

Lee Wei Hern Jason – Project Portfolio

Project: Gazeeebo

About the project

My team of 4 computer engineering (CEG) students and I chose to enhance a command-line desktop chatbot into a school life planner for CEG students called **Gazeeebo** for our Software Engineering project. **Gazeeebo** enables users to manage their school academic progress, future school work plan, create notes and reminders as well as manage their contacts, places and financial expenses. This portfolio serves to document my contribution to the project.

My role was to design and write the codes for the CAP, contacts, edita task, recurring task and password features. The following sections illustrate CAP enhancements in more detail, as well as the relevant documentation I have added to the user and developer guides in relation to CAP enhancements. More information about the rest of the features can be found in the user guide and developer guide. Links will be provided under other contributions.

This is what Gazeeebo looks like:

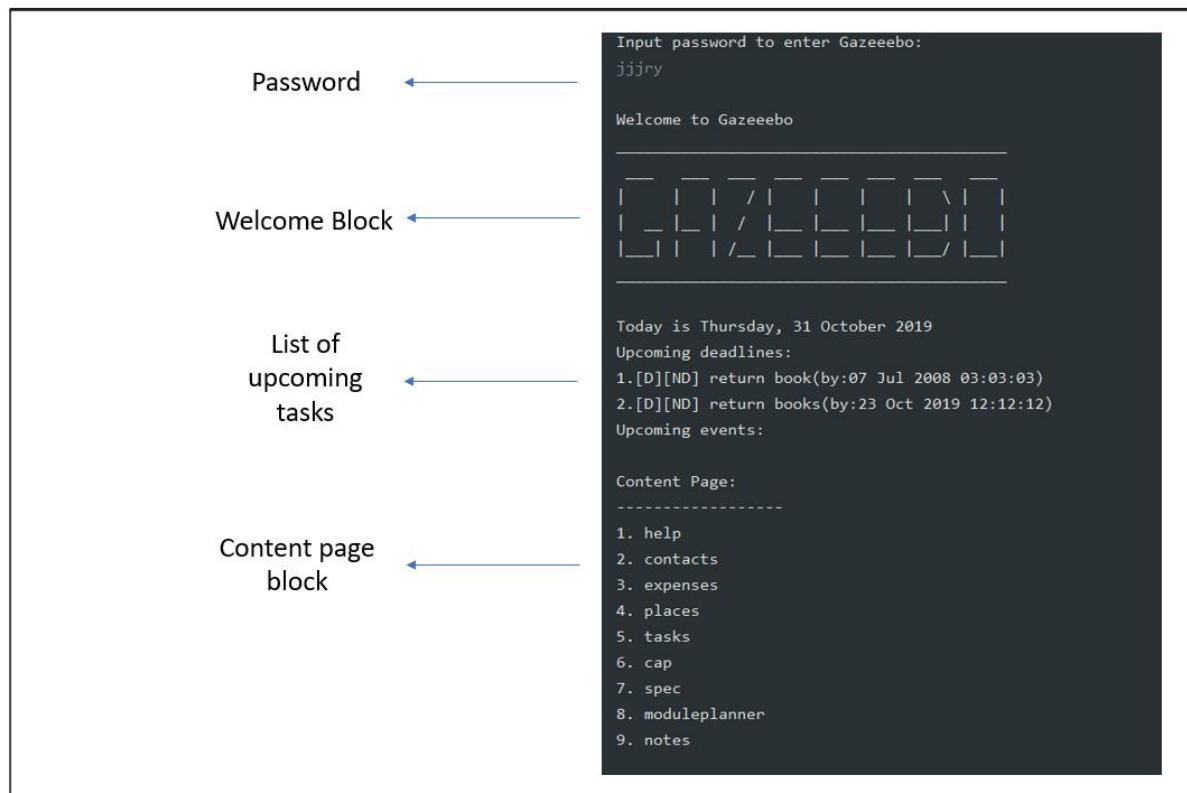


Figure 1. The user interface for **Gazeeebo**.

