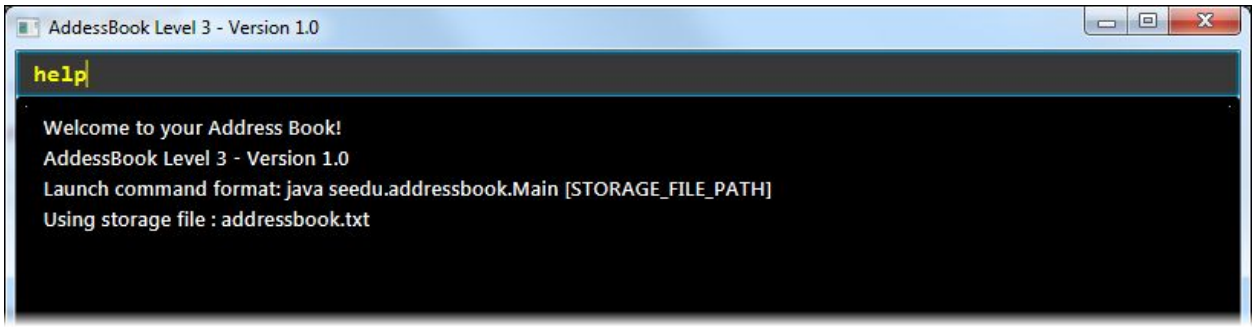
User Guide

This product is not meant for end-users and therefore there is no user-friendly installer.
Please refer to the DeveloperGuide section to learn how to set up the project.

Starting the program

1. Find the project pane (usually located at the left side)
2. Open up `src/seedu.ClassRepo` folder
3. Right click on `Main`
4. Click `Run Main.main()`
5. The GUI should appear in a few seconds



Viewing help : help

Format: help
Help is also shown if you enter an incorrect command e.g. `abcd`

Listing all persons : list

Shows a list of all persons in the ClassRepo.
Format: list

Finding all persons containing any keyword in their name: find

Finds persons whose names contain any of the given keywords.
Format: find KEYWORD [MORE_KEYWORDS]

The search is case sensitive, the order of the keywords does not matter, only the name is searched, and persons matching at least one keyword will be returned (i.e. `OR` search).

Examples:

- find John
  Returns `John Doe` but not `john`.

- find Betsy Tim John
  Returns Any person having names Betsy, Tim, or John.

View non-private details of a person : view

Displays the non-private details of the specified person.
Format: view INDEX

Views the person at the specified INDEX.
The index refers to the index number shown in the most recent listing.

Examples:
list
view 2
Views the 2nd person in the ClassRepo.

find Betsy
view 1
Views the 1st person in the results of the find command.

View exam dates of all subjects: viewexam

Displays the exam dates of all the subjects
Format: viewexam

Increasing privileges: authenticate

Authenticate the user as admin/tutor to allow access to more commands.
Format: authenticate USER_TYPE PASSWORD
USER_TYPE refers to either “tutor” or “admin”
PASSWORD is the preset password for authentication, set by the admin.

Exiting the program : exit

Exits the program.
Format: exit

The below commands are only available to users with privileges of tutor/admin.

Adding a person: add

Adds a person to the ClassRepo.
Format: add NAME [p]p/PHONE_NUMBER [p]e/EMAIL [p]a/ADDRESS [t/TAG]

Words in UPPER_CASE are the parameters, items in SQUARE_BRACKETS are optional, items with ... after them can have multiple instances. Order of parameters are fixed.

Put a p before the phone / email / address prefixes to mark it as private. private details can only be seen using the viewall command.

Persons can have any number of tags (including 0)

Examples:

- add John Doe p/98765432 e/johnd@gmail.com a/John street, block 123, #01-01`
- add Betsy Crowe pp/1234567 e/betsycrowe@gmail.com pa/Newgate Prison t/criminal t/friend`

Deleting a person : delete

Deletes the specified person from the ClassRepo. Irreversible.
Format: delete INDEX

Deletes the person at the specified `INDEX`.
The index refers to the index number shown in the most recent listing.

