

Schedule Management System

Generated by Doxygen 1.9.4

1 Schedule Management System	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 AddRequest Class Reference	9
5.1.1 Detailed Description	9
5.1.2 Constructor & Destructor Documentation	10
5.1.2.1 AddRequest()	10
5.1.3 Member Function Documentation	10
5.1.3.1 getClassCode()	10
5.1.3.2 getStudentID()	10
5.1.3.3 getUCCode()	11
5.1.3.4 setClassCode()	11
5.1.3.5 setStudentID()	11
5.1.3.6 setUCCode()	11
5.2 ControlUnit Class Reference	12
5.2.1 Detailed Description	13
5.2.2 Member Function Documentation	13
5.2.2.1 CheckAdd()	13
5.2.2.2 CheckRemove()	14
5.2.2.3 CheckSwitch()	14
5.2.2.4 classStudents()	15
5.2.2.5 clearMemory()	15
5.2.2.6 courseStudents()	15
5.2.2.7 createAdd()	15
5.2.2.8 createRemove()	16
5.2.2.9 createSwitch()	16
5.2.2.10 DisplayClassSchedule()	16
5.2.2.11 DisplayStudentSchedule()	16
5.2.2.12 formatConflicts()	16
5.2.2.13 getClassInUc()	17
5.2.2.14 IsThereConflict()	17
5.2.2.15 LoadCSV()	17
5.2.2.16 maxSgSize()	18
5.2.2.17 NumBalanced()	18
5.2.2.18 processAddRequest()	18

5.2.2.19 processAllRequests()	19
5.2.2.20 processRemoveRequest()	19
5.2.2.21 processRequest()	19
5.2.2.22 processSwitchRequest()	20
5.2.2.23 removeLastPendingRequest()	20
5.2.2.24 saveChanges()	20
5.2.2.25 setCap()	20
5.2.2.26 StudentsInAtLeastNUcs()	20
5.2.2.27 StudentsInAtMostNUcs()	21
5.2.2.28 StudentsInExactNUcs()	21
5.2.2.29 UCWithMostStudents()	22
5.2.2.30 undoRequest()	22
5.2.2.31 yearStudents()	22
5.3 lesson Class Reference	23
5.3.1 Detailed Description	23
5.3.2 Constructor & Destructor Documentation	23
5.3.2.1 lesson()	23
5.3.3 Member Function Documentation	24
5.3.3.1 getDuration()	24
5.3.3.2 getEndTime()	24
5.3.3.3 getStartTime()	24
5.3.3.4 getType()	25
5.3.3.5 getUccode()	25
5.3.3.6 getWeekday()	25
5.4 lesstime Class Reference	25
5.4.1 Detailed Description	26
5.4.2 Constructor & Destructor Documentation	26
5.4.2.1 lesstime() [1/2]	26
5.4.2.2 lesstime() [2/2]	26
5.4.3 Member Function Documentation	27
5.4.3.1 displayHourFormat()	27
5.4.3.2 getHour()	27
5.4.3.3 getMinute()	27
5.5 Menu Class Reference	28
5.5.1 Detailed Description	28
5.5.2 Member Function Documentation	29
5.5.2.1 optionStudentMenu()	29
5.5.2.2 SeeStudentsInClass()	29
5.5.2.3 SeeStudentsInUc()	29
5.5.2.4 SeeStudentsInYear()	29
5.6 RemoveRequest Class Reference	30
5.6.1 Detailed Description	30

5.6.2 Constructor & Destructor Documentation	30
5.6.2.1 RemoveRequest()	30
5.6.3 Member Function Documentation	31
5.6.3.1 getClassCode()	31
5.6.3.2 getStudentID()	31
5.6.3.3 getUCCode()	31
5.6.3.4 setClassCode()	31
5.6.3.5 setStudentID()	32
5.6.3.6 setUCCode()	32
5.7 Request Class Reference	32
5.7.1 Detailed Description	33
5.7.2 Constructor & Destructor Documentation	33
5.7.2.1 Request()	33
5.7.3 Member Function Documentation	33
5.7.3.1 getType()	33
5.8 Schedule Class Reference	34
5.8.1 Detailed Description	34
5.8.2 Constructor & Destructor Documentation	34
5.8.2.1 Schedule()	34
5.9 Student Class Reference	34
5.9.1 Detailed Description	35
5.9.2 Constructor & Destructor Documentation	35
5.9.2.1 Student()	35
5.9.3 Member Function Documentation	36
5.9.3.1 addStudentGroup()	36
5.9.3.2 getName()	36
5.9.3.3 getStudentGroups()	36
5.9.3.4 getStudentID()	36
5.9.3.5 isInClass()	37
5.9.3.6 isInUC()	37
5.9.3.7 removeGroup()	37
5.9.3.8 setName()	38
5.9.3.9 setStudentID()	38
5.10 studentGroup Class Reference	38
5.10.1 Detailed Description	39
5.10.2 Constructor & Destructor Documentation	39
5.10.2.1 studentGroup()	39
5.10.3 Member Function Documentation	39
5.10.3.1 getClassCode()	39
5.10.3.2 getUcCode()	39
5.11 SwitchRequest Class Reference	40
5.11.1 Detailed Description	40

5.11.2 Constructor & Destructor Documentation	40
5.11.2.1 SwitchRequest()	40
5.11.3 Member Function Documentation	41
5.11.3.1 getClassCode1()	41
5.11.3.2 getClassCode2()	41
5.11.3.3 getStudentID()	41
5.11.3.4 getUCCode1()	42
5.11.3.5 getUCCode2()	42
6 File Documentation	43
6.1 src/AddRequest.h File Reference	43
6.2 AddRequest.h	43
6.3 src/ControlUnit.h File Reference	44
6.4 ControlUnit.h	44
6.5 src/lesson.h File Reference	46
6.6 lesson.h	46
6.7 src/lesstime.h File Reference	47
6.8 lesstime.h	47
6.9 src/Menu.h File Reference	48
6.10 Menu.h	48
6.11 src/RemoveRequest.h File Reference	49
6.12 RemoveRequest.