Product Name

CanoE-COACH

Project Direction

Direction 1: Evolve AB3 into a more powerful or more optimized contact management app.

Target User Profile

1) **CS2103 Instructors**
2) **Canoeing Coach for Secondary School Students CCA**
    a) **Keeping track of attendance (3 coaches with 90+ students)**
        i) **Need to sound warnings if certain students do not attend consecutive trainings**
    b) **Inventory Management (boat/equipment allocation to students)**
        i) **Renewal of boat permits**
        ii) **Allocation to students**
    c) **Task List**
        i) **each of the member has certain responsibility to follow up with**
    d) **Performance Tracking**
        i) **Physical Fitness**
        ii) **Time trials**
    e) **Scheduling (Availability)**
3)

Tips

- **Don't be too ambitious in the beginning (start with small amounts)**

Value Proposition

1) **Canoeing Coach**
    - Attendance, Performance (with sorting feature?)
    - Managing large class sizes
    - Data Visualisation (Ability to look at overall performance of team and analyze general statistics)

- One-Command to see overall statistics

Problem Addressed

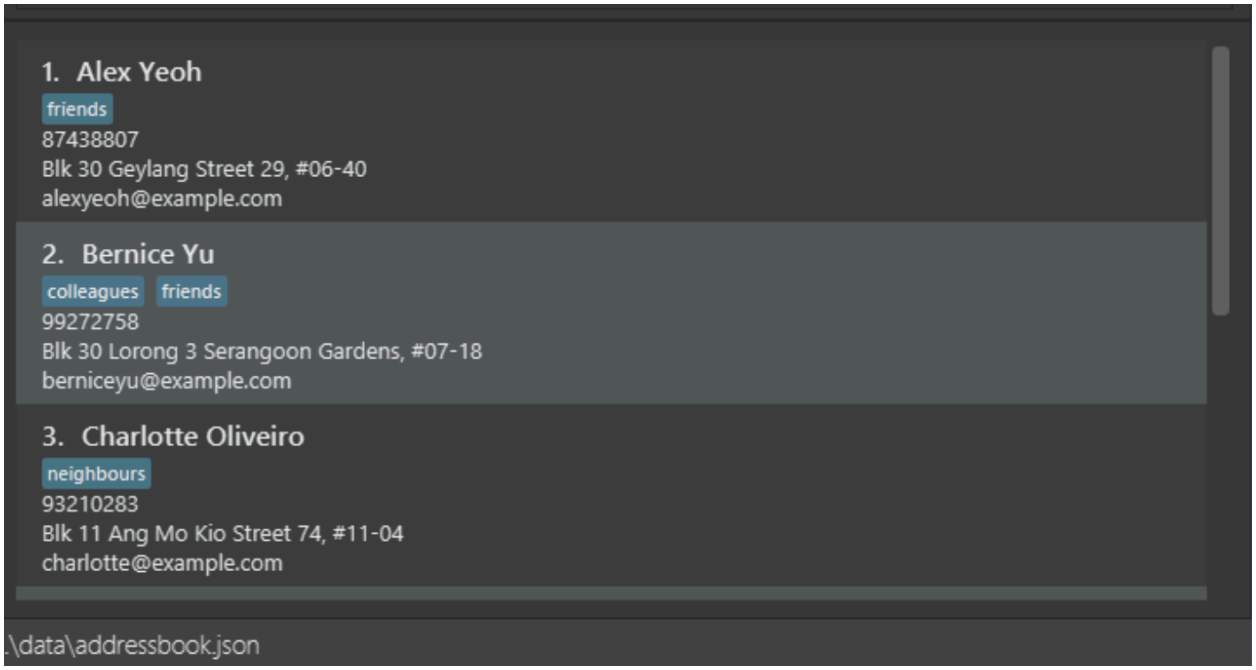
- Tedious to keep track of all the individual performance and attendance as everything is keyed into excel sheets now.
-Many processes mentioned above are very manual. The individual has to find the excel sheet manually, and input.
- No quick way to pull up schedules of different students
-

A sample feature list from a simple Minesweeper game (only a brief description has been provided to save space):

1. Basic play – Single player play.
2. Difficulty levels
    - Medium-levels
    - Advanced levels
3. Versus play – Two players can play against each other.
4. Timer – Additional fixed time restriction on the player.
5. ...