Examples:

list
delete 2
Deletes the 2nd person in the ClassRepo.

find Betsy
delete 1
Deletes the 1st person in the results of the find command.

View all details of a person : viewall

Displays all details (including private details) of the specified person.
Format: viewall INDEX

Views all details of the person at the specified INDEX.
The index refers to the index number shown in the most recent listing.

Examples:
list
viewall 2
Views all details of the 2nd person in the ClassRepo.

find Betsy
viewall 1
Views all details of the 1st person in the results of the find command.

Clearing all entries : clear

Clears all entries from the ClassRepo.
Format: clear

Editing entries: edit

Edits the details of a specified person
Format: edit INDEX FIELD NEW_VALUE
Views all details of the person at the specified INDEX.
The index refers to the index number shown in the most recent listing.
FIELD refers to the field to be edited
NEW_VALUE refers to the new value to replace the old field with.

Adding a person to a class: classadd

Adds a person to the class.
Format: classadd PERSON_INDEX CLASS_NAME
PERSON_INDEX indicates which person to add from the list
CLASS_NAME indicates which class to add to.
If a class with CLASS_NAME does not exist, this will create a new class and add the person to it.

Listing all persons : classlist

Shows a list of all persons in the class.
Format: classlist CLASS_NAME

Deleting a person from a class: classdelete

Deletes the specified person from the ClassRepo. Irreversible.
Format: classdelete INDEX CLASS_NAME

Deletes the person at the specified INDEX from the class with the name of CLASS_NAME.
The index refers to the index number shown in the most recent listing.

Start a new attendance log for the class: addlog

Creates a new attendance log for the day
Format: addlog DATE
DATE is the date to log the attendance as, specified in DDMMYYYY.

Mark a student’s attendance: marklog

Marks a student's attendance for the day.
Format: marklog DATE INDEX PRESENCE
DATE is the date to log the attendance as, specified in DDMMYYYY.
INDEX is the index of the student when listed in the class list.
PRESENCE is either “absent” or “present”.

Mark all students: markalllog

Marks all student's attendance for the day
Format: unmarklog DATE INDEX PRESENCE
DATE is the date to log the attendance as, specified in DDMMYYYY.
INDEX is the index of the student when listed in the class list.
PRESENCE is either “absent” or “present”.

Add exam dates : addexam

Adds a new exam dates

Format: addexam EXAM_NAME EXAM_DATE EXAM_TIME_START EXAM_TIME_END
Words in UPPER_CASE are the parameters.

Edit exam dates : editexam

Adds a new exam dates
Format: editexam EXAM_INDEX FIELD NEW_VALUE
Words in EXAM_INDEX is the index of the exam in the list
FIELD is the field to be edited
NEW_VALUE is the new value to be written to the field.

Delete exam dates : deleteexam

Deletes an exam dates
Format: deleteexam INDEX
INDEX specifies the index of the exam in the list.

Change privilege password: changepw

Authenticate the user to change the password to authenticate other users.
Format: changepw USER_TYPE OLD_PASSWORD NEW_PASSWORD
USER_TYPE refers to either “tutor” or “admin”
OLD_PASSWORD refers to the current password in use
NEW_PASSWORD refers to the new password to be used from now on to authenticate user for USER_TYPE.

