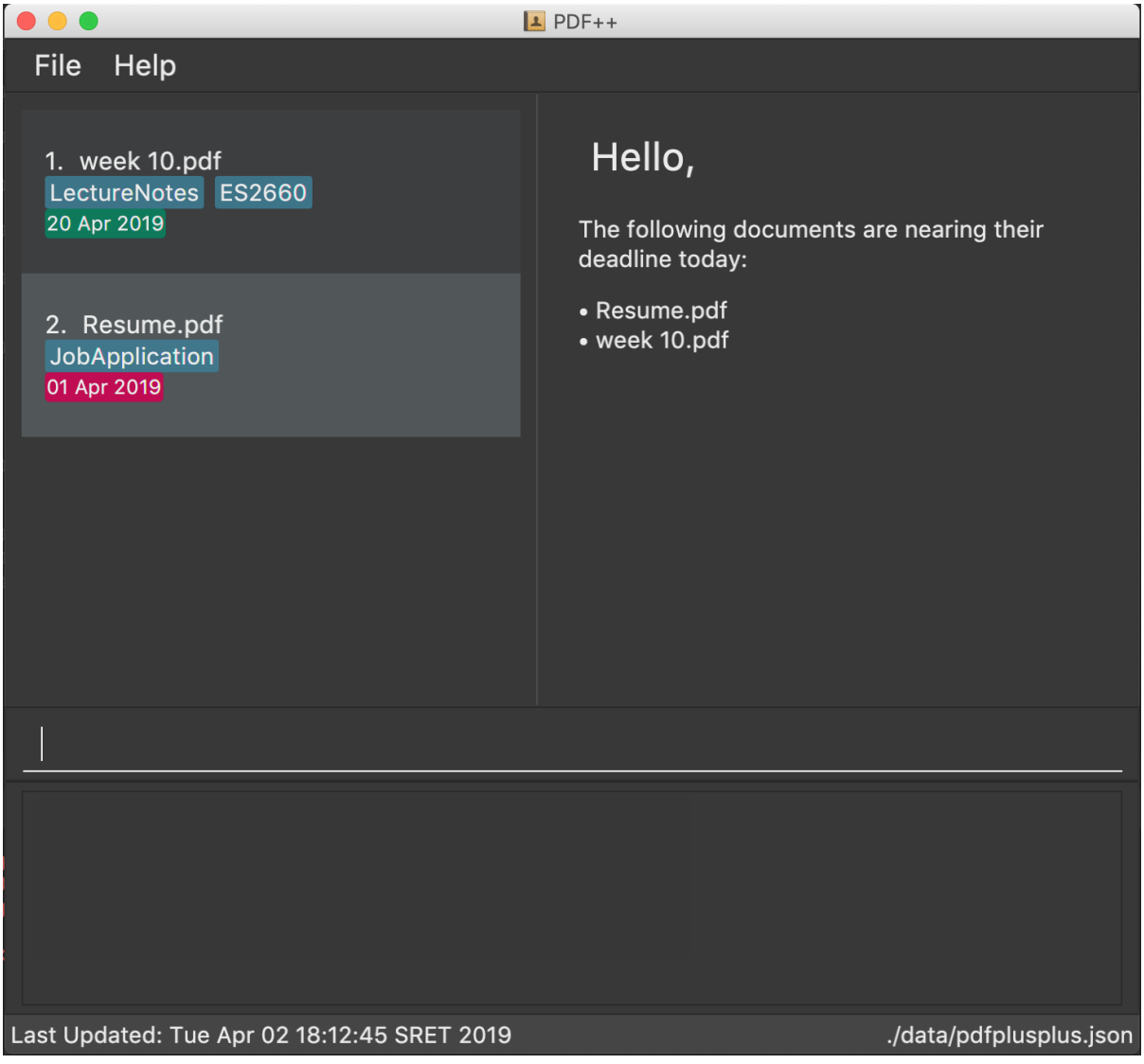
Jeremy Low - Portfolio for *PDF++*

The Project

Our team was tasked with enhancing this addressbook for our Software Engineering Project. We made the decision to morph it into a personalized file manager, ***PDF++***. Inspired by applications that bring an upgrade to the atypical solution like ***Notepad++***, we wanting to make an application relevant for students. This application allows you to set datelines and tags for files, with managing PDFs as its main specialty. It is mainly a CLI (Command Line Interface) application i.e. commands are executed through typing the desired command in the input line provided. However, several commands also allow for input through a GUI (Graphical User Interface), where the user is able to click on the desired input.