- Features
    - Viewing help : `help`
    - Adding a person: `add`
    - Listing all persons : `list`
    - Editing a person : `edit`
    - Locating persons by name: `find`
    - Deleting a person : `delete`
    - Clearing all entries : `clear`
    - Exiting the program : `exit`
    - Saving the data
    - Archiving data files `[coming in v2.0]`

https://se-edu



cation.org/addressbook-level3/UserGuide.html

Feature List - v1.1

1. **Student records -** (Name, ID, Number, Email, Level [tag], Academic Year [tag])
    a. Add
    b. Remove
    c. View by name
    d. Dismissal Time for weekdays?
2. **Help** - Viewing of available commands (and how to use)
3. ~~**[Stretch] Training schedules**~~
    ~~a. Create (based on tag) and display~~

Feature List - v1.2

1. **Find a common time to be able to conduct a training session.**
    a. Based on maximum number of available students (best case)
2. **Training sessions** - (Date, time, list of students)
    a. Create
    b. Read
    c. Update
    d. Delete
3. **Add student into training session. Delete student from training session.**
4. **Add training session as part of student schedule.**

5. [GUI] Additional window to display training sessions.

Issues - v1.2a

Faqih

1. Find a common time to be able to conduct a training session.
    a. Filter common time - high

Andy

1. Add student ID as part of student's fields - HIGH
    a. Id will be inner class of Student
    b. Constraints and guarantee:
        i. Unique (creation is done WITHIN student class)
        ii. Id CANNOT be edited
        iii. If student is delete, the used Id will NOT be reused for future students
2. Find by ID

Andy, Wei Heng

2. Training sessions - CRD
    a. Create training session class - high (date and start-time)
        i. created blank
    b. Create command to create training session - high
    c. Delete command to update training session - high
3. Add student into training session.
    a. Manual adding by coach - e.g. "ts-add 1, 2, 10, 21" - high
    b. Mass auto adding by system - e.g. "ts-add all" - medium

Jiadong

4. Add a training session (date) as part of student's fields.
    a. Another attribute in the student class - high

Kerk (Swimming Spongebob Lead/Consultant)

Not resizable

5. [GUI] Additional window to display training sessions - medium
    a. Sadjalskdjasklj (urgent)
6. [GUI] Minor improvements (display of student information) - high
7. [GUI] water design (CANOE-ish design)

Issues - v1.2b

Faqih

1. Create a new predicateList class for find common time to anyMatch instead of allMatch.

Wei heng

2. Add student into training session.
    a. Manual adding by coach - e.g. “ts-add 1 [STUDENT ID]” - high
    b. Mass auto adding by system - e.g. “ts-add all” - low
3. Remove student from training session.

Andy, Faqih

4. Editing student must also make sure training session updates
    a. Set student (copy method from **UniqueStudentList**)
    b. Auto remove student if the dismissal time doesnt make it anymore
        i. This requires DismissalTime to have a method
        ii. if (!student.**isAbleToAttend()**) training.remove(student)
5. Remove student also must make sure training session updates

JD Kerk

6. boolean student.**isAbleToAttend**(LocalDateTime trainingDateTime)
    a. switch (trainingDateTime.getDay())
    b. case (MONDAY):
    c. getMondayDismissalTime().isBefore(LocalTime other)
        i. `DismissalTime other` is the training’s time
        ii. If true, then the student can STILL make it for the training
        iii. Else, false

JD, Kerk

7. Take in a certain dismissal time as input and display all students that can make it at that time
8. Reorganise the Student Card in GUI

User Stories - v1.2:

1) As a user, I can add a time/date for a class, so that I can know when the next class is.
2) **As a user, I can see everyone's available schedules, so that I can find a common available time to have classes.**
3) As a user, I can check if any of the training timings scheduled are clashing with each other, so that I can avoid double booking incidents.

4) As a busy user, I can have a quick and fast glance at student grouping for the day's session, so that I can allocate more time to preparing equipment and lesson material.

V1.2:

Summary
-> Support the add, delete, modify students' details.
-> Find a common time to be able to conduct the training.
-> Basic Error handling?

