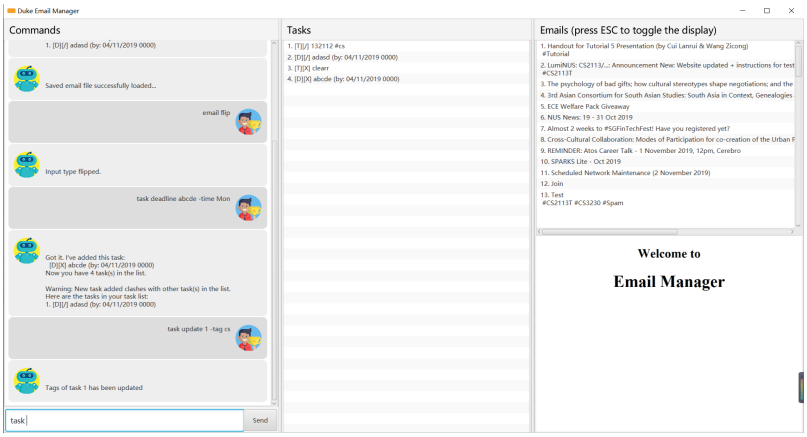
Project Portfolio Page

By: F11-3 Yan Zhiwen

Since: Oct 2019

1. Product Overview



After being constantly spammed by the school emails and missing all kinds of important emails, our group decided to develop this email manager to better organize our mailbox as a NUS Computing student.

Email Manager is an email and task manager app, specifically designed for NUS School of Computing students to manage their emails and busy schedules. As a text-based application, it is optimized for those who prefer typing and working with Command Line Interface (CLI). Email Manager also has a developed Graphical User Interface (GUI) that allows users to view email and task details in an appealing, well-organized format. If you can type fast, Email Manager can get your emails and tasks organized quickly and easily.

2. Summary of Contributions

- Code Contributed
- Feature Implemented (starting from the most significant):
  1. Email format and content(keyword) auto parsing
  2. Email keyword adding and highlighting

3. Email fuzzy search
4. Task reminder
5. Infrastructure on Email fetching over Internet, Storage, Command Parsing etc.

- Other Contributions
  1. Cross-platform and project structure of release

3. Contributions to User Guide

3.1. Quick Start

This section serves as a tutorial for a new user to Email Manager.

3.1.1. Installation

1. Ensure you have Java version 11 or later installed in your computer.
2. Download the latest email_manager.zip here and extract it to a folder.
3. If the download is too slow, please download it from google drive.

3.1.2. Run Email Manager

WARNING	The program might freeze for a while when fetching and parsing the email. Please wait patiently.

Windows User

1. Double click the jar file in the extracted folder.
2. If any error occurs, try to follow the mac/linux user instructions to run the program from command line.

3.1.3. Mac/Linux User

1. Open the command prompt.
2. Change the working directory of the command prompt to the extracted program folder using `cd` command.
3. Check java version using `java -version` command. It should be java 11.
4. Run `java -jar email_manager-1.3.jar` and you are good to go.

WARNING	Make sure that your command prompt is at the directory of the folder extracted, otherwise the data structure might not be recognised. Creating data file structure coming in v1.4

TIP	Please send an email to jokeryan1997@gmail.com if you cannot successfully run the program.

Use and Test with/without Internet

Our product can automatically parse and manage the emails in your mailbox. It is best to be used and tested with Internet connection. If you do not have Internet connection, we have also prepared some data for you to test offline.

- If you have Internet connection:
  1. You will be directed to a web page to authorize our access to your mailbox. Please login to the **dummy email** we provided to you.
  2. Close the browser and return back to the Email Manager. Wait for a while for the email to be fetched and parsed.