Note the following symbols and formatting used in this document:

`Grey highlights` (called mark-ups) indicate that this is a command that can be executed by the application.

Summary of Contributions

The following section details a summarised overview of my contributions to the team project in the areas of coding, documentation and design.

Enhancements added:

- `merge` command
  - Description: I added the ability to append two of more PDF files within the application to so that a new file is created with the merged content.
  - Justification: Merging PDF files is highly utilised, especially by students or users handling numerous documentations. However, merging of PDF files is usually a service provided by paid versions of PDF managers or other external online services.
  - Highlight: Merging can be done quickly through the application, without the inconvenience of relying on other software and online services to do it.
  - Credits: *Apache PDFBox® library* (*PDFMergerUtility*, *PDDocument*)
- `find` command optimisation
  - Description: Based on the originial ability of addressbook to find files based on name fully matching search keyword, I added the ability to find files based on partial matching of file name, as well as matching of any text content within the file.
  - Justification: ***PDF***++ is intended to be complete replacement for managing files. For finding a file on most other filesystems, the full name of the file is not required, hence it should also be made the same for the application. Additionally, users should be able to find for files based on the content, which can be helpful in refine the scope of the search and make it more accurate.
  - Highlight: Similar to a Google search engine, finding of files becomes faster!
  - Credits: *Apache PDFBox® library* (*PDDocument*, *PDFTextStripper*)
- `move` command
  - Description: I added the ability to move files within the application to any permissible directory within the user's local device.
  - Justification: ***PDF***++ is intended to be complete replacement for managing files. The user should be able to perform all the desired actions regarding file management through the application.
  - Highlight: When moving a file to a deeply nested directory, it is much more convenient as compared to manually navigating to the directory to move the file.

- `filter` command
  - Description: I added the ability to filter files within the application based on the file tag(s) specified.
  - Justification: For files that are tagged, it is necessary to have a means to use the tags as identification - if not then tags would be purely aesthetic!
  - Highlight: Filtering files by tag allows you to group files strategically within the application, which would make file management much more convenient.
- `delete` command optimisation
  - Description: Based on the original ability of addressbook to delete files recorded within the application (but not actually deleting he file from the local filesystem), I upgraded it so that there is an option for the user to delete the file both from the application as well as from the local filesystem.
  - Justification: ***PDF***++ is intended to be complete replacement for managing files. From a user perspective, the intention behind deleting a file might be to remove the file completely. Otherwise, the user might encounter additional inconvenience e.g. deleting a file to create a new file of the same name, only to be unable to because the old file is still physically present.
  - Highlight: There is flexibility in how you want to delete a file from the application, as well as the convenience of using CLI which has the potential to be faster than clicking on the file directly.
- `sort` command optimisation
  - Description: Based on the original ability of addressbook to sort the files based on ascending order of name, I upgraded it so that the user can specify if the sort should be done in ascending or descending order.
  - Justification: It is an implicit requirement that sorting features should have the flexibility for sorting to be done in either ascending or descending order.
  - Highlight: Sorting ability becomes more robust and gives the user the flexbility to sort as desired.

Code contributions:

Please click on any of these links to see samples of my code:
(v1.4) Merge command
(v1.3) Upgrade find command
(v1.3) Filter command

Documentation / Design contributions:

- Developer Guide
  - Added detailed documentation for the following sections under **Implementation** to give a comprehensive view of the feature design, execution process, as well as any additional design considerations:
    - *Merge Feature*
    - *Move Feature*
    - *Open Feature*
    - *Delete Feature*
  - Created UML Activity Diagrams for the following sections under **Implementation**:
    - *Merge Feature*
    - *Add Feature*
    - *Rename Feature*
    - *Encrypt Feature*
    - *Decrypt Feature*
    - *Clear Feature*
  - Created UML Sequence Diagram for the *Merge Feature* section under **Implementation**.
- User Guide
  - Added a step by step walkthrough to guide the user through merging of files. The walkthrough includes detailed instructions as well as visual aids of the application interface.