5) As a user, I can add the available schedules of a particular student, so that I can know when that student is available to have classes.
6) As a user, I can add a time/date for a class, so that I can know when the next class is.
7) **As a user, I can see everyone's available schedules, so that I can find a common available time to have classes.**
8) As a new user, I can easily create new student record entries, so that I can store information/data of students in a structured way.
9) As a user, I can delete students that have quit and automatically remove them from my schedule, so that I can keep track of only current active students.
10) As a user, I can check if any of the training timings scheduled are clashing with each other, so that I can avoid double booking incidents.
11) As a new user experimenting with the app, I want to receive help prompts should I key in mispelled commands, so that I can quickly learn and pick up the correct commands.
12) As a user using the app, I want to be able to easily modify the personal details of my students, so that the most accurate information can be displayed.
13) ~~As a user, I want to be able to know which are the days that are available for me to hold a training session.~~ Duplicate of (3)

V1.3:

- Holding sessions
- Accounting for how many students should attend the session (based on that v1.2)
- Attendance of the students (who actually attend)
-

Faqih:

14) As a user, I can assign a particular exercise to a student for him/her to do, so that he/she can do that exercise as part of their training programme.
15) As a user, I can track a student's training programme, so that I can assess his/her abilities.
16) As a user, I can add a time/date to a task, so that I can record when a task needs to be done by.

Andy:

1) As a user, I can rank the student's fitness scores, so that I can identify which students are weaker and require more help.
2) As a user, I can get notified of students that miss classes, so that I can approach them to find out if there are any issues from them.
3) As a busy user, I can have a quick and fast glance at student grouping for the day's session, so that I can allocate more time to preparing equipment and lesson material.
4) As a clumsy user, I can create email templates of lesson schedules and student grouping, so that I can send students their timetable without human error.
5) As an expert user, I can create shortcuts that combine several commands together, so that my workflow is faster.

Wei Heng:

1) As an expert user, I can schedule automatic email reminders to remind students of their training nearer to the training date, to avoid no-show incidents.
2) As a potential user exploring the app, I can see sample data on the app to see how it may benefit me in scheduling trainings.
3) As an experienced user, I can add notes tagged to certain students to remind me of their quirks, so as to offer more personalised trainings for them.
4) As a user, I can easily notify all of my students when I go on vacation, so that I do not have to tell them individually.

Jiadong:

1) As an impatient user using the app, I want to be able to quickly modify the training schedules of my students, so that last-minute emergencies and contingencies can be accounted for.
2) As an experienced user, I want to have a default overview of my team statistics once I launch the app, so that I can have a bird's eye view of how the team is doing in various aspects.
3) As an impatient user, I can see all the pending tasks with upcoming deadlines on a single command, so that I can quickly know what needs to be done next.
4) As a user, I want to be able to add new metrics to track the performance of my students should the existing set of metrics be insufficient, so that I can keep track of different elements of my students' performance.
5) As a user, I want to be able to quickly identify which students are falling behind the average performance of the team, so that I can easily identify students who need additional training.

Kerk:

1) As an impatient user of the app, I want to be able to easily compare the performance of any two given students.
2) As a user, I want to be able to account for the attendance of all the students, and if they are not present, the reasons for so.
3) As a forgetful user, I want to be alerted if there are any upcoming events (within the next week), so I will not miss the event.
4) As an impatient user, I want to be able to update the attendance of all the students in a single command.
5) As a user, I want to be able to see what's the progress of the student over time.

User stories

Priority	As a …	I want to …	So that I can…
* * *	new user	see usage instructions	refer to instructions when I forget how to use the App
* * *	user	see everyone’s available schedules	find a common available time to have classes.
* * *	user	delete students that have quit and automatically remove them from my schedule	keep track of only current active students
* * *	user	add the available schedules of a particular student	know when that student is available to have classes
* * *	user	easily create new student record entries	store information/data of students in a structured way
* * *	user	easily modify the personal details of my students	see the most accurate information
* * *	user	filter students by their name	easily retrieve the details of a particular student without having to go through the entire list
* *	user	filter students by their Academic Year	easily retrieve all students of a particular academic level
* *	user	filter students by their Phone Numbers	easily find students by their contact details
* *	user	filter students by their dismissal time / schedules	schedule training sessions
* *	user	tag students by their Academic Year	schedule trainings catered for students in a particular academic year

Use cases

(For all use cases below, the System is the CanoE-COACH and the Actor is the user, unless specified otherwise)

UC01: Add a student

MSS

User requests to add a student to the student list

CanoE-COACH adds the student Use case ends.

