TeoShyanJie - Project Portfolio

[github] [linkedin]

Hello! My name is Teo Shyan Jie. I am currently a Year 3 Computer Science Major at National University of Singapore.

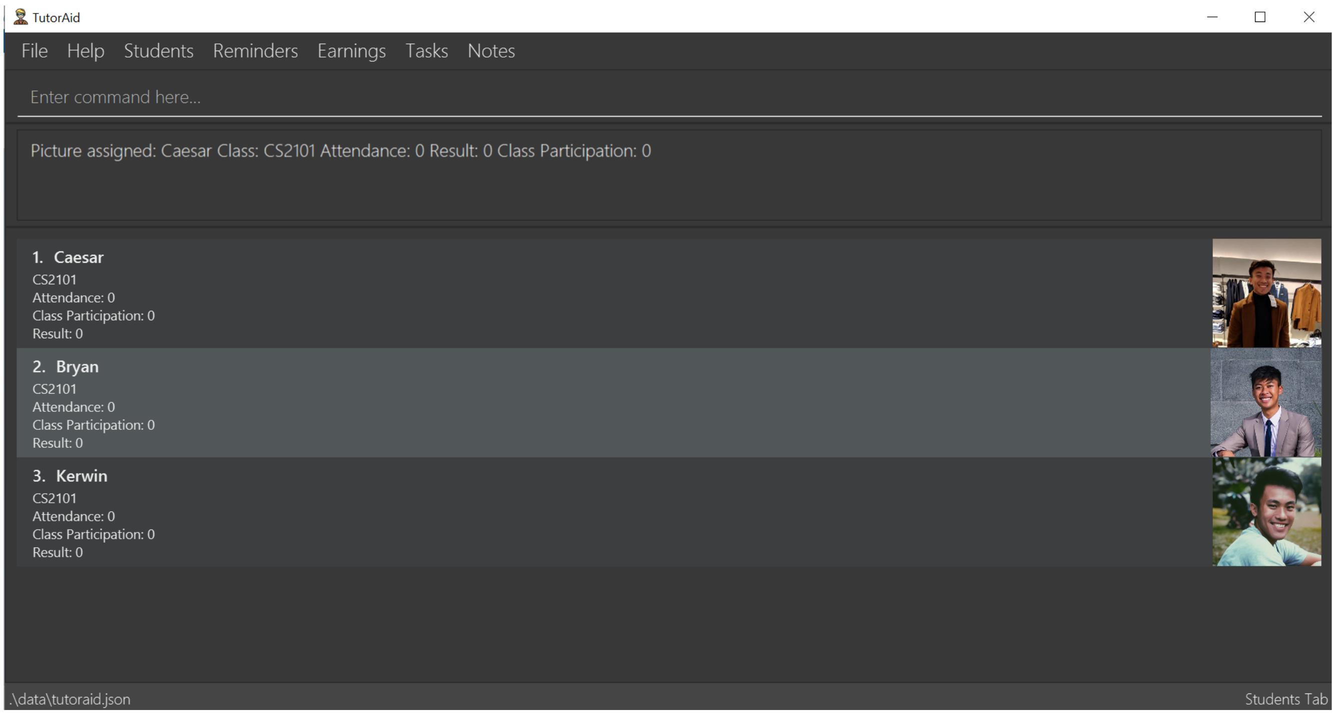
This project portfolio aims to document the contribution that I have made during the development of TutorAid. TuturAids is a project my team and I have developed throughout our studies in CS2103T module. TutorAid is one of the great software that is developed with great amount of benefits and I hope that teaching assistant can enjoy the benefits that this software brings and aids them along their journey in teaching.

PROJECT: TutorAid

Overview

TutorAid is a software build by a team of 5 Computer Science Software Engineering Student. I were tasked to implement the features of Notes for our software engineering project. We chose to morph the addressbook 3 to a teaching assistant assisting software which is called TutorAid. TutorAid is an imporved version of addressbook 3 which allows teaching assistant record student attandence, module management, earning management and notes taking.