Minor contributions:

- Set up RepoSense configuration for the team on our GitHub repository.
- Integrated a third-party library *Apache PDFBox®* to the project which allowed for additional features to be performed on PDF files (e.g. reading of content, merging multiple files).

Contributions to Developer Guide

The following sections are excerpts from my additions to the ***PDF++* Developer Guide** (continued next page):

Merge Feature

Current Implementation

The `merge` feature is facilitated by both **MergeCommand** and **MergeCommandParser**. This feature utilises the *Apache PDFBox® library*, specifically the *PDFMergerUtility* API to append two or more PDFs and create a new file with the merged content. As there will be one additional file added to the application, this feature also implicitly performs the `add` feature to add the new PDF to the application.

The implementation of the **MergeCommand** execution can be summarised in the following activity diagram:

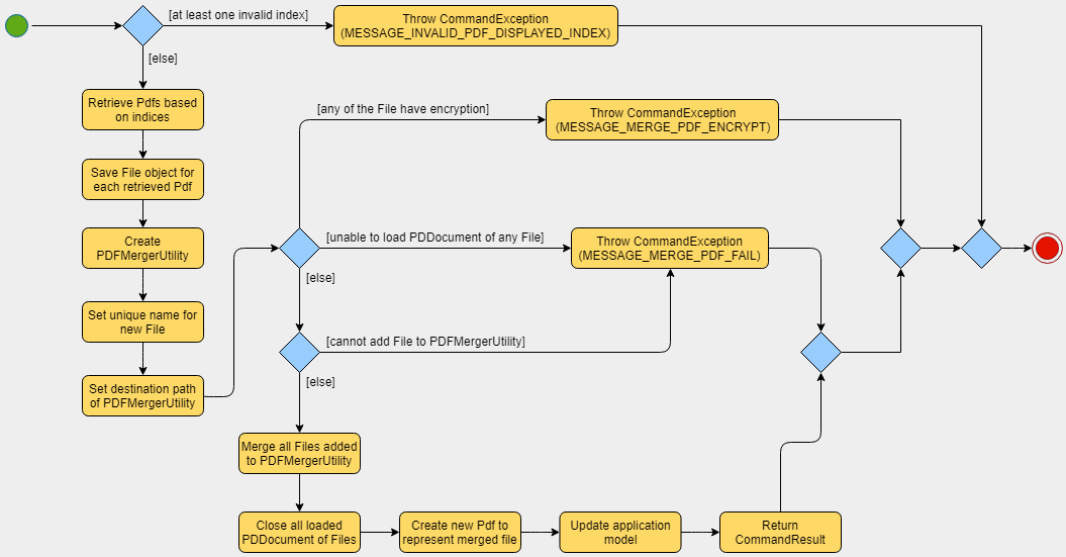


Figure 1. Merge Command Activity Diagram

1. The provided indices are checked to be valid i.e. referring to a specific Pdf in the PdfBook.
    a. If there is at least one invalid index, a **CommandException** will be thrown and the execution will be ended.
2. The required Pdfs are retrieved from the PdfBook based on indices.
3. A File object is created for each Pdf which allows file operations to be performed on the Pdf.
4. *PDFMergerUtility* is created. The directory must be set for the merged file prior to merging, which also includes the name of the file. By default, the directory of the Pdf based on first index is used.
5. A unique name is created for the merged file and combined with the aforementioned directory to create the full directory for the merged file.
    a. To avoid issues with duplicate name at the directory, the name is created based on hash code. The created name will also be verified unique at the directory - and changed if necessary.
6. The Files will be loaded as *PDDocument*, which is an indicator if the application can perform other operations on the Pdf that need it to be handled as a `.pdf` file.
    a. Errors in accessing Pdf would throw **IOException**. Errors would most likely be due to:

i. File not found at location

ii. Lack of user permissions to open file

iii. File has encryption

iv. File corruption

b. Thrown **IOException** is intercepted, a **CommandException** will be thrown and the execution will be ended.