Mark ups used in the document

Grey Highlight	Words that are highlighted in grey are classes and components.
Courier New font	This font is used for commands to be keyed into the command line of the product.

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 CAP feature.

- **What it does:** Allows the user to add a module, list out all the modules or modules in a semester, find modules, delete modules and calculate their CAP.
- **Justification:** This feature improves the product significantly because a user can keep track of his grades and students can make a more informed decision on which modules to S/U during the S/U period to achieve a better CAP.
- **Highlights:** It requires an in-depth analysis of design alternatives. The implementation too was challenging as it required changes to existing commands.
- **Credits:** [NUS Grading System](#)

Code contributed: [\[Reposense\]](#)

Other contributions:

- Enhancements to existing features:
 - Created a password feature that provides security measures for users. Additionally, the user is able to change his password too. Found in section 4.1 of [Developer Guide](#) and sections 3.9. and 3.10. of [User Guide](#).
 - Created a contact page that allows users to keep track of their friend's contact. Has an inbuilt NUS and CEG category to help CEG students organise their contacts better. Found in section 4.4 of [Developer Guide](#) and section 3.5 of User Guide.
 - Created a function to edit tasks. The user is able to edit the description or time of the task instead of recreating the task whenever he wants to update the task. Found in section 3.2.13 of User Guide.
 - Designed a recurring feature. The recurring feature saves CEG students the hassle of recreating tasks weekly, monthly or yearly. When a recurring task is marked as done, it recreates a new task with the updated date. Found in Appendix B of [Developer Guide](#) and section 3.2.24 of User Guide.
- Documentation:
 - Wrote instructions for the usage of features (recurring command, edit command, contact command and CAP command) in User Guide.
 - Wrote Java documentation for the codebase
 - Update Developer Guide.

- Community:
 - PRs reviewed and merged (with non-trivial review comments): [#318](#), [#303](#), [#302](#), [#297](#), [#295](#), [#294](#), [#283](#), [#268](#), [#267](#), [#178](#), [#169](#), [#159](#), [#149](#), [#146](#), [#123](#), [#120](#), [#115](#), [#101](#), [#99](#), [#85](#), [#74](#), [#73](#), [#71](#), [#60](#), [#58](#), [#57](#), [#56](#), [#55](#), [#49](#), [#48](#)
 - Added labels for organising and prioritising issues, pull request, merging. This makes project management more efficient and effective.

Contributions to the User Guide

Purpose of Section

The following is an excerpt from our Gazeebo [User Guide](#), showing additions that I have made for the CAP features. For the full documentation, please refer to section 3.6 of my group's [User Guide](#).

Part of the CAP section documentation

A function to calculate the CAP of modules and to store the module's semester, module code, module credit, and module grade. The following shows that the user is able to input the commands in one line, but these features are also able to input multiple steps command which can be seen in Gazeebo User Guide in section 3.6.

Entering the CAP page.

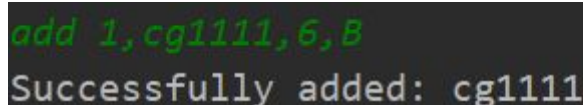
- Enter `cap` in the main menu page and press ENTER. Alternatively, the user can type in the index of `cap`, 6 and press ENTER.
- The user can only go to the contacts page from the main menu.

Adding a new module

Adds and stores a new module.

Different ways to add a new module:

1. Type in the command `add semester,code,credit,grade` and press ENTER. A success message will appear as seen in Figure 2.
2. Alternatively, you can input `add` or the index of `add`, 1. The system will then prompt you to enter the module's information, `semester,code,credit,grade`.



```
add 1,cg1111,6,8
Successfully added: cg1111
```

Figure 2. An example of adding a module

Finding a module(s)

Finds all the modules by module's code.