This is how TutorAid looks like:



Picture assigned: Caesar Class: CS2101 Attendance: 0 Result: 0 Class Participation: 0

1. Caesar
CS2101
Attendance: 0
Class Participation: 0
Result: 0

2. Bryan
CS2101
Attendance: 0
Class Participation: 0
Result: 0

3. Kerwin
CS2101
Attendance: 0
Class Participation: 0
Result: 0

.\data\tutoraid.json

Students Tab

Figure 1. TutorAid Graphical User Interface

Role

My main role was to design the `Notes` features which allows teaching assistant to record their notes based on the module and the type of class they are teaching. Notes consists a variety of functions which allow great flexibility for teaching assitant which is `addnote`, `editnote`, `deletenote`, `findnote` and also `listnote`. The list of functions allows teaching assistant to create note, make an amendment from the note, delete the note or find the desire the note from the list of notes. Besides, I also wrote all sort of test cases to ensure that all the code that I wrote has reached the expected quality and reduce the chance of errors in the code.

Summary of contributions

This sections will show my contribution to the team project which consists of my code, documentation and other helpful informations.

Enhancement added: Added the `Notes` features with `addnote`, `editnote`, `deletenote`, `findnote` and `listnote` commands.

- What it does: allow user to `addnote` the desire notes, `editnote` to make amend to the note that want to change, `deletenote` the desire note, `findnote` the specific note from the list of notes and last but not least `listnote` down all the notes that previously have save in TutorAid.
- Justification: Notes is a very important features in TutorAid as it provides a great flexibility to teaching assistant. With the `addnote` functions, teaching assistant will be able to add the note based on the module he/she teaches and the type of class that they teach with any content of the note. However, when mistake is made on the note amendment is needed. Hence, `editnote` functions is added which allow teaching assistant to edit the note either on module, class type or the content. Therefore, there is no need to worry about getting worry of typing error while

taking note. Besides, `deletenote` function allow teaching assistant to remove the notes that are unwanted. When the notes save getting more, it is very difficult to find the notes that desire. Hence `findnote` function is implemented to aid this problem. It allows to search for any related keywords such as the module code, class type or the content itself. Last but not least, `listnote` function. This is significantly important as after the searching of notes, viewing the list of notes allow teaching assistant to check all the notes that are save.

- Credits: The Note features is inspired by AddressBook 3 and 4, however, notes features has a great complexity when come to implementation.

Minor Enhancement: Delete button is added to allow user to delete note easily.

Code contributed: [All Commits] [Project Code Dashboard]

Other contributions

- Project Management:
  - Manage release `v1.3.1` to `v1.4` (2 release) on Github.
  - Manage to assign feature with labels and milestone: (Pull request: #50, #100, #180)
  - Ensure that all test is pass then merge to the team repository: (Pull request: #322, #344)
- Enhancements to existing features:
  - Implement variety of test cases for features which increase coverage from 44% to 50%: (Pull request: #343, #383, #393)
- Documentation:
  - Made improvement to Developer Guide with diagram and details issues: (Pull request: #243)
  - Improve UserGuide to allow reader to have a clearer understanding on how does the software works: (Pull request: #212, #387)
- Community:
  - Review and provide feedback to each of the team members commits: (Pull request: #216, #337)
- Tools:
  - setup Netlify for team repository
  - setup Travis for team repository