Extensions

1a. Name, phone number, email, or academic year is missing.

1a1. CanoE-COACH displays an error message. Use case resumes at step 1.

1b. Student with the same name already exists.

1b1. CanoE-COACH displays an error message. Use case ends.

UC02: Delete a student

MSS

User requests to list students

CanoE-COACH shows a list of students

User requests to delete a specific student in the list

CanoE-COACH deletes the student Use case ends.

Extensions

2a. The list is empty. Use case ends.

3a. The given index is invalid.

3a1. CanoE-COACH shows an error message. Use case resumes at step 2.

UC03: Edit a Student

MSS

User requests to list students

CanoE-COACH shows a list of students

User requests to edit a specific student in the list

CanoE-COACH edits the student's particulars Use case ends.

Extensions

2a. The list is empty. Use case ends.

3a. The given index is invalid.

3a1. CanoE-COACH shows an error message. Use case resumes at step 2.

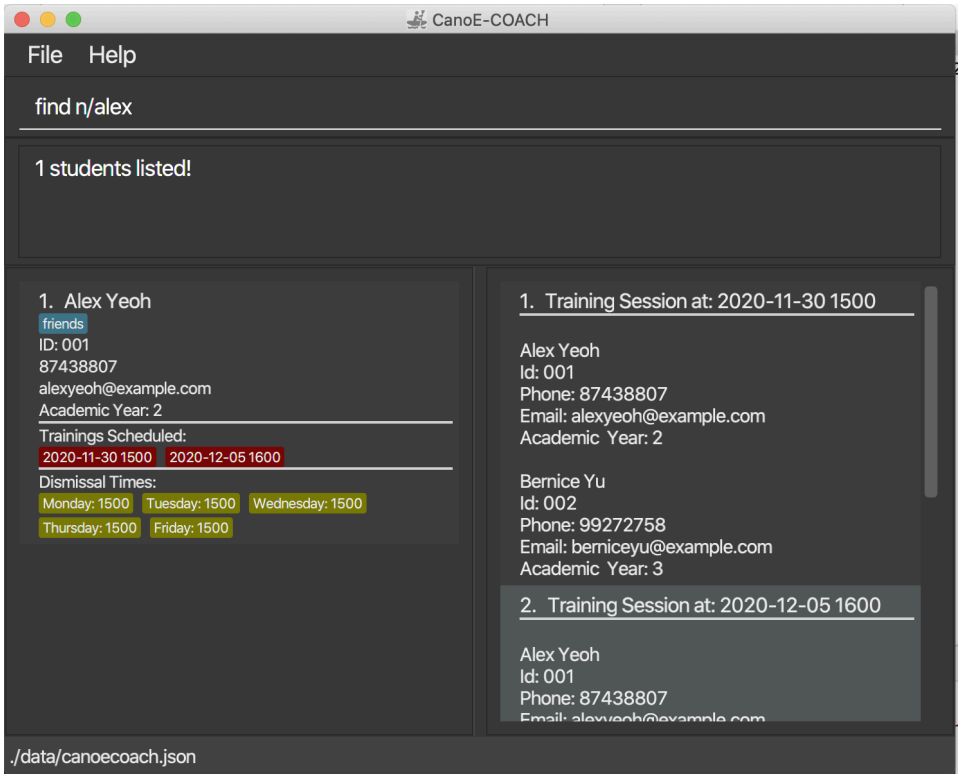
UC04: Find a Student
MSS
User requests to find students
CanoE-COACH shows a list of students who match the criteria Use case ends.
Extensions
1a. There are no parameters specified in the find command.
1a1. CanoE-COACH shows an error message. Use case resumes at step 1.

UC05: Clear all Students
MSS
User requests to clear all students
CanoE-COACH deletes all existing students in the student list Use case ends.

