ModPlan v1.1 - Developer’s Guide

By: `Team AY1920S1-CS2113T-F10-1` Since: `Aug 2019` License: `NUS School of Computing`

1. Introduction

ModPlan

ModPlan is a module planning system that expands on NUSMODS to help NUS students to plan their modules for all four years of their bachelor’s degree.

ModPlan is currently only designed to work for Computer Engineering students, and the requisite modules for graduation.
ModPlan uses NUSMODS API to extract official module data from the NUS Registrar’s Office.

This Developer Guide is meant for any software developer who wishes to contribute to or test ModPlan, and can find architecture, implementation methods and high-level design considerations within this file.

2. Setting up

Prerequisites

- JDK version: `11` or above
- Recommended IDE: `IntelliJ IDEA`
- Fork this repo to your GitHub account and clone the fork to your computer

Importing the project into IntelliJ

1. Open IntelliJ (if you are not in the welcome screen, click `File` > `Close Project` to close the existing project dialog first).
2. Set up the correct JDK version.
   - Click `Configure` > `Structure for new Projects` (in older versions of Intellij:`Configure` > `Project Defaults` > `Project Structure`).
   - If JDK 11 is listed in the drop down, select it. If it is not, click `New…` and select the directory where you installed JDK 11.
   - Click `OK`.
3. Click `Import Project`.
4. Locate the project directory and click `OK`.
5. Select `Create project from existing sources` and click `Next`.
6. Rename the project if you want. Click `Next`.
7. Ensure that your src folder is checked. Keep clicking `Next`.
8. Click `Finish`.

Running the Project

- If you wish to run the .jar file in a mainstream OS, simply double click the .jar file, and a command line interface should run in a few seconds.
  This is assuming you have a JDK version of `11` and above.
- If you are not using `gradle`, simply locate the `Duke.java` file under `PROJECT_DIRECTORY/src/main/java/Duke.java`, and run it in your IDE.
- If you are using `gradle`, import the project as a gradle project using the `Import Gradle Project` function in IntelliJ, and run `Duke.java` from the Gradle toolbar.

Appendix A: Product Scope

A better module planner

We aim to fullfill a need that is currently lacking in module planning, which in this case the is ability to plan ahead for more semesters up until graduation. Additional features would likely include the ability to generate a projection report for CAP computation and CCA planning.

Appendix B: User Stories

Priorities: High (must have) - `* * *`, Medium (nice to have) - `* *`, Low (unlikely to have) - `*`

Priority	As a …	I want to …	So that I can…
* * *	NUS CEG Student	Search for a module’s workload	Balance my workload for the current semester
* * *	NUS CEG Student	Monitor my total workload from my modules	Track my total workload for the current semester
* * *	NUS Student	See my daily timetable	Keep a schedule of what classes and extra-curricular activities I have
* * *	NUS CEG Student	Check if I have completed the required prerequisite modules	Plan ahead for what modules to take
* * *	Forgetful NUS CEG Student	Add up my total number of MCs taken	Track my progress towards graduation
* * *	NUS CEG Student	View the core modules required for graduation	Know what are the modules I still need to take to graduate
* *	NUS Student	Add CCAs to my class timetable	Take CCAs that do not clash with my lessons
* *	NUS Student	Create a custom module for my CCAs	Personalise the timing and location of my CCA in my timetable
* *	NUS Undergraduate Student	Know requirements for a Master’s/PHD at NUS	Plan my course of action if I wish to apply for post-graduate studies

Priority	As a …	I want to …	So that I can…
* *	NUS CEG Student	Easy access to my recommended study schedule	Know what modules I should prioritise bidding for
* *	NUS CEG Student	Plan to take modules ahead of the current semester	Alter my holiday/graduation plans as required
* *	NUS CEG Student	Know what GE modules I have not completed	Plan to take GE modules over a few semesters
* *	NUS CEG Student	View the total number of Level-1000 modules taken	Check if I have exceeded the 60MC limit for Level-1000 modules
* *	NUS CEG Student	Know if the module has S/U options	Plan ahead for my S/U usage
* *	NUS CEG Student	Project my future CAP based on my expected and past grades	See how hard I must work to hit my target CAP
*	NUS CEG Student	Download my timetable as a photo	View it on other mediums such as my mobile phone
*	NUS Student	Know the directions to my classes	Plan my route accordingly
*	NUS Student	Know my priority score when bidding for a module	Plan my module bidding appropriately