Add a Student’s Grade: addgrades

Allows the user to add a student’s grades for any subject.
FORMAT: addgrades INDEX SUBJECT GRADE YEAR
INDEX: The index number of the student being selected
SUBJECT is used to specify which class the grade is for

Edit a Student’s Grade: editgrades

Allows the user to edits a student’s grades for any subject.
FORMAT: editgrades INDEX FIELD NEW_VALUE
INDEX Refers to the student whose grades have to be edited
FIELD is the field to be edited

Delete a Student’s Grade: deletegrades

Allows the user to delete a student’s grades.
FORMAT: deletegrades INDEX

INDEX Refers to the index number of the student whose grades have to be edited

View a Student's Past Grade: viewallgrades

Allows the user to view all past records of a student
FORMAT: viewallgrades INDEX
INDEX Refers to the student whose grades have to be edited

Saving the data

ClassRepo data are saved in the hard disk automatically after any command that changes the data.

There is no need to save manually. ClassRepo data are saved in a file called classrepo.txt in the project root folder.

Developer Guide

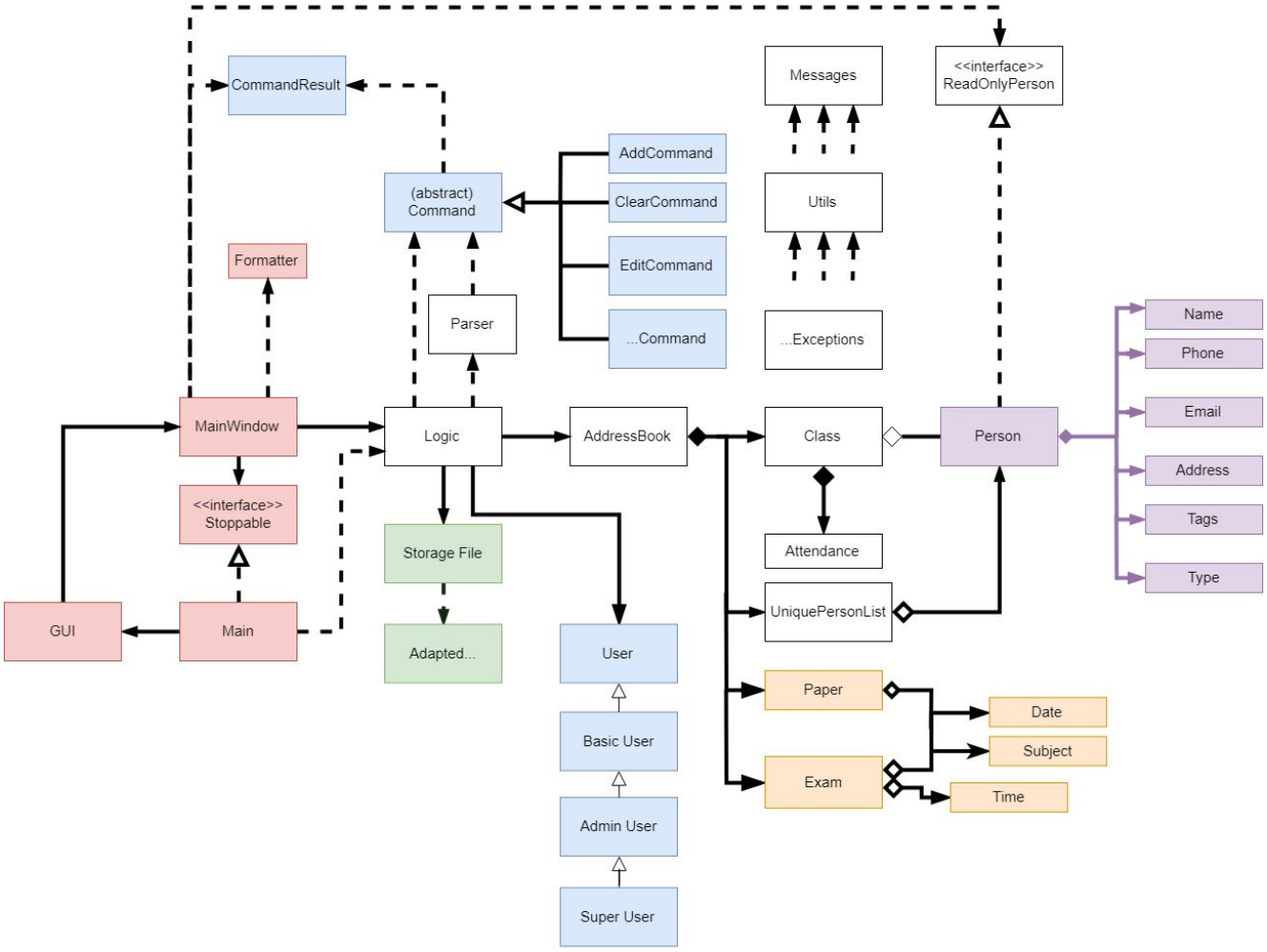
Setting Up

Prerequisites:
* JDK 9 or later
* IntelliJ IDE

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
    a. Click `Configure` > `Project Defaults` > `Project Structure`
    b. If JDK 9 is listed in the drop down, select it. If it is not, click `New...` and select the directory where you installed JDK 9
    c. Click `OK`
3. Click `Import Project`
4. Locate the `build.gradle` file and select it. Click `OK`
5. Click `Open as Project`
6. Click `OK` to accept the default settings
7. Run the `seedu.ClassRepo.Main` class (right-click the `Main` class and click `Run Main.main()`) and try executing a few commands
8. Run all the tests (right-click the `test` folder, and click `Run 'All Test's`) and ensure that they pass
9. Open the `StorageFile` file and check for any code errors
    a. Due to an ongoing https://youtrack.jetbrains.com/issue/IDEA-189060[issue] with some of the newer versions of IntelliJ, code errors may be detected even if the project can be built and run successfully