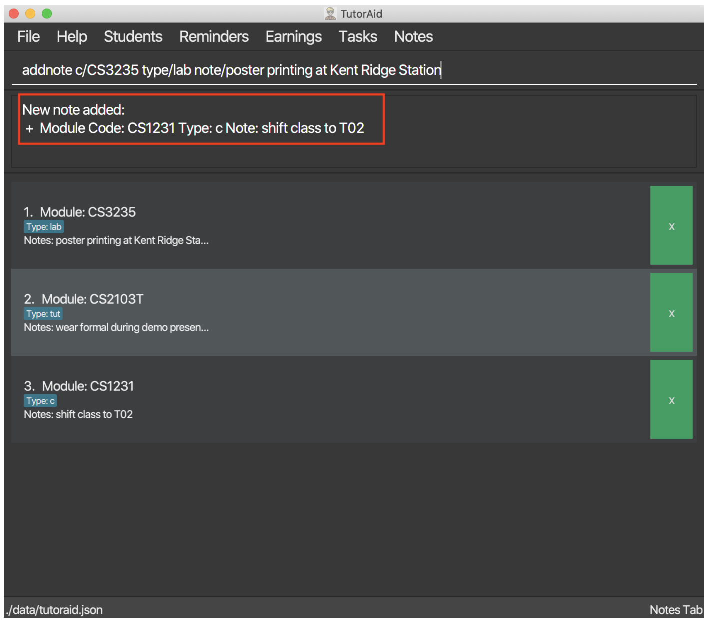
Contributions to User Guide

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

Notes

Add Note: `addnote`

Adds Note to the list of notes.
Format: `addnote c/MODULE_CODE type/CLASS_TYPE note/NOTE_CONTENT`



addnote c/CS3235 type/lab note/poster printing at Kent Ridge Station

New note added:
+ Module Code: CS1231 Type: c Note: shift class to T02

1. Module: CS3235
Type: lab
Notes: poster printing at Kent Ridge Sta...

2. Module: CS2103T
Type: tut
Notes: wear formal during demo presen...

3. Module: CS1231
Type: c
Notes: shift class to T02

./data/tutoraid.json
Notes Tab

Figure 2. Add Note Example

Examples:

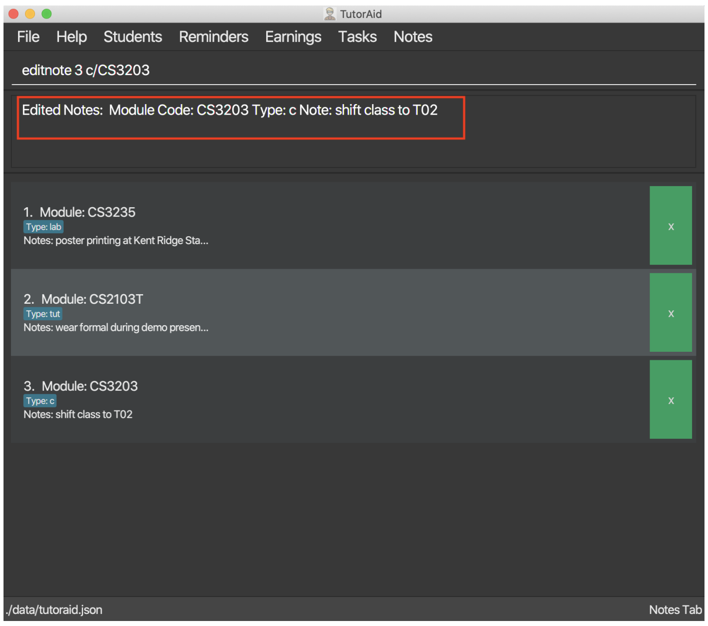
- `addnote c/CS2103T type/lab note/Check for project submission date`

NOTE Only `tutorials`/ `tut` / `lab` / `consultations` / `c` / `sectionals` / `s` arguments are allowed for `TYPE`.

Edit Note: `editnote`

To provide a great flexibility, editing of the notes is allowed.

Update any Note in the list of notes.
Format: `editnote INDEX c/MODULE_CODE type/CLASS_TYPE note/NOTE_CONTENT`



```
editnote 3 c/CS3203
```

```
Edited Notes:  Module Code: CS3203 Type: c Note: shift class to T02
```

1. Module: CS3235
Type: lab
Notes: poster printing at Kent Ridge Sta...

2. Module: CS2103T
Type: tut
Notes: wear formal during demo presen...

3. Module: CS3203
Type: c
Notes: shift class to T02

./data/tutoraid.json
Notes Tab

Figure 3. Edit Note Example

Examples:

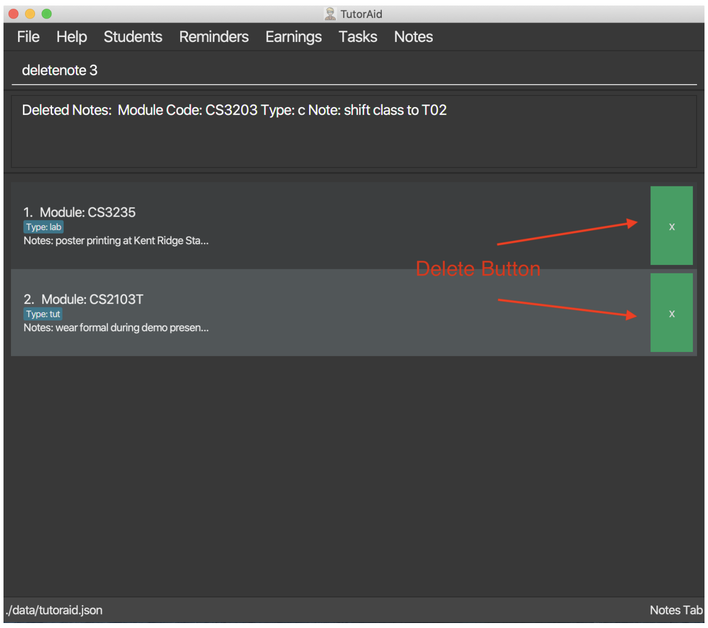
- `editnote 1 c/CS2103T`
- `editnote 2 type/lab`
- `editnote 3 note/check for meeting time`
- `editnote 1 c/CS2103 type/tut note/update project content`

Delete Note: `deletenote` / `deletebutton`

Let's say that if you would like to delete the note from the list there are two option available.

Option 1: Delete Note in the list of notes.
Format: `deletenote INDEX`

Option 2: Delete Note with DeleteButton.



```
deletenote 3
```

```
Deleted Notes:  Module Code: CS3203 Type: c Note: shift class to T02
```

1. Module: CS3235
Type: lab
Notes: poster printing at Kent Ridge Sta...

2. Module: CS2103T
Type: tut
Notes: wear formal during demo presen...

Figure 4. Delete Button Example

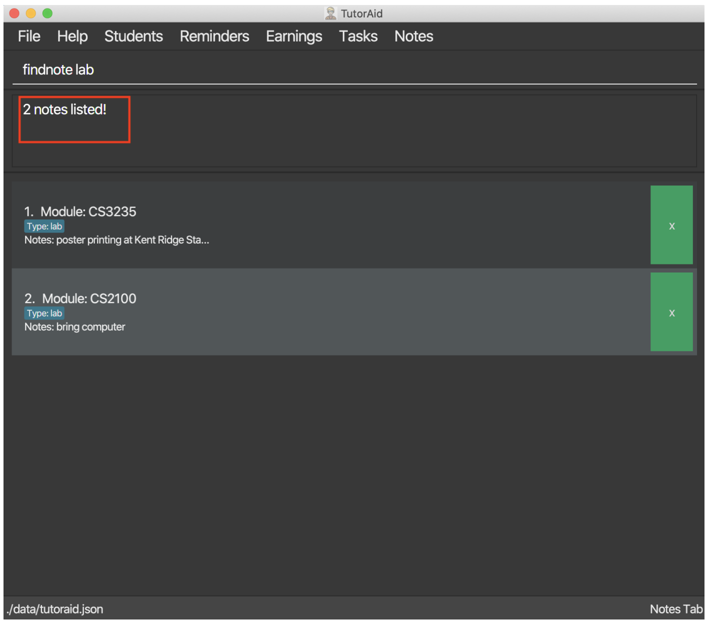
Examples:

- `deletenote 1`
- press the `x` delete button to delete the desire notes.

Find Note: `findnote`

In order to find the desire note, finding with keyword such as the module_code, class_type or note_content is allow.

Delete Note in the list of notes.
Format: `findnote KEYWORD`



```
findnote lab
```

```
2 notes listed!
```