Different ways to find a module:

1. Type in the command `find code` and press ENTER. A success message will appear as seen in Figure 3.

2. Alternative you can input `find` or the index of find, 2. The system will prompt you to enter the module's code.

```
find CS1231
Sem | Module code | MC | CAP
-----
1   | CS1231       | 4  | B+
```

Figure 3. An example of finding a module.

Listing of modules

Display all modules or modules from a particular semester.

Different ways to list a module:

1. Type in the command `list semester` and press ENTER. A success message will appear as seen in Figure 4.
2. Alternative you can input `list` or the index of the list, 5. The system will prompt you to enter the semester you want to list out.

```
list 1
Sem | Module code | MC | CAP
-----
1   | ES1103       | 4  | B+
-----
1   | CG1111       | 6  | B+
-----
Sem 1 CAP: 4.0
```

Figure 4. An example of listing a semester.

Deleting a module

Delete an existing module.

Different ways to delete a module:

1. Type in the command `delete code` and press ENTER. A success message will appear as seen in Figure 5.
2. Alternative you can input `delete` or the index of the list, 3. The system will prompt you to enter the module code you want to delete.

```
delete CS1231
Successfully deleted: CS1231
What do you want to do next ?
```

Figure 5. An example of deleting a module.

Contributions to the Developer Guide

Purpose of Section

The following section shows my additions to the Gazeeebo [Developer Guide](#), which showcase my ability to write technical documentation and draw technical diagrams. For the full documentation of the CAP feature, please refer to section 4.2 of the [Developer Guide](#).

CAP Page

The CAP feature is facilitated by `CapCommandParser`. The user will go into the CAP page by typing `cap` on the main page. The CAP feature is mainly implemented by `AddCapCommand`, `DeleteCapCommand`, `ListCapCommand`, `FindCapCommand`, `CalculateCapCommand`, and `ConvertGradeToScoreCommand` take care of the logic for the CAP page, which is implemented by `CapCommandParser` class. In turn, `CapCommandParser` implements the following operations:

- `CapCommandParser#add()` - Adds a module into the caplist.
- `CapCommandParser#delete()` - Delete a module from caplist.
- `CapCommandParser#find()` - Find a module from caplist based on the name.
- `CapCommandParser#list()` - list modules from caplist and it shows the cap.
- `CapCommandParser#calculatecap()` - calculate the CAP in the semester or for all modules
- `CapCommandParser#convertergradetoscore()` - converts alphabetical grade to numerical score

Here is a class diagram shown in Figure 6 to show the structure of the classes that implement the CAP page feature:

