

Note filtering feature

Description

The user can filter notes by providing a filter in the `note list` command.

Types of filter:

- No prefix - filter by String keyword
- Prefix `#` - filter by hashtag
- Prefix `@` - filter by contact
- Prefix `%` - filter by group

Implementation

The note filter mechanism is facilitated by the `NoteFilter` class. It contains the filter value and the enum `FilterType`.

A `NoteFilter` is generated by the `NoteFilterUtil` inner class in `ListNoteParser` and passed into `ListNoteCommand`.

`ListNoteCommand` then creates a `Predicate` based on the filter and updates the list of notes in the UI via `Model`.

Filter by String keyword

Filter by keyword is facilitated by the following classes:

- `KeywordFilter` - implementation of `NoteFilter` that is passed into `ListNoteCommand`
- `NoteContainsKeywordsPredicate` - `Predicate` passed into `Model#updateFilteredNoteList()` to list only notes that contain the keywords.

Given below is an example scenario where the user enters a command to filter notes by keywords.

Step 1: The user command is passed through the `LogicManager` to `ListNoteParser`. `ListNoteParser` checks the input arguments and identify the String keywords.

The keywords are passed into `NoteFilterUtil#generateKeywordFilter()` which returns a `KeywordFilter` containing the keywords and `FilterType.KEYWORD`.

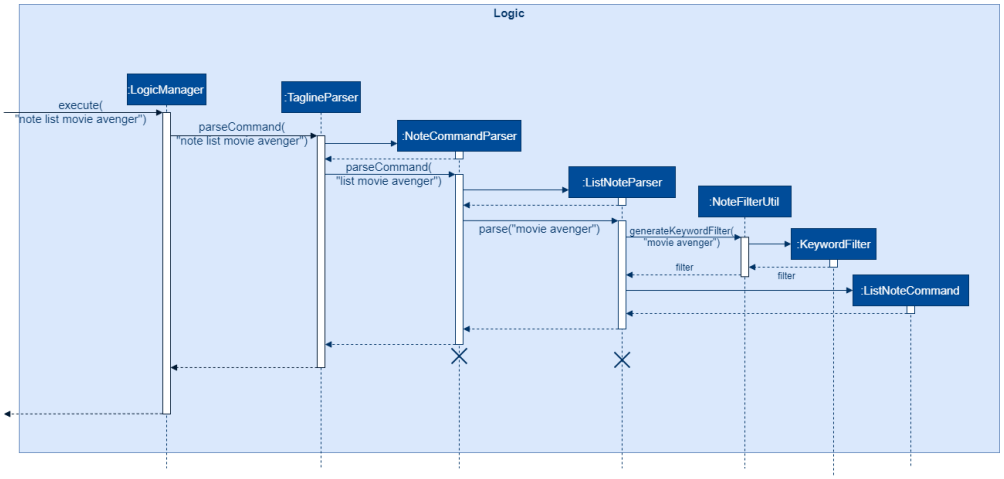


Figure 10. Sequence diagram of parsing `note list` user command to obtain a `ListNoteCommand`

Step 2: The `ListNoteCommand` returned will be executed by the `LogicManager`. If a `NoteFilter` exists and is of `FilterType.KEYWORD`, `ListNoteCommand#filterAndListByKeyword()` will be called.

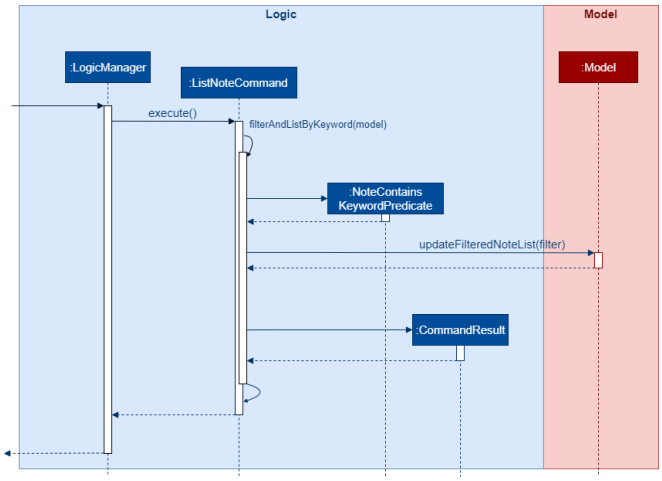
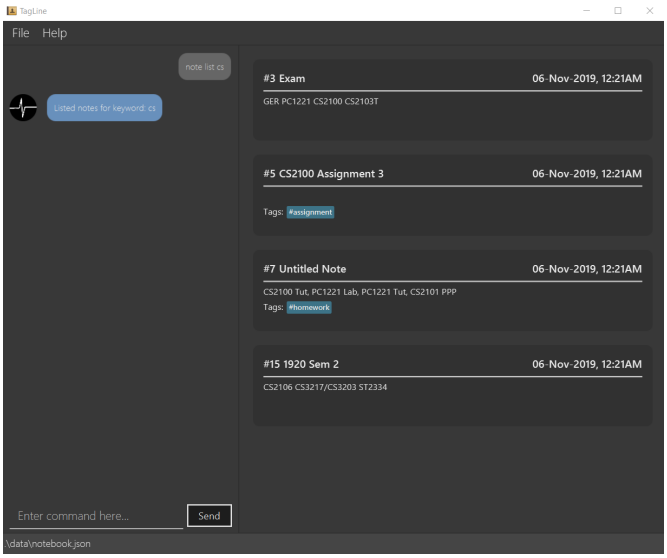