    b. To resolve this, place your cursor over any of the code section highlighted in red. Press kbd:[ALT + ENTER], and select `Add '--add-modules=java.xml.bind' to module compiler options`

Design



Testing

1. In IntelliJ, right-click on the `test` folder and choose `Run 'All Test's`

Appendix A: User Stories

Appendix B: Use Cases

Appendix C: Non-Functional Requirements

Appendix D: Glossary

[Appendix A: User Stories]

Priorities of the User Stories:

1. * = LOW (unlikely to have)
2. ** = MEDIUM (nice to have)
3. *** = HIGH (must have)

No	Priority	I am a ...	Who wants to...	So that I can...
1	***	Admin	Add in new students	Update and Manage the Database
2	***	Admin	Edit details of students	Update and Manage the Database
3	***	Admin	Delete details of students	Update and Manage the Database
4	***	Admin	View list of all students	View details of Database
5	***	Admin	Sort students by classes	Check the list of students in each class
6	***	Admin	Calculate mean score of students	Collect statistics of students
7	***	Admin	Calculate mode score of students	Collect statistics of students
8	***	Admin	Calculate median score of students	Collect statistics of students
9	***	Admin	View list of all tutors	View details of a person
10	***	Admin	Update the syllabus of subjects as per necessary	Keep up to date with the learning standards of curriculum
11	***	Admin	View students with outstanding fees	Track number of students for facilitation of payment
12	***	Admin	View exam dates	See the exam dates
13	***	Admin	Edit exam dates	Edit the exam dates
14	***	Admin	View past year question papers	Will be able to give to students as per necessary
15	***	Admin	Edit access to answers of past year question papers	Be able to give to students/tutors as required
16	***	Admin	Delete tutors	To update current roster of tutors
17	***	Admin	View leave quota of each tutor	Keep track of tutor's leave
18	***	Admin	Delete admin	Update and manage the database
19	***	Admin	Add in new tutor	Update and Manage the Database
20	***	Admin	View all past records of students	Ensure the student progresses as per intended
21	***	Admin	Add new Admin	Update and Manage the Database