7. The Files are added to the PDFMergerUtility.

a. Errors in adding to PDFMergerUtility would throw **IOException**. The cause for error would be similar to above.

b. Thrown **IOException** is intercepted, a **CommandException** will be thrown and the execution will be ended.

8. All loaded **PDDocument** are closed.

9. A new Pdf is created to represented the merged file created.

10. The Pdf is recorded in the Model and the changes are committed.

11. **CommandResult** is returned upon successful execution.

This sequence diagram demonstrates the interactions involved from start of **MergeCommandParser** to end of **MergeCommand** execution:

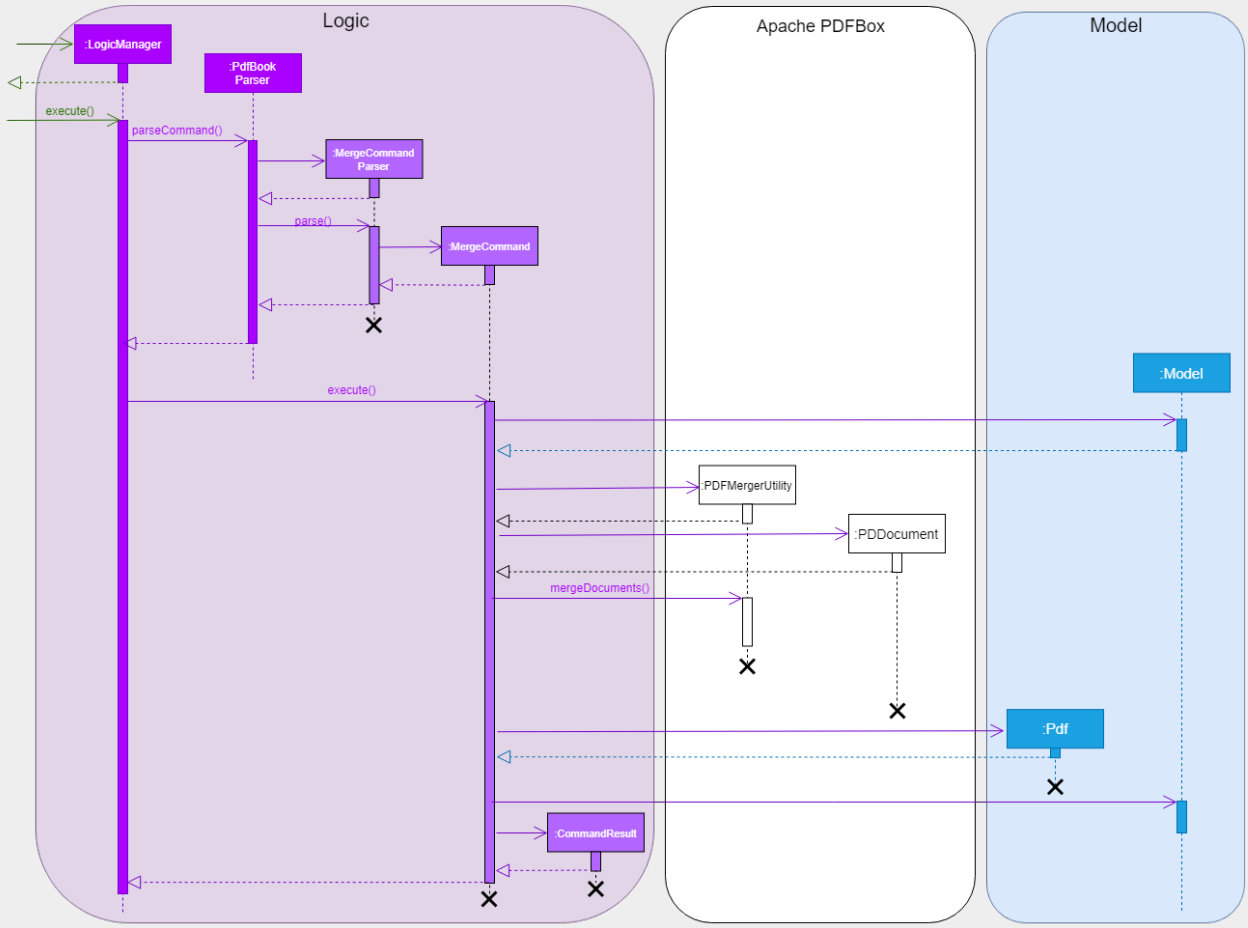


Figure 2. Merge Command Sequence Diagram

Considerations

The default directory of the merged file is currently set to the directory of the first index of the files to be merged. There were considerations to make flexibility in the merge command input to allow for the user to specify the desired directory of the merged file. As the current version of ***PDF***++ is focused on a working product, it was decided to simplify the command to focus on the merge operation. Users can also make use of `move` feature to move the file; such implementation is more intuitive if the user is going to be using the application for everyday needs.