Figure 11. Sequence diagram of executing `ListNoteCommand` to update filtered note list by keyword in `Model`

The method will create a `NoteContainsKeywordsPredicate` and update the list of notes to be displayed via `Model#updateFilteredNoteList()`.



Filter by Tag

Filter by `Tag` is facilitated by the following classes/methods:

- `TagParserUtil#parseTag()` - to obtain the `Tag` objects from the user input tag strings
- `TagFilter` - implementation of `NoteFilter` that is passed into `ListNoteCommand`
- `NoteContainsTagsPredicate` - `Predicate` passed into `Model#updateFilteredNoteList()` to list only notes that is tagged by specified `Tag`

Given below is an example scenario where the user enters a command to filter notes by tag.

Step 1: Similar to filtering by keyword, the user command is passed to the `ListNoteParser`. The `ListNoteParser` checks the input arguments and identify the tag strings.

The tag strings are passed into `NoteFilterUtil#generateTagFilter()`. `TagParserUtil#parseTag()` is called to get `Tag` from the tag string. `TagFilter` containing the list of tags and `FilterType.TAG` is returned.

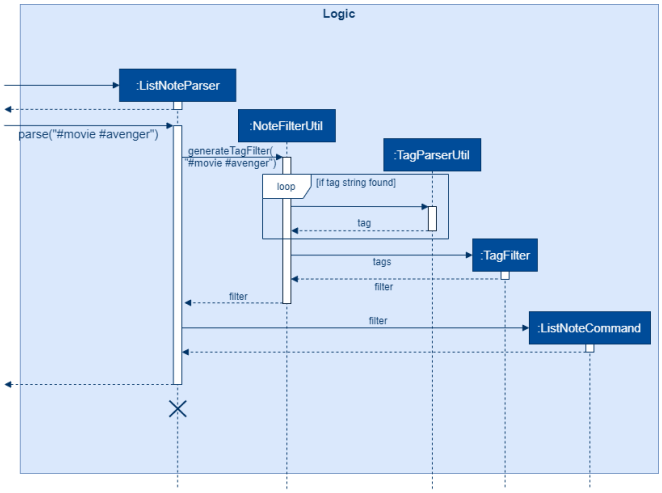


Figure 12. Sequence diagram of parsing user input tag strings to obtain a `ListNoteCommand`

Step 2: The `ListNoteCommand` returned will be executed by the `LogicManager`. If a `NoteFilter` exists and is of `FilterType.TAG`, `ListNoteCommand#filterAndListByTag()` will be called.

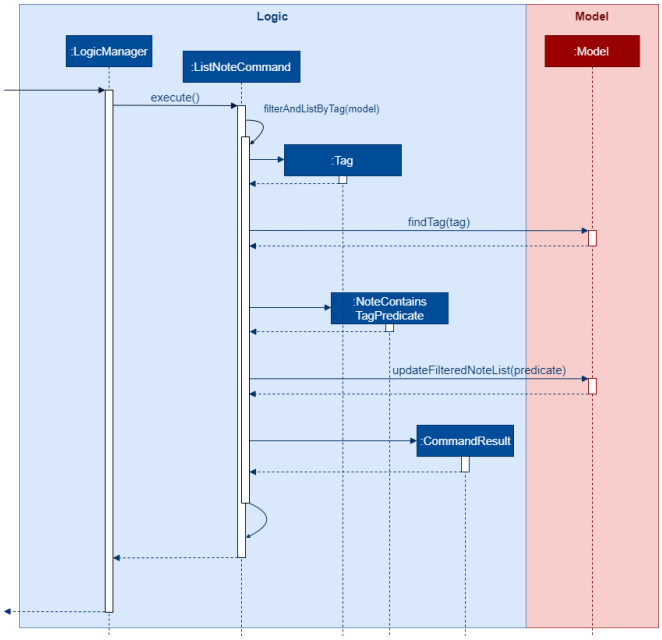


Figure 13. Sequence diagram of executing `ListNoteCommand` to update filtered note list by `Tag` in `Model`

The method will check if the tags in the `NoteFilter` exists via `Model#findTag()`. If a `Tag` does not exist, an error message will be displayed.

If all tags exist, the tags will be passed into the `NoteContainsTagsPredicate` and update the list of notes to be displayed via `Model#updateFilteredNoteList()`.