22	**	Admin	View private sensitive information such as finance	View private information
23	**	Admin	Check on Tutor’s academic records	Filter out Tutors into specialized subjects
24	**	Admin	Edit details of tutors	Update and Manage the Database
25	**	Admin	Delete details of tutors	Update and Manage the Database
26	**	Admin	Undo actions	Undo my mistakes
27	**	Admin	Redo action	Revert my "undo"
28	**	Admin	Print the information	Print out the information
29	**	Admin	View tutors on leave	Manage current availability of tutors
30	*	Admin	Group students based on address location	Facilitate teacher and student paring (private tuition)
31	*	Admin	Have a list of tutors who have health insurance	Know who can claim etc.
32	*	Admin	Have list of external contacts such as maintenance, aircon etc.	Important information to run the centre
33	*	Admin	Encrypt student's data files	Restrict access
34	*	Admin	Update the calendar of the center	Reflect public holidays and weekly timings of the centre
35	*	Admin	Contact parents of student if necessary	Able to contact tutor for emergencies
36	***	Admin	Delete student	To update current list of students
37	***	Parents/ Guardian	View child's progress	Be updated about child's progress
38	***	Parents/ Guardian	Edit child's particulars	Help to keep record up to date
39	**	Parents/ Guardian	View child's homework	Be updated about child's progress
40	**	Parents/ Guardian	View child's exam grade	Be updated about child's progress
41	**	Parents/ Guardian	View tutor's credential	Be updated about child's progress
42	**	Parents/ Guardian	View centre calendar	Able to send kids when they want
43	**	Parents/ Guardian	View child's exam schedule	Be updated about child's progress
44	**	Parents/ Guardian	View any comments/ remarks made by tutor	Be updated about child's progress

45	**	Parents/ Guardian	Edit emergency contact information	Help to keep record up to date
46	**	Parents/ Guardian	Indicate subject preferences of student	Help to keep record up to date
47	*	Parents/ Guardian	Add credit card information	Make payment easier
48	***	Student	Update my particulars	Up to date contact information
49	**	Student	View my tutors remarks	Understand my errors
50	***	Student	View my tutors particulars	Understand my tutor better
51	***	Student	Undo actions	Undo my mistakes
52	***	Student	Redo action	Revert my "undo"
53	***	Student	View my attendance	Keep track of attendance to ensure minimum is met
54	***	Student	View exam dates	See the exam dates
55	***	Student	View my tutors	Know who is teaching students
56	**	Student	Declare my reason for absence	Facilitate checking of absence by tutors
57	**	Student	View tutor's timetable	Check available timing for meeting
58	**	Student	Give in feedback after every session	Be able to review the teacher and let admin know about any problems etc
59	*	Student	Block off days	Inform the center of my absence
60	*	Student	View unpaid fees	Reminder for how much the student owes
61	*	Student	View date due for fees	Know when to pay fees by
62	*	Student	Link credit card to account	Enable automatic transaction per month
63	*	Student	Be able to view answer key of past year papers	Check his answers
64	*	Student	Print the information	Print out the information
65	*	Student	Upload MC in case of sick leave	To show that absence was taken for a valid reason
66	*	Student	Change preferred mode of tutorial (solving questions given by tutor, bringing own questions etc).	Ensures a productive session
67	*	Student	Book classes with particular teachers based on availability	Attend the classes according to my convenience