WARNING	Our product does not support non UTF-8 characters. Using your own mailbox for testing may lead to redundant saving and loading with the current version. More charsets will be supported in v2.0

- If you do not have Internet connection:
  1. Paste everything under `test_data` file in the program folder, to the `data` folder. Replace all the files in the `data` folder.
  2. Close the Email Manager and Start it again.
  3. Wait for a while for all the emails to be parsed.

3.1.4. Introducing the Interface

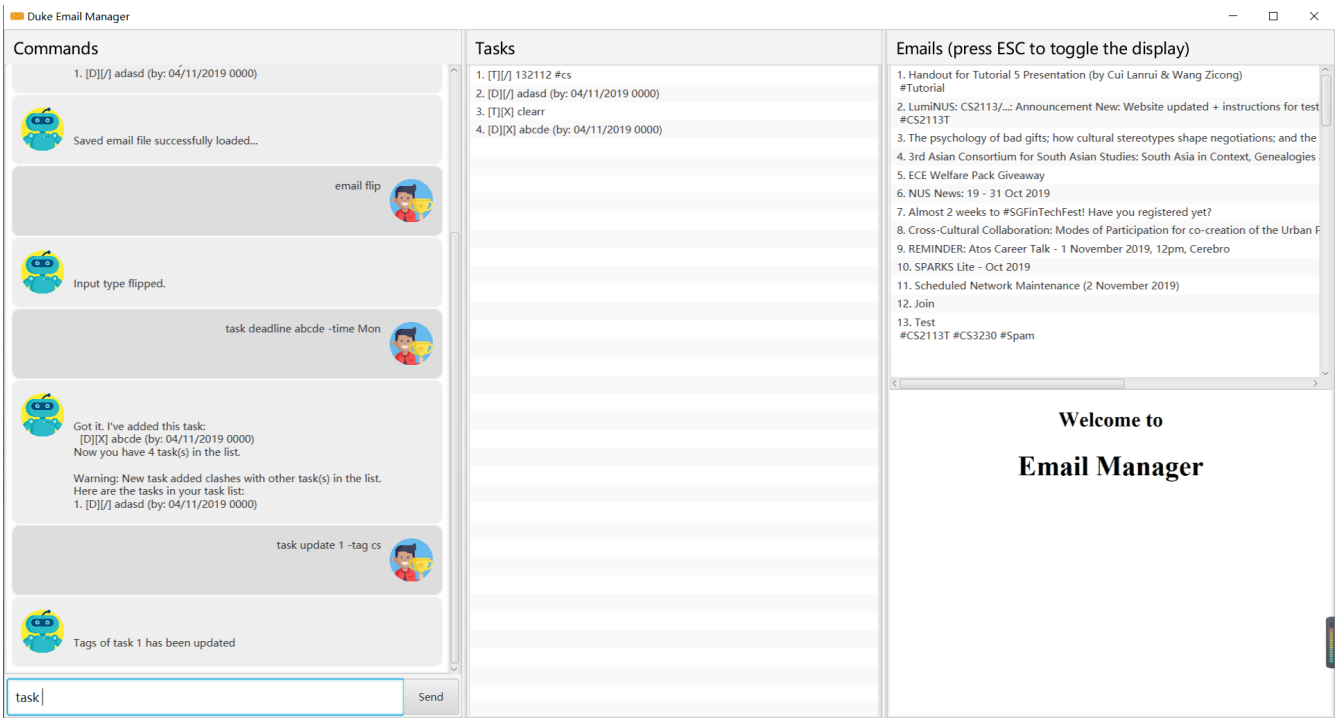


Figure 2: Main window of Email ManagerMain Window

As shown in Figure 2, the main window of Email Manager has three sections.

- On the left, it is the command interface, where you should be entering your command and get response from the Email Manager.
- In the middle, it is the task list, which displays all tasks information.
- On the right, it is the email list and content display. You can press `Esc` key on your keyboard any time to switch display between the list and content view of emails.

3.2. Email Features

The command listed here can only be used under email mode/prefix. Use `flip` command to toggle the mode.