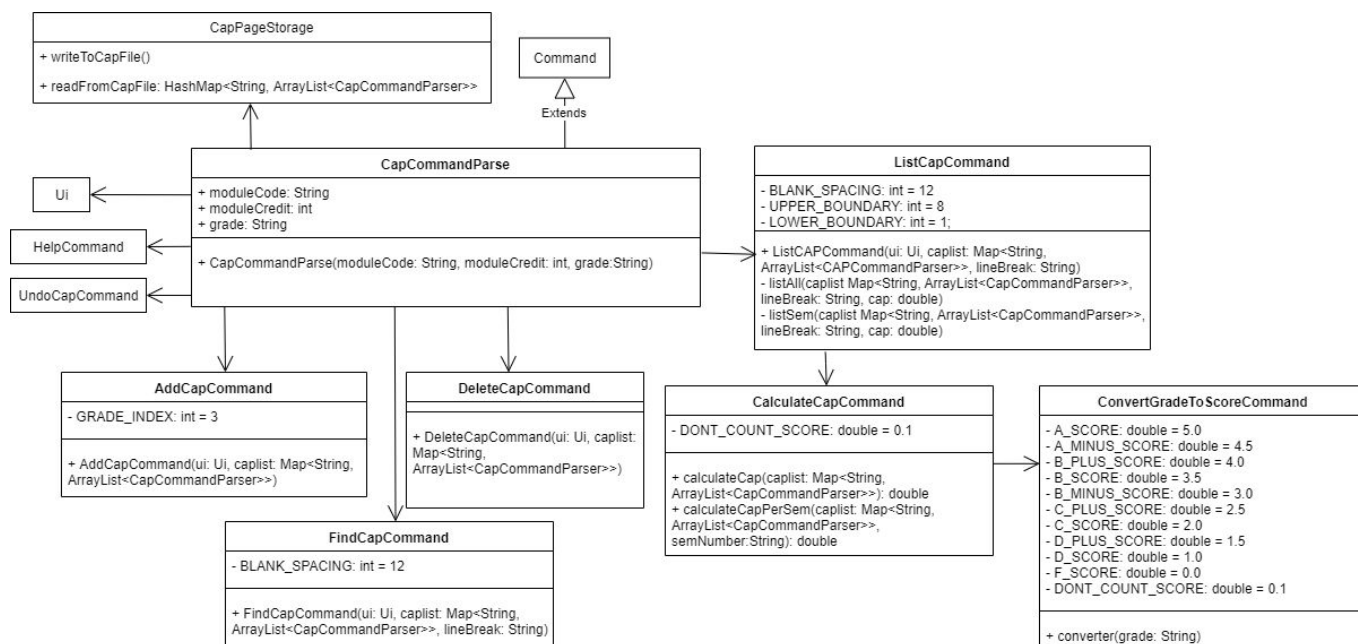


Figure 6. Class diagram for CAP page feature

The following diagram shows the interactions for the method call `execute()` on a `CapCommandParser` object:

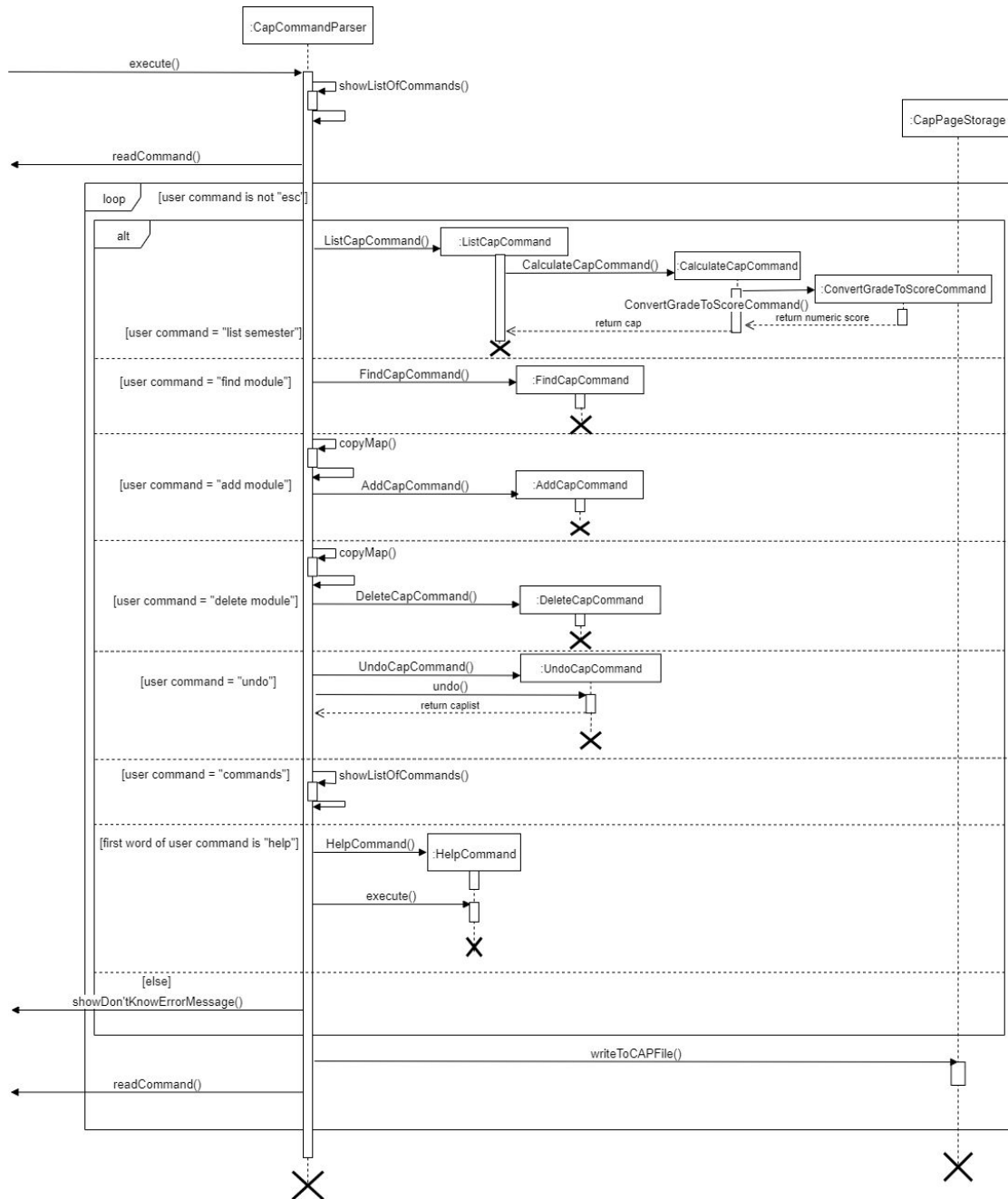


Figure 7. Sequence diagram for method `execute()` of `CapCommandParse`.

The following steps show a usage scenario:

When an `AddCapCommand` is called:

Step 1. User will have to input the name and the user wants to add. User enters add sem number, module's code, module's credit, module's grade.

Step 2. The new module will be added to `Map<String, ArrayList<CAPCommandParser>> caplist`.

When an `DeleteCapCommand` is called:

Step 1. User will have to input the name the user wants to delete. User enters `delete module's code`.

Step 2. The module specific will be removed from `Map<String, ArrayList<CAPCommandParser>> caplist`.

Step 3. If there is no such name in the `Map<String,String> contactList`, a "not found" message will be shown.

When a `ListCapCommand` is called:

Step 1. User will have to input which semester the user wants to list. User enters `list semester number`.

Step 2. `CalculateCapCommand` is called to calculate the cap of the semester. In `CalculateCapCommand`, `ConvertGradeToScoreCommand` is called to convert the user's alphabetical grade to numerical grade.

Step 3. The `Map<String, ArrayList<CAPCommandParser>> caplist` and the CAP will be printed out.

When a `FindCapCommand` is called:

Step 1. User inputs the module code the user want to find. User enters `find module code`.

Step 2. The module information will be printed out.

Step 3. If there is no such module code in the `Map<String, ArrayList<CAPCommandParser>> caplist`, a "not found" message will be shown.