68	*	Student	Indicate any specific projects or homework assignments that help is needed with	Ensure that the tutor is prepared in time
69	***	Tutor	View current class list	Know class size and students
70	***	Tutor	Edit a homework grade of a student	Manage the students end results
71	***	Tutor	Edit exam grades of a student	Manage the students work
72	***	Tutor	Delete homework grades of a student	Remove grades awarded wrongly
73	**	Tutor	View exam dates	See the exam dates
74	**	Tutor	Edit exam dates	Useful when help is needed with question
75	**	Tutor	View details of other tutors	Contact them if needed
76	*	Tutor	View list of all students	Can be informed of the number of students I teach
77	*	Tutor	View all information about any particular student	Will be view the syllabus
78	*	Tutor	Mark attendance of students	Keep track of attendance to ensure minimum is met
79	*	Tutor	View the syllabus	Will be know about the syllables
80	*	Tutor	View the homework given for each subject	view the homework assigned
81	*	Tutor	Edit the syllabus of a subject	Update the syllabus for a subject
82	*	Tutor	View his timetable	Arrange his schedule accordingly
83	*	Tutor	Edit number of hours worked	Update workload
84	*	Tutor	Add comments on submitted homework	Facilitate feedback on assignments
85	*	Tutor	View classes that he is taking	Know number of classes to take
86	*	Tutor	Undo actions	Undo my mistakes
87	*	Tutor	Redo action	Revert my "undo"
88	*	Tutor	View past year question papers	Be able to teach and show students
89	*	Tutor	Print the information	Print out the information
90	*	Tutor	Ask other tutors to take over session if needed (in case of emergency)	Ensure that the student is not left without a tutor during the session
91	*	Tutor	Edit access to answers of past year question papers	Will be able to give to students as per necessary
92	*	Admin	Delete everything	Delete everything
93	***	Admin	Sorts the whole database	Manage database

[Appendix B: Use Cases]

Note: For all use cases, the system is the ClassRepo, unless otherwise mentioned.

Actor: Tutor, Admin
Use Case: View people belonging to a subject group
MSS:

1. User requests to list people
2. ClassRepo displays the list of people
3. User requests to filter list by group
4. ClassRepo displays the list of people in the group

Extensions:

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

4a. The list is empty.
- Use case ends.

Actor: Tutor, Admin
Use Case: Edit subject details
MSS:

1. User requests to list subject
2. ClassRepo displays the list of subjects
3. User requests to edit target subject's details
4. ClassRepo requests which field to edit
5. User enters the field to edit and the new value to update with
6. ClassRepo updates the subject's details.

Extensions:

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

3a. The given index is invalid.
- 3a1. ClassRepo shows an error message.
- Use case resumes at step 2.

5a. The field to request is invalid.
- 5a1. ClassRepo shows an error message.
- Use case resumes at step 3.

5b. The new value to update with is in an invalid format.
- 5b1. ClassRepo shows an error message.
- Use case resumes at step 3.

Actor: Tutor, Admin, Student, Parent/Guardian
Use Case: View exam details

MSS:

1. User requests to list all exams of a particular subject
2. ClassRepo displays the list of exams
3. User requests to view target exam's details
4. ClassRepo displays exam's details.

Extensions:

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

3a. The given index is invalid.
- 3a1. ClassRepo shows an error message.
- Use case resumes at step 2.

Actor: Tutor, Admin
Use Case: Edit exam details
MSS

1. User requests to list exam
2. ClassRepo displays the list of exams
3. User requests to edit target exam's details
4. ClassRepo requests which field to edit
5. User enters the field to edit and the new value to update with
6. ClassRepo updates the exam's details.

Extensions

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

3a. The given index is invalid.
- 3a1. ClassRepo shows an error message.
- Use case resumes at step 2.

5a. The field to request is invalid.
- 5a1. ClassRepo shows an error message.
- Use case resumes at step 3.

5b. The new value to update with is in an invalid format.
- 5b1. ClassRepo shows an error message.
- Use case resumes at step 3.

Actor: Tutor, Admin, Student
Use Case: View past year papers
MSS:

1. User requests to list past year papers
2. ClassRepo displays the list of past year papers
3. User requests to view target past year paper's details
4. ClassRepo displays past year paper's details.

Extensions

2a. The list is empty.

- Use case ends.

3a. The given index is invalid.

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

Actors: Tutor, Admin
Use Case: Edit person particulars
MSS

1. User requests to list people
2. ClassRepo displays the list of people
3. User requests to edit target user
4. ClassRepo requests which field to edit
5. User enters the field to edit and the new value to update with
6. ClassRepo updates the field of the person

Extensions

2a. The list is empty.