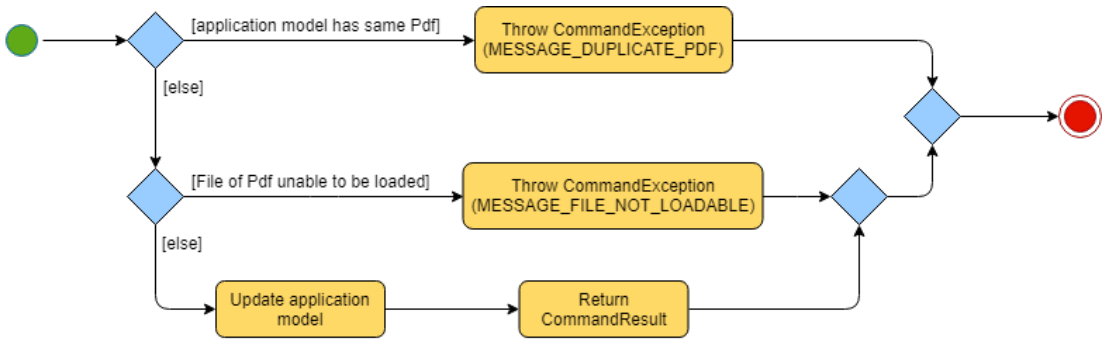
For the merging of files, the current implementation involves adding all files to a single *PDFMergerUtility* to merge together. One alternative to merging multiple files would be to create a separate **PDFMergerUtility** for every pair of files and merge the content recursively into a final merged file. The advantage of this would be better stability in performance when merging multiple large files as there will be lesser workload on each **PDFMergerUtility**. However, this would undoubtedly cause the performance to be slower as more merge operations are done overall. As the case of instability occurs only in very large files, it was decided to go with focus on performance.

Future Implementation

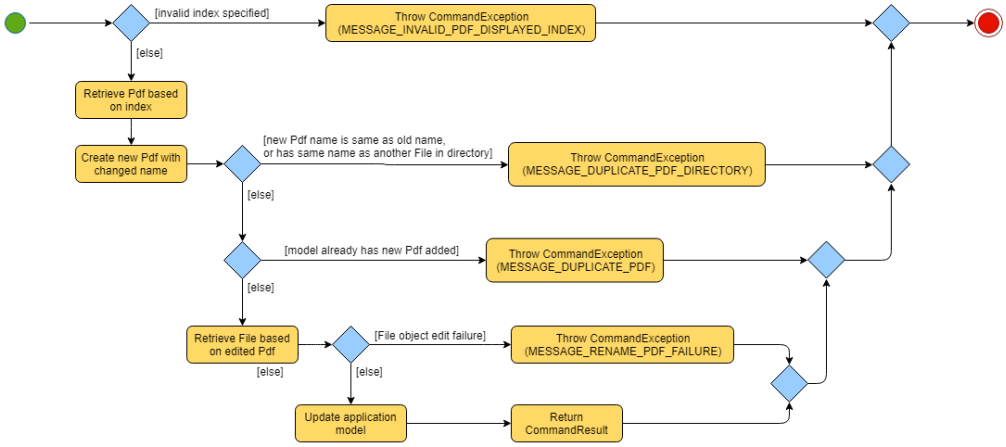
There are performance issues encountered when performing the merge operations with many files of large size. In future versions, the merge operation has to not only ensure performance but stability as well. By v2.0, the merging will be able to support larger files without any concern for the application to freeze or crash while merging. Currently, there are no means to make the merging operation perform faster due to the merging operation being performed through the *PDFMergerUtility* API.