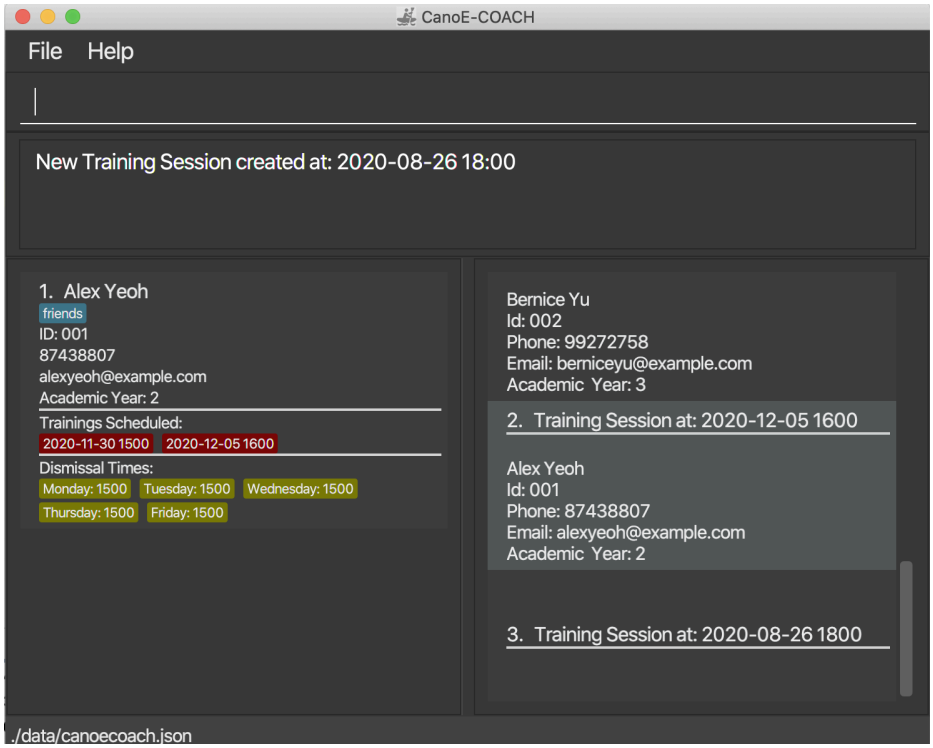
UC06: Add

v1.2 features demo

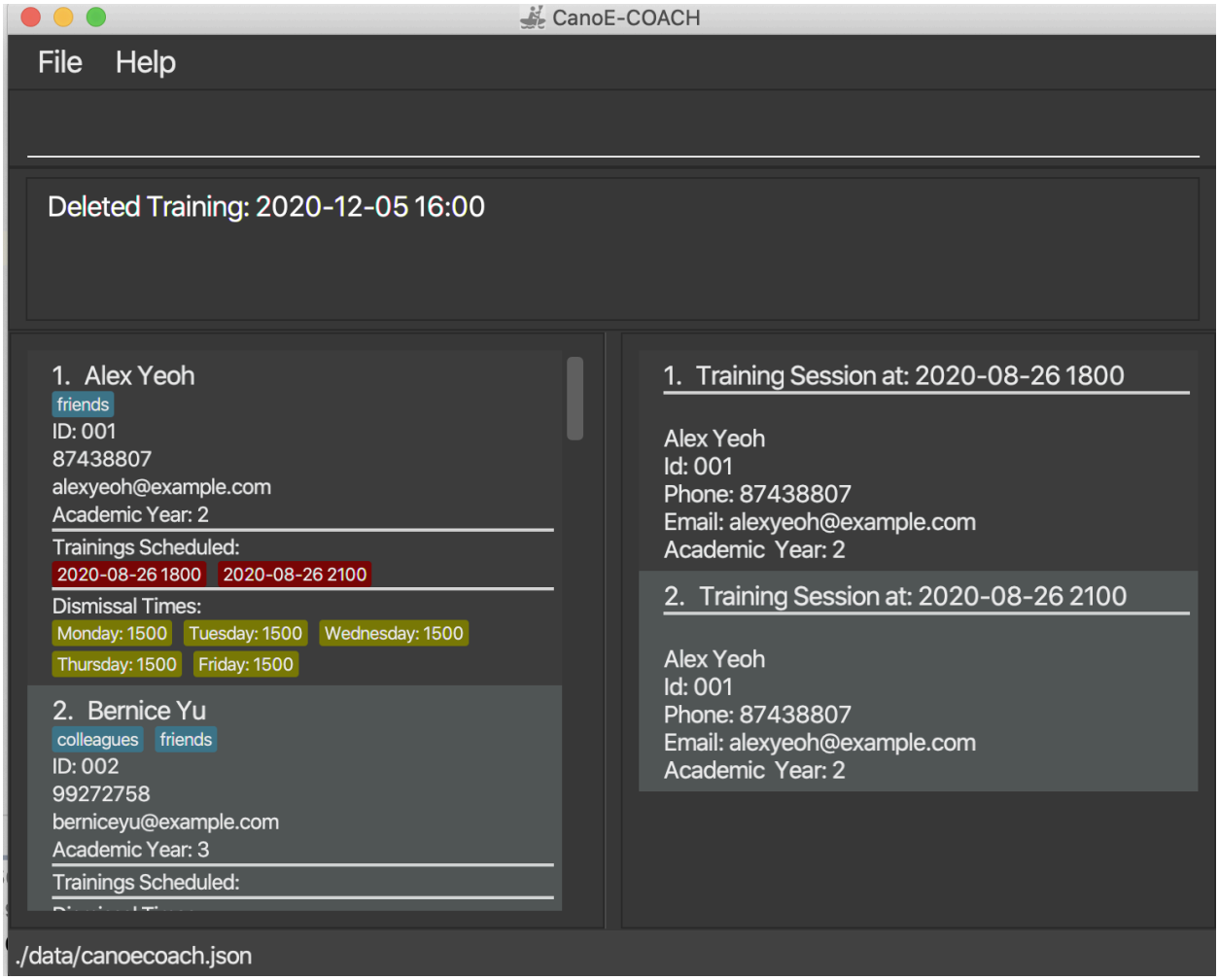
1). find student by name/contact/academic year/dismissal time/phone/email (find n/alex)