1. Module: CS3235
Type: lab
Notes: poster printing at Kent Ridge Sta...

2. Module: CS2100
Type: lab
Notes: bring computer

Figure 5. Find Note Example

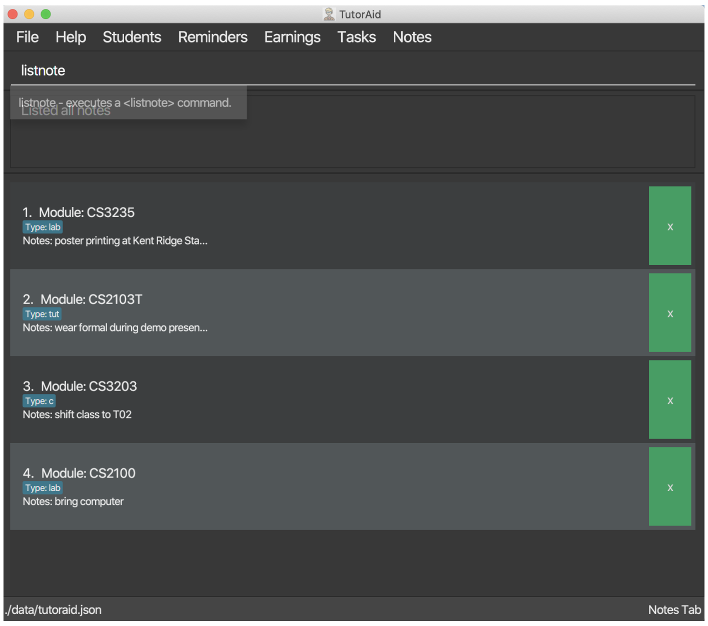
Examples:

- `findnote CS2103T`

Listing all note : `listnote`

To view the list of note. Type `listnote` to view the full list of note.

List all note.
Format: `listnote`



listnote

listnote - executes a <listnote> command.
Listed all notes

1. Module: CS3235
Type: lab
Notes: poster printing at Kent Ridge Sta...

2. Module: CS2103T
Type: tut
Notes: wear formal during demo presen...

3. Module: CS3203
Type: c
Notes: shift class to T02

4. Module: CS2100
Type: lab
Notes: bring computer

./data/tutoraid.json
Notes Tab

Figure 6. List Note Example

Contributions to Developer Guide

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

Notes Features

Add Notes

The `addnote` command allows for tutors to add their notes into TutorAid.

The format for the `addnote` command is as follows:

```
addnote c/<MODULE_CODE> type/<CLASS_TYPE> note/<NOTE_CONTENT>
```

Overview

The add claim `addnotes` mechanism is facilitated by `AddNotesCommand` and `AddNotesCommandParser`, taking in the following input from the user: `Module_Code`, `Class_Type` and `Note_Content`, which will construct `Notes` objects.

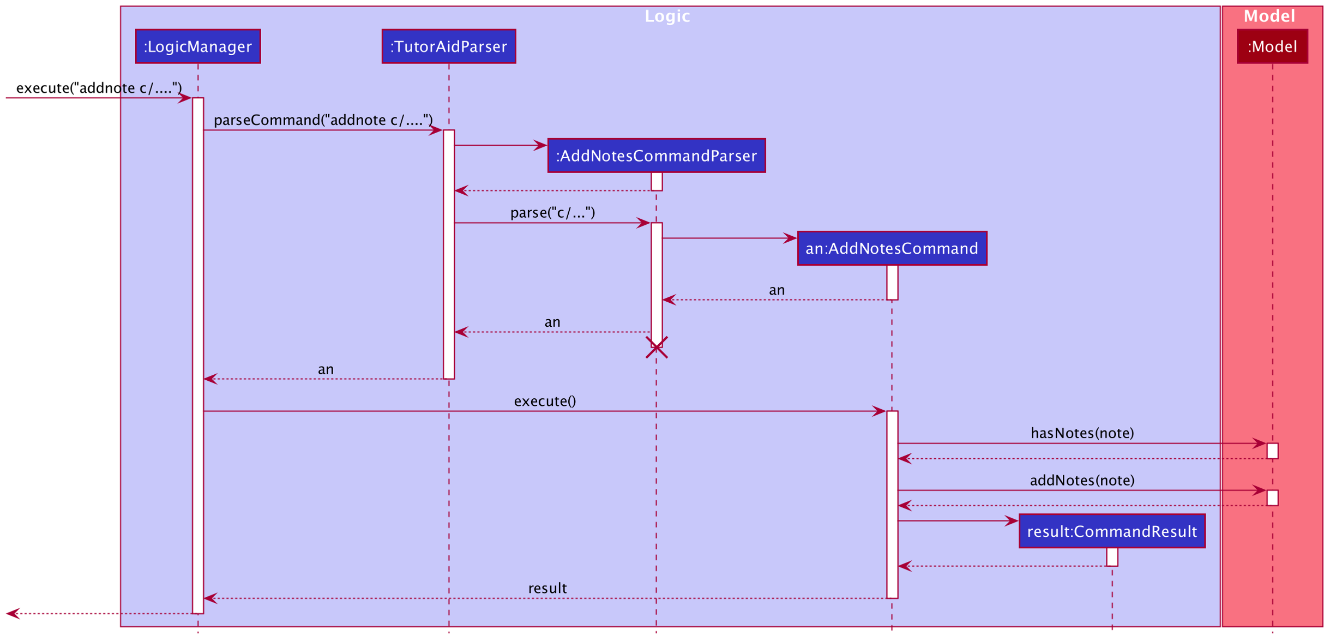
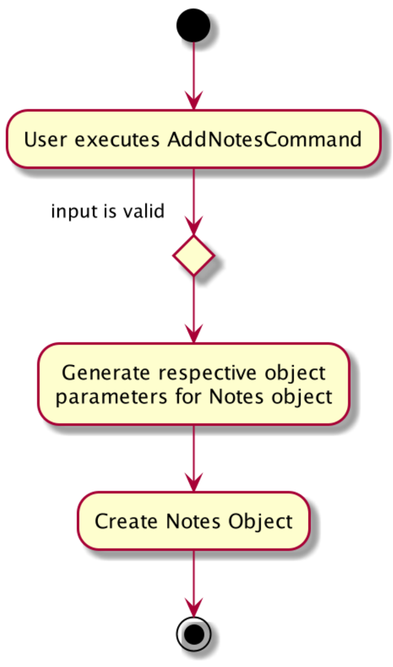