3.2.1. Auto parsing and tagging of emails

This feature is automatically triggered by fetching emails from Outlook Server and adding new kayword. It will automatically parse "keywords" that might be shown in your emails in different forms/expressions.

For example, the keyword `CS2113T` might be in the expression of `CS2113T`, `CS2113`, `Akshay Narayan`, `Akshay` etc. All emails containing these expressions will be recognized as containing the keyword `CS2113T`. This helps the email manager to better categorize your emails. For example, some emails might not explicitly include the module code, still belong to that module as it is sent by the lecturer.

For easier usage and testing, we have prepared a few default keyword + expressions pair as shown in Figure 3, which will be loaded when the program starts.

Keyword	Expressions
CS2113T	CS2113
	TAN KIAN WEI, JASON
	Leow Wei Xiang
	Akshay Narayan
	Akshay
CS2101	CS2101
	Anita Toh Ann Lee
CG2271	CG2271
	Djordje Jevdjic
CS2102	CS2102
	Adi Yoga Sidi Prabawa
CS3230	CS3230
	Divesh Aggarwal
CEG Admin	Low Mun Bak
SEP	SEP
	Student Exchange Programme
Tutorial	Tutorial
Assignment	Assignment
Spam	UHC Wellness
	luminus-do-not-reply
	NUS Libraries

If the email contains some expressions, the respective keyword will be used to tag that email. It can be seen from the `#` sign on the email list. When showing the email using `show` command, the parsed expressions will also be highlighted in yellow.

NOTE The parsing only allows exact matching of words, which means expression `2113` will not be matched with text `CS2113`. Any extra characters before and after the word will not be accepted. This is to avoid unintentional match for short expressions. Please add more expressions if you want more matching patterns.

NOTE Some emails have pictures in it, and obviously we do not intend to capture the words in those pictures.

TIP To test this feature, you are recommended to use `addKeyword` command, or sending email to the dummy mailbox. Remember to call `fetch` command after sending and leave a few seconds for the email to be transmitted before calling the `fetch`.

3.2.2. Add Keyword: `addKeyword`

Format: `addKeyword KEYWORD -exp EXPRESSION1 [-exp EXPRESSION2]…`
Adds a keyword or expressions to the keyword list. If the keyword is already in the list, the expression will be added to the expressions belonging to that keyword. So this is a command to add both keyword and expression.

TIP	The easiest way to verify whether this command is working properly is to check whether the email containing the expression has a tag of this keyword in the email list, or just use `show ITEM_NUMBER` command to see whether the expressions are correctly highlighted.
NOTE	All emails will be parsed again upon the updates in keyword and expression, so the window might freeze for a while. Please wait patiently.

4. Contribution to Developer Guide

4.1. Email Management

4.1.1. Email Auto Parsing

The emails fetched or stored locally will be automatically parsed to extract important information for tagging, task creation and reminder purposes. The parsing consists of two stages, the **format parsing** and **content parsing**. Email format parsing is to parse the email components like subject, sender and body from the raw string fetched from the server or stored in local file. The content parsing is to parse the keyword included all components of email.