Priority	As a …	I want to …	So that I can…
*	NUS Student	See a list of my course’s modules available in SEP/NOC	Plan what modules to take should I go for SEP/NOC
*	Exchange Student	Know if a module can be mapped to my home university	Plan what modules to take in NUS

(more to be added as necessary)

Appendix C: Use Cases

C.1. Use Case C01: Adding modules to user’s timetable

Actor: NUS CEG Student

1. User inputs the module code
2. ModPlan shows the module information to the user, such as description, number of MCs, prerequisite modules etc. and requests confirmation from the user to add this module
3. User confirms they want to add the module
4. ModPlan shows the non-clashing available timings of the module to the user
5. User confirms which class timing they wish to add to their timetable
6. ModPlan adds that specific class to the user’s timetable, and prints the user’s updated timetable
   Use case ends.

Extensions

2a. If the module is a Level-1000 module, ModPlan checks for the user’s current number of Level-1000 modules taken
2b. If the limit is not exceeded, proceed to step 3
2c. If the limit will be exceeded, warn the user, and prevent addition of the module
2d. Additionally, if the prerequisites of the module have not been fulfilled, prevent addition of the module, and inform user of the modules needed to be taken
Return to step 3.

C.2. Use Case C02: Viewing graduation requirements

Actor: NUS CEG Undergraduate Student

1. User inputs their course name

2. ModPlan shows the courses that match the user’s input
3. User selects the correct course they wish to check graduation requirements for
4. ModPlan displays all the modules required for graduation, and lists the number of MCs required for graduation
   Use case ends.

Extensions

3a. User can input the modules they have taken already that count towards graduating that course
3b. ModPlan will exclude these modules from the list and MC count
Return to Step 4.

C.3. Use Case C03: Viewing class and CCA timetable

Actor: NUS Student

1. User inputs the command to `view timetable`
2. ModPlan shows the user their current timetable, including class and CCA timings

C.4. Use Case C04: Viewing of modules taken in past semester

Actor: NUS Student

1. User inputs the command to `view past modules`
2. ModPlan shows the user a list of all modules taken, and those they are currently taking.

C.5. Use Case C05: Generation of current CAP based on past modules

Actor: NUS Student

1. User inputs the command to `generate CAP report`
2. ModPlan shows the user modules they had taken, and requests user to input their grades obtained
3. User inputs the modules they have taken, as well as the respective grades obtained
4. After inputting the grades, ModPlan calculates and shows the user their current MCs accumulated and CAP

Extensions

4a. User can then input a future module they plan to take and project their CAP 4b. ModPlan will show the projected CAP using grades the user obtained from the module’s prerequisite classes

C.6. Use Case C06: Check Module information via input search

Actor: NUS Student

1. User inputs the command to `search module information`
2. ModPlan shows the user key information regarding the module, if it is SU-able or if it has any prequisites.

C.7. Use Case C07: Creating a personalised Module (eg. for CCAs)

Actor: NUS Student

1. User inputs the command to `create custom module`
2. ModPlan prompts the user for additional details of the custom module, such as description and times
3. User inputs the description and date/times
4. ModPlan prompts user to confirm addition of custom module to timetable
5. User confirms addition
6. ModPlan adds custom module to timetable, and shows user updated timetable

Extensions

5a. User can cancel addition 5b. ModPlan will cancel addition of custom module, and delete information inputted

C.8. Use Case C08: Checking number of MCs taken

Actor: NUS Student

1. User inputs the command to `check MC`
2. ModPlan will show the total MCs taken up to this point

Extensions

1a. User can specify additional parameters to check MCs completed for specific periods+ * eg. `check MC 1-1` will check for MCs taken in Year 1 Semester 1

Appendix D: Non-Functional Requirements

1. ModPlan should run on any machine with JDK `11` and above installed.
2. ModPlan should be fast to view and input commands.

3. ModPlan should require as few steps as possible for the user to do what they want to do.
4. ModPlan should store data between sessions so the user does not have to input all their information again.
5. ModPlan should scrape data from NUSMODS API at least once a day to keep up to date with any changes in modules.

Appendix E: Glossary

CEG : Computer Engineering