UML Diagrams

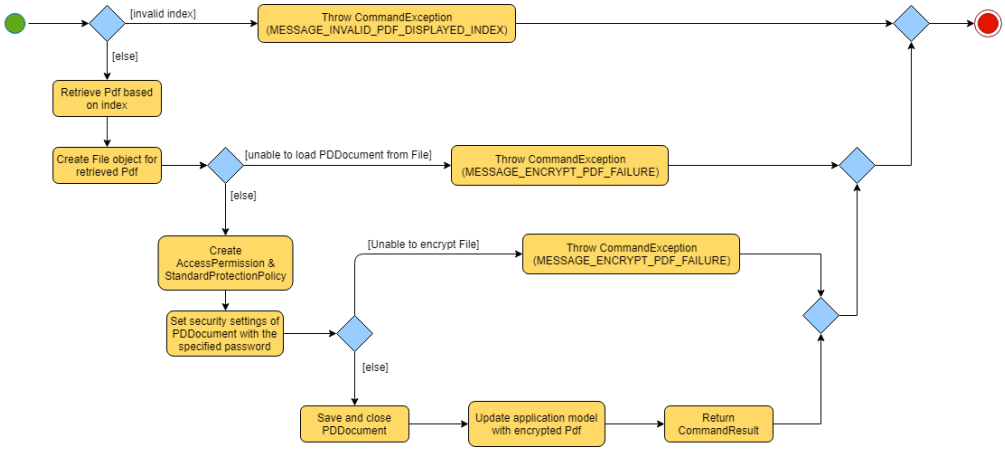
Add Command Activity Diagram



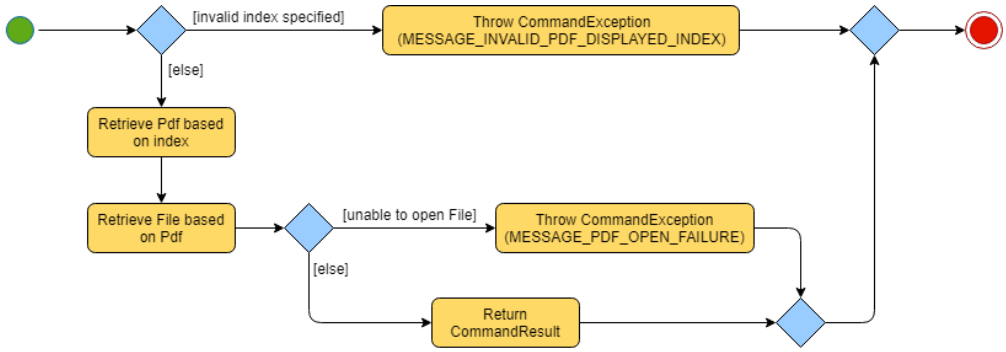
Rename Command Activity Diagram



Encrypt Command Activity Diagram



Open Command Activity Diagram



Merge Command Activity & Sequence Diagrams

The aforementioned diagrams have been included in [Documentation] above respectively under **Merge Feature**.

Contributions to User Guide

The following is an excerpt from my additions to the ***PDF++* User Guide** (continued next page):

Merging Files: `merge`

Step-By-Step Guide

Illustrated below is a sample usage scenario that provides a clear view to the inner workings of the `merge` feature.

Step 1: Launch the application by double clicking the `pdfplusplus.jar`. To view the following screen.

NOTE Files you observe may be different and depends on the actions you have previously carried out on our application

Step 2: From the main interface of the application, the user chooses the file(s) that they wish to merge, and enters the `merge` command into the **TI**, following the outlined syntax as illustrated below.