Email Format Parsing

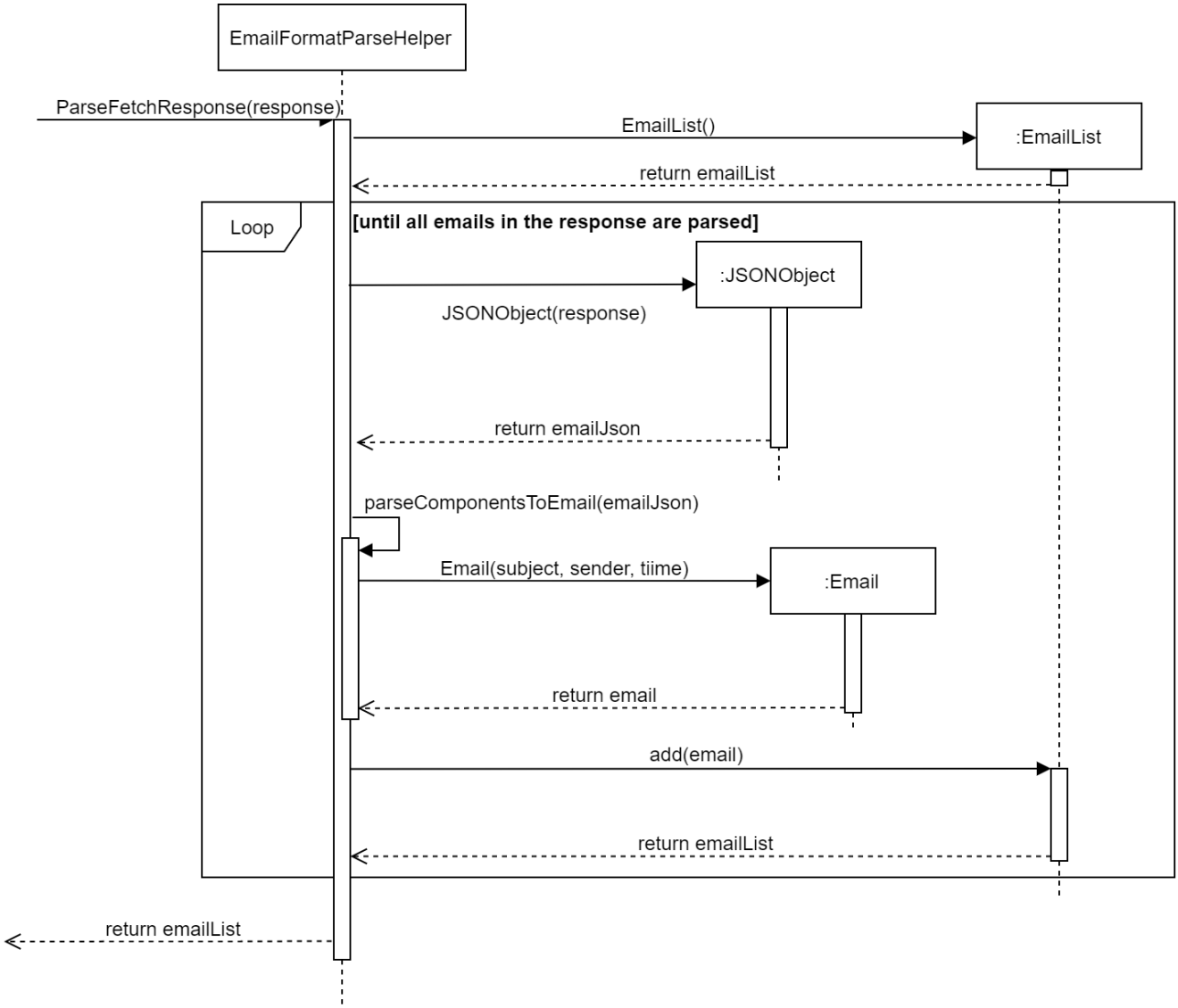


Figure N: Email Format Parsing

The email format parsing starts at the call of `ParseFetchResponse(response)`. The variable `response` here is a string of the http response from the Outlook server to the fetch API call.

It first creates an `EmailList` to store all the emails parsed from the response. This function only returns the list instead of directly adding the email parsed to the model because the storage or network component will decide whether and how the emails are to added to the model.

Then each email contained in the response of is parsed to a `JSONObject` called `emailJson` for easier manipulation.

With this `emailJson`, `parseComponentsToEmail(emailJson)` is called to extract different components of the json and instantiate an `Email` object to be added to the `emailList` created earlier.

This process repeats until all the email information in the response is processed.