2). Create Training/Schedule Empty Training (training 2020-08-26 1800)



3). Delete whole trainings (delete-training 1)

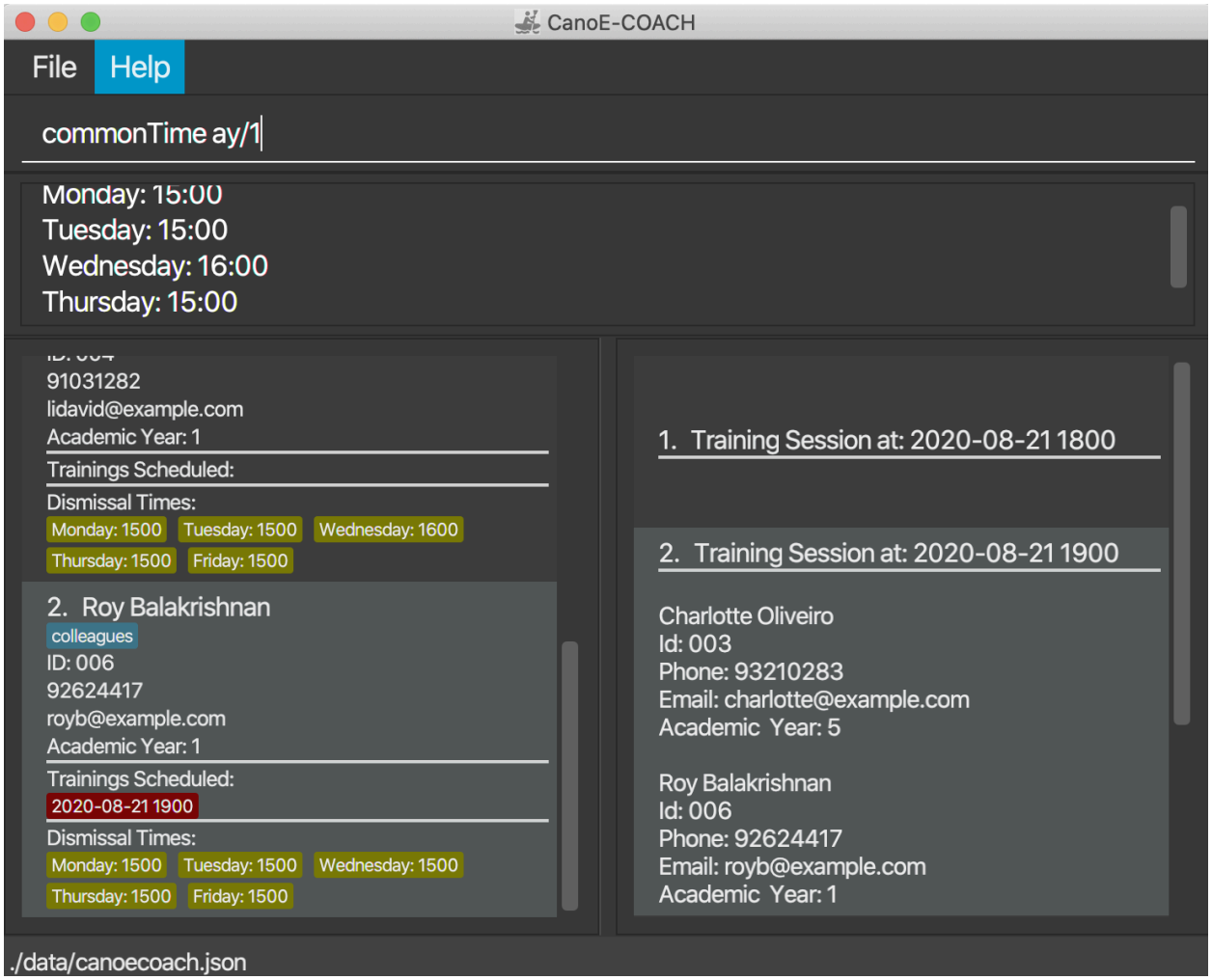


4). Add/Remove students from trainings (ts-add/ts-delete 1 1,2,3)

Trainings should show up on both student and training panel as they are synced. Any add/remove will update both panels as required. Each training can have multiple students -> increases complexity of code/program.

5). FindCommonTime (commonTime ay/2)

This allows the user to find the earliest common time to schedule a training. Find commonTime by Academic year and names are supported at the moment.



All other basic CRUD operations for students in the student list are still supported. (e.g. add, delete, edit)

POST MORTEM v1.2

Discuss with the team how the iteration went (i.e., what worked well, what didn't), and your plans to improve the process (not the product) in the next iteration.

1. What worked well?
    a. We worked to help each other out whenever there were issues.
    b. Regular updates on the progress, so we could rectify issues quickly.
    c. Merging of our features did not cause any major breaks to other features.
    d. Adding tests for implemented features during implementation.
    e. Proper comments on PRs that gave meaningful insights.

2. What could have been done better?
    a. Testing could have been more rigorous before PR
    b. Test more promptly upon implementation and before PR