- Use case ends.

3a. The given index is invalid.

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

5a. The field requested is invalid

- 5a1. ClassRepo shows an error message.
- Use case resumes at step 3.

5b. The new value to update with is in an invalid format

- 5b1. ClassRepo shows an error message.
- Use case resumes at step 3.

Actor: Tutor, Admin
Use Case: Delete person
MSS

1. User requests to list people
2. ClassRepo displays the list of people
3. User requests to delete a specific person in the list
4. ClassRepo deletes the person.

Extensions

2a. The list is empty.

- Use case ends.

3a. The given index is invalid.

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

Actor: Tutor, Admin
Use Case: Add new person

MSS

1. User requests to add Person
2. ClassRepo requests the details of the person to add
3. User enters the detail of the person
4. ClassRepo adds the person.

Extensions

3a. The details entered is of an invalid format
- 3a1. ClassRepo shows an error message.
- Use case resumes at step 2.

3b. The person to add already exists in ClassRepo.
- 3a1. ClassRepo shows an error message.
- Use case resumes at step 1.

Actor: Tutor, Admin

Use Case: View student past records

MSS

1. User requests to list students
2. ClassRepo displays the list of students
3. User requests to view target students past records
4. ClassRepo displays target students past records.

Extensions

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

3a. The given index is invalid.
- 3a1. ClassRepo shows an error message.
- Use case resumes at step 2.

Actor: Student, Parent/Guardian

Use Case: Update user's particulars

Precondition: User is logged in

MSS:

1. User requests to update their particulars
2. ClassRepo shows the fields that can be edited
3. User enters the details that they want to update
4. ClassRepo updates the respective fields

Extensions:

1a. The user login is invalid
- ClassRepo shows an error message
- Use case restarts at 1

5a: The details entered is of an invalid format
- ClassRepo displays an error message.
- Use case resumes at step 4.

Actor: Student, Parent/Guardian, Tutor, Admin
Use Case: View a tutor's particulars
MSS:

1. User requests to list all tutors
2. ClassRepo displays a list of all tutors
3. User selects a particular tutor from list
4. ClassRepo displays the selected tutor's details

Extension:

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

3a. The given index is invalid.
- ClassRepo shows an error message.
- Use case resumes at step 2.

Actor: Student, Parent/Guardian
Use Case: View user's attendance
Precondition: User is logged in
MSS:

1. User requests to view past attendance
2. ClassRepo displays the attendance record of student

Actor: Tutor, Admin
Use Case: View attendance
MSS:

1. User requests to list all students
2. ClassRepo displays the list of students
3. User requests to view target students attendance record
4. ClassRepo displays target students attendance record.

Extensions

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

3a. The given index is invalid.
- ClassRepo shows an error message.
- Use case resumes at step 2.

Actor: Tutor, Admin, Student, Parent/Guardian
Use Case: Undo/ Redo Actions
MSS:

1. User requests to go back and re-enter something
2. ClassRepo goes back to the last saved version
3. User re-enters details/ command
4. ClassRepo saves again and resumes action

Extension:

2a. Past saved version does not exist
- ClassRepo shows an error message

3a: The details entered is of an invalid format
- ClassRepo displays an error message.
- Use case resumes from step 2.

Actor: All Users
Use Case: Increase privilege
Precondition: User is logged in
MSS:
1. User requests to increase his privilege
2. ClassRepo requests the level of privilege to raise to
3. User enters privilege level
4. ClassRepo requests authentication password
5. User enters password
6. ClassRepo increases the user's privileges

Extension:

3a. The details entered is of an invalid format
- ClassRepo displays an error message.
- Use case resumes from step 2.