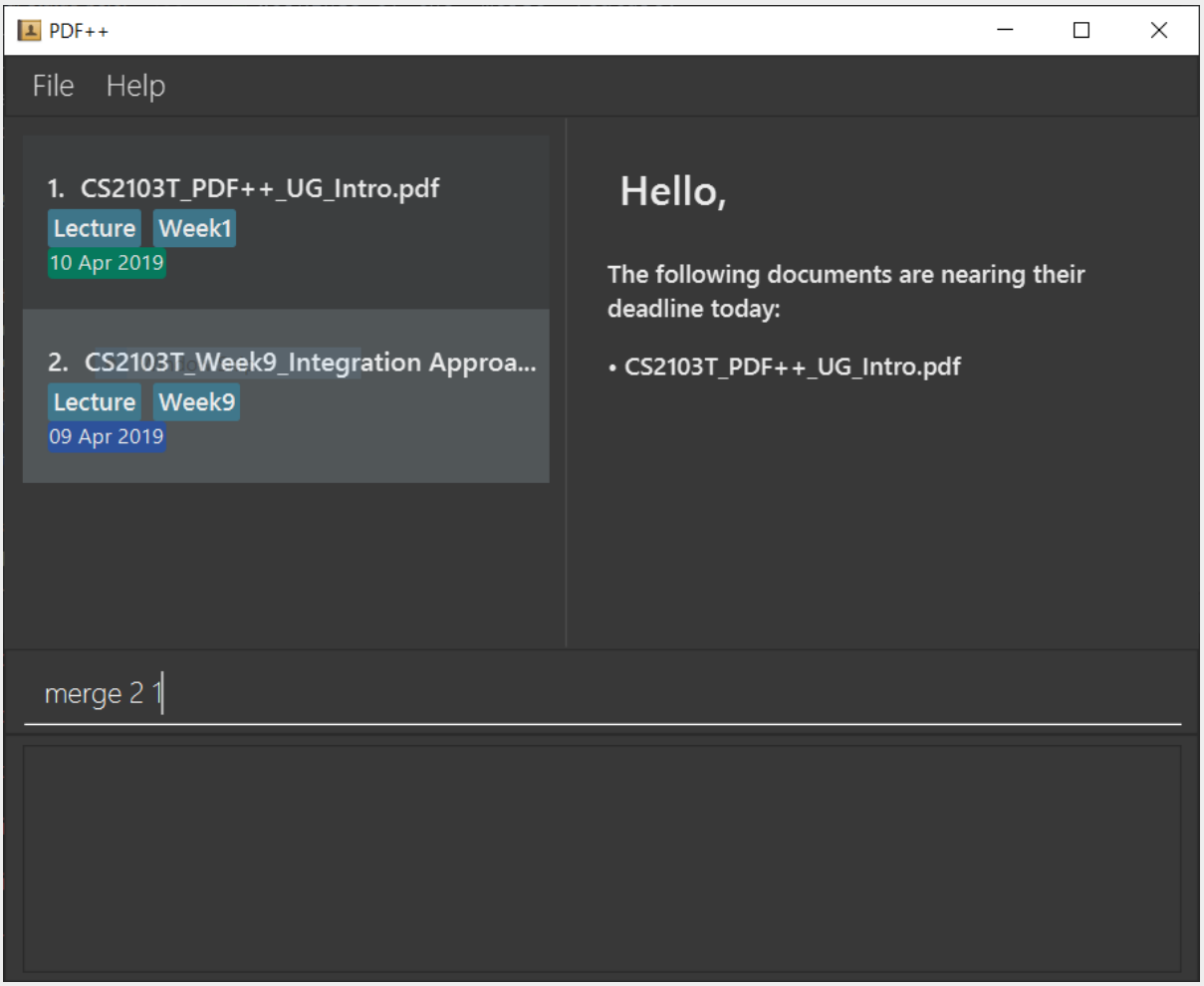


Figure 3. Merge Command Step 2

Step 3: After executing the command, the above two files will be merged, with the *"CS2103T_PDF++_UG_Intro.pdf"* file attached behind your other selected file.

Step 4: The **MergeCommand** is then executed. The new name of the merged file follows the format: "merged[hashcode].pdf", where hashcode is a random string of letters and numbers. This prevents any naming conflicts between files.