Email Content Parsing

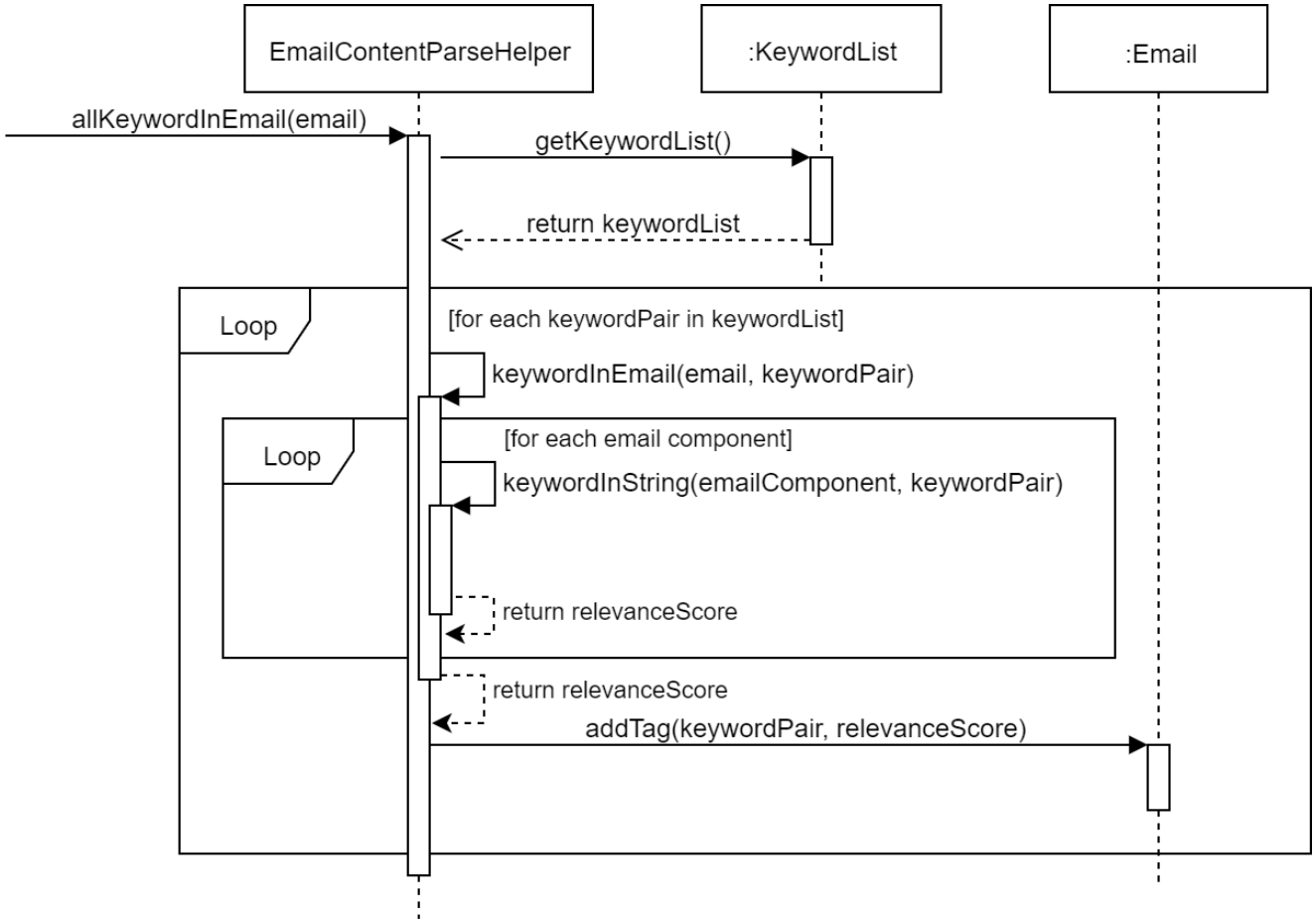


Figure N+1: Email Content Parsing