    c. Work allocation could have been better
    d. Reduce coupling.
    e. Better and clearer annotations for code blocks
    f. Communicate the logic of the code (e.g. in PR)
    g. Do the branching workflow so that we can (locally) test each other's code before merging to reduce bugs.

Issues/New Features for V1.3 Trial

- Common time giving default 1500 - add a check in case no student matches - Faqih
- Editing student will cause that student to go to the bottom of the training list
    - Sort Training by Unique ID Order - Wei Heng [MID]
- Training display -> 2020-11-18 1500 (Mon) [QOL] - Wei Heng
- Need `ts-addall 1` which adds all students from the current filtered list to training 1. - Andy [HIGH]
    - Add all students who are able to make it for that training session index
    - Add all students under a certain ay/ who are able to make it for that training session index `
- Attendance (KERK)
    - Mark student's attendance command attend 1 id/1,2,3,4,5 (yes)

        - Add an attendance container? Into student class to store attendance of students
        - GUI updates + storage updates for attendance class
        - Regress: StudentBuilder, EditStudentDescriptorBuilder, TypicalStudent, JSON
    - `find-training id/2` Filter to show all Trainings for ONE student (JD)
    - Polishment: Currently we can create Training in the PAST (JD)
    - Notify coach student who did not attend training? (V1.3)