5a. Incorrect password entered
- ClassRepo shows an error message
- User case resumes from step 4

Actor: Tutor, Admin
Use Case: Change Password
Precondition: User is logged in
MSS:
1. User requests to change the password of a certain user type
2. ClassRepo requests the current password
3. User enters current password
4. ClassRepo requests the new password
5. User enters the new password
6. ClassRepo changes the password to the new one

Extension:

1a. User type entered is invalid
- ClassRepo displays an error message.
- Use case ends.

3a. Incorrect password entered
- ClassRepo shows an error message
- User case resumes from step 2

5a. New password in invalid format

- ClassRepo shows an error message
- User case resumes from step 4

5b. New password is the same as old password
- ClassRepo shows an error message
- User case resumes from step 4

Actor: Tutor, Admin
Use Case: Assign a person to a class
MSS

1. User requests to assign person to a class
2. ClassRepo displays list of people
3. User select the person
4. ClassRepo adds the person to the class.

Extensions

1a. Class to assign to does not exists
- 1a1. ClassRepo shows an error message.
- Use case ends

3a. The person to add already exists in the class.
- 3a1. ClassRepo shows an error message.
- Use case resumes at step 2.

3b. The specified index is invalid.
- 3b1. ClassRepo shows an error message.
- Use case resumes at step 2.

Actor: Tutor, Admin
Use Case: Remove a person from a class
MSS

1. User requests to remove person from a class
2. ClassRepo displays list of people
3. User select the person
4. ClassRepo removes the person from the class.

Extensions

1a. Class to remove from does not exists
- 1a1. ClassRepo shows an error message.
- Use case ends

3a. The specified index is invalid.
- 3a1. ClassRepo shows an error message.
- Use case resumes at step 2.
-

Actor: Tutor, Admin
Use Case: View student past records
MSS

1. User requests to list students

2. ClassRepo displays the list of students
3. User requests to view target students past records
4. ClassRepo displays target students past records.

Extensions

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

3a. The given index is invalid.
- 3a1. ClassRepo shows an error message.
- Use case resumes at step 2.

Actor: Admin
Use Case: Sort student into classes
MSS

1. User requests to list students
2. ClassRepo displays the list of students
3. User requests to sort by classes
4. ClassRepo displays the students by classes

Actor: Admin
Use Case: Calculate mean, mode and median score of students
MSS

1. User requests to list all students
2. ClassRepo displays the list of students
3. User requests to calculate mean/ mode/ median as needed
4. ClassRepo displays required statistics

[Appendix C: Non-functional Requirements]

1. Should work on any mainstream OS as long as it has Java 9 or higher installed.
2. Should be able to hold up to 1000 persons without a noticeable sluggishness in performance for typical usage.
3. A user with above average typing speed for regular English text (i.e. not code, not system admin commands) should be able to accomplish most of the tasks faster using commands than using the mouse.
4. Should work at decent speed even with below average PC specifications.
5. Only write to file when necessary

[Appendix D: Glossary]

Mainstream OS: Windows, Linux, Unix, OS-X

Private contact details: A contact detail that is not meant to be shared with others

Draft of the plan for project:

Version	***Deadline***		***Features Included***				
			Attendance System	Exam System	Authentication	Grade System	Finance System
		Owner:	Bo Kai	Mick	Tek In	Meghana	Lucas
v1.1	Week 7		Allow viewing of a student's overall attendance	Ability to add in a new tuition exam details	Allow increasing privileges	Add new grades to any student	Keying in of respective dues for staff and students by admin
v1.2	Week 9		Allow adding a student's attendance to a class session	Ability to edit and delete tuition exam timings	Different privileges have different commands	Edit and delete grades of specific student	Adds tracking of list of people w overdue fees
v1.3	Week 11		Allow editing a student's attendance to a class session	Ability for admin to assign student a tuition exam	Allow password changing and storage	View all past records of student	Lets students and staff view their own fees
v1.4	Week 13		Allow viewing class attendance	Ability for student to view details of all his exams(or any single one)		Be able to calculate mean, median etc. of grades to determine toppers	