Figure 7. Add Note Command Sequence Diagram

The `AddNotesCommand` implements `Parser` with the following operation:

- `AddNotesCommandParser#parse()` - This operation will take in a `String` input from the user and create individual objects based on the prefixes `c/`, `type/` and `note/`. The `String` value after the respective prefixes will create the respective objects. A validation check will be done to ensure that the strings that are entered by the user is entered correctly. If any of the strings entered are invalid, an error will be shown to the user to enter the correct format of the respective objects.
  - `c` would use `ParserUtil#parseClassId()` to ensure that the module code entered by the user is in the correct format of CSXXXX.
  - `type` would use `ParserUtil#parseClassType()` to ensure that the class typed input by the user is in the correct format of CSXXXX.
  - `note` would use `ParserUtil#parseContent()` to ensure that the content typed in by the user is in not empty.
- After validation of the individual objects, a `Notes` object would be created with the parameters `code`, `type` and `content`.
- `AddNotesCommandParser` would then return a `AddNotesCommand` object with the parameter, `Notes` object.

The following activity diagram summarizes what happens when a user executes a new command.


User executes AddNotesCommand
input is valid
Generate respective object parameters for Notes object
Create Notes Object

Example Scenerio

- Step 1: The user enters `addnote c/CS2103T type/tut note/Update Project` to add a note for teaching classes. This adds a `Notes` object that the user has added to record what needs to be done for the class.
- Step 2: `LogicManager` would use `TutorAidParser#parse()` to parse input from the user.
- Step 3: `TutorAidParser` would match the command word given by the user with the correct command. In this example, the given command is `addnote`, thus, `AddNotesCommandParser` object would be created with the user's input.
- Step 4: `AddNotesCommandParser` performs a validation check on each of the respective objects through `AddNotesCommandParser#parse()`. In this case, it would use `ParserUtil#parseClassId()`, `ParserUtil#parserClassType()` and `ParserUtil#parseContent()`. It would then return a `AddNotesCommand` object with an `Notes` object.
- Step 5: `LogicManager` would execute `AddNotesCommand#execute`. In this particular method, the `Notes` object will be check with the rest of the prior `Notes` object, to ensure that there is no duplicate `Notes` object. If there are no similar `Notes` object with the same parameters created previously, it would then be added into the notes list.
- Step 6: `AddNotesCommand` would then return a `CommandResult` to `LogicManager`, which would show the user that the new `Notes` object have been successfully added.