V1.3b

- Problem: Can only see 3 trainings for each student, hard to tell a student's overall attendance for all trainings
    - Solution: List all trainings for each student
- Refactor Attend to Attendance - done
- Refactor the Packaging - done
- Attendance::mark() and Attendance::unmark()
- Refactor ts-add & ts-delete to parse by student id instead of index
- Refactor commands to use Training::cloneTraining() and Student::cloneStudent()
- Make DeleteStudentToTrainingCommand AddStudentToTrainingCommand less weird. (WeiHeng) - Design principles
- Update UG
- Add to `find-training` command to filter for all students from one training
    - Input `find-training dt/2020-10-26` will show only the 2020-10-16 training AND all the students who are added to that training
    - Filters training list and student list (FindStudentTrainingCommand) preds
- Edit help's link to go to user guide straight, instead of README
- Ensure tests are sufficient for commands we wrote to ensure minimal bugs (individual)
- Change default tags (typical student) to more "CANOE-ish" - QOL

Features in UG:

- Viewing help: `help`
- Adding a student: `add`
- Editing a student : `edit`
- Delete student: `delete`
- Find : `find - Andy/Faqih`
- Common Time : `commonTime - Faqih`
- Create Training : `training - Wei Heng`
- Delete Training : `delete-training - Andy`

- Add Student to Training : `ts-add - Wei Heng`
- Add All Student to Training : `ts-addall - Andy`
- Delete Student from Training : `ts-delete - Wei Heng`
- Find all of a student's training : `find-training - JD`
- Mark student as having attended/not-attended a training : `mark-attend - Kerk`
- Clearing all entries: `clear`
- Exiting the program: `exit`

We can put archive as a proposal part of DG.

Attendance → Student

/* TODO on probation
Training ← TimeTrial / Race

List<Timing> timings
Timing {
  // is there a Java library for hh mm ss
}
Student.getTimings() -> [Timing1, Timing2, Timing3]
*/
Overall (Architecture / Sequence /)- Faqih [team lead]
Student - JD
Training - Wei Heng
Attendance - Kerk
Commands/Parsing (Logic DG) - Andy
GUI/Storage/Test (UI) - KIV

Faqih - Find, bug fix
Jiadong - bug fix, Student:DimissalTime, GUI: Highlight, ts-add
Wei Heng - bug fix, Training, ts-add
Kerk - bug fix, DimissalTime, GUI: Panels, ts-add ts-delete: Parser
Andy - Find, Student:Id, bug fix, Student:comparison/clone, ts-delete

https://nus-cs2103-ay2021s1.github.io/tp/DeveloperGuide.html#proposed-undoredo-feature

V1.3 Features

Extension:

Attendance

Fitness Score

v1.3 features demo

Watch On:

https://www.youtube.com/watch?v=vPYZQ3_fQbM

DRY-RUN ISSUES:

1) Student ID showing leading zeros but input ID does not accept any leading zeros
    a) REMOVE LEADING ZERO [ANDY]

2) Error messages need to be consistent with the error that is thrown
    a) Helps the user identify what is wrong with his input

3) Long error messages can be broken into separate lines

4) Marking of attendance should not be allowed for future trainings

5) Add more past training sessions into the sample data to enable testers to test the find-bad-students command

6) Edge case for valid datetime - Issue #184
    a) Specifically inside **ParserUtil::parseTraining**

```
LocalDateTime.parse("2021-02-31 1500",
DateTimeFormatter.ofPattern("yyyy-MM-dd
HHmm").withResolverStyle(ResolverStyle.STRICT))
```

7) Common-time command unfilters the student list after completion of command - need to remove the unfilter - Issue #183

8) Update UG to be more clear
    a) Mark/Unmark attendance
    b) ts-add/ts-delete/ts-addall

c) Students can be the same => they will be differentiated by their unique IDs
d) Dismissal time cannot be removed and defaults to 1500
e) Explain that dismissal time is for dismissal from SCHOOL not TRAINING
f) Remove ongoing training from UG. Only past and future
g) Ambiguity in the meaning of the word “scheduled”
h) Formatting error under Add Student command feature

Command error handling:

- Make error handling consistent throughout all commands
- Check validity of commands from left to right using parser
- Check whether student/training exists using command execution
- Command::execute() to throw the error

Add AllMatch and AnyMatchPredicateList in DG
Add in all commands that we implemented in the DG
Make UG more clear for our commands