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

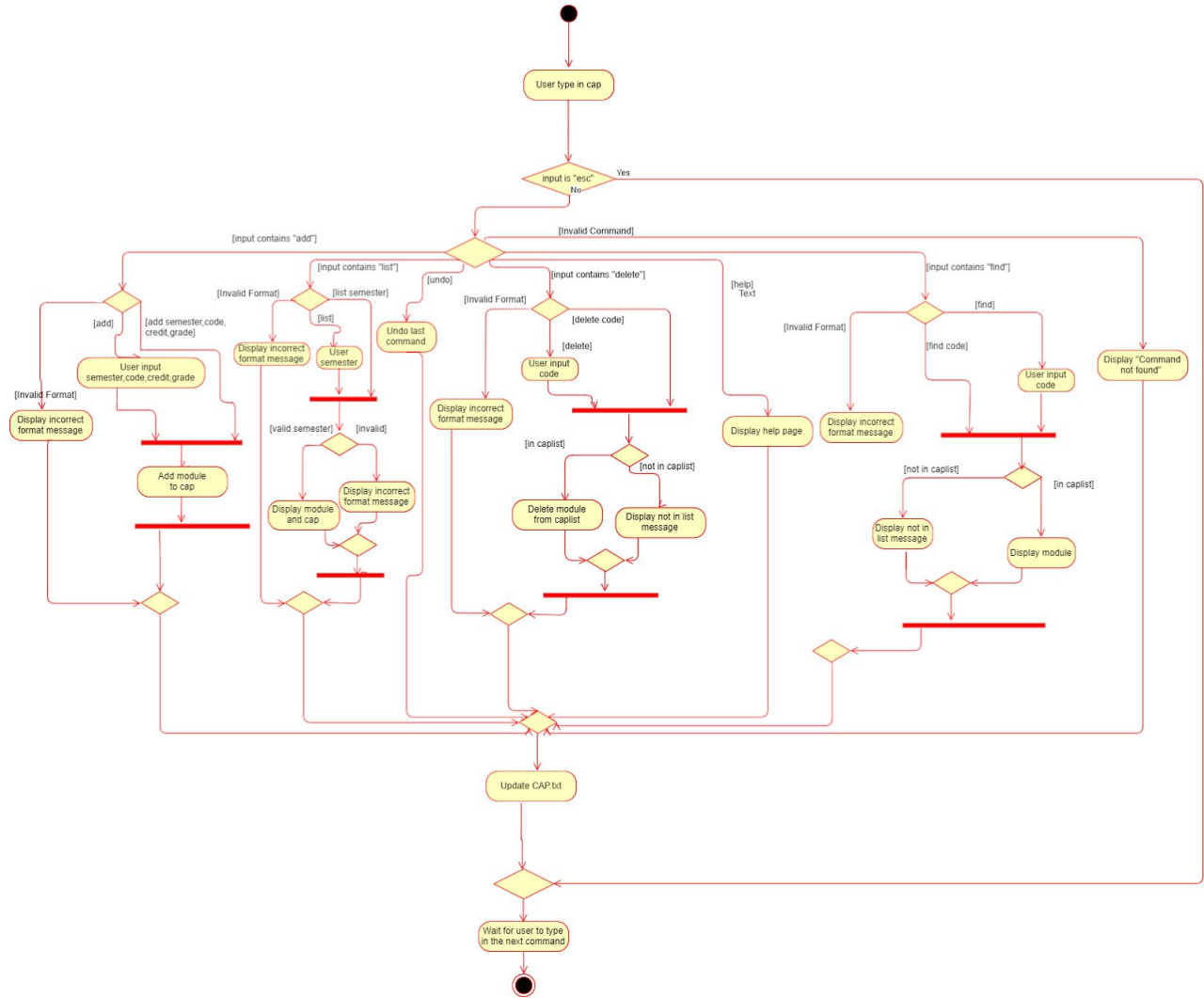


Figure 8. Activity diagram for the CAP page.

Design Considerations

Aspect: Data structure to support CAP

- Alternative 1(current choice): Using TreeMap to store all the module's information (semester number, code, credit, grade).
 - Pros: TreeMap is has a natural ordering and the keys are sorted, thus reducing the need to sort the map when it is listed out.
 - Cons: TreeMap's time complexity of insertion and delete is $O(\log(n))$, which is slower than HashMap.
- Alternative 2: Using a HashMap to store names and numbers
 - Pros: HashMap has a time complexity of $O(1)$ for insertion and deletion, which is faster than TreeMap.
 - Cons: Null values/keys are allowed in a HashMap. Requires an additional validation to ensure users do not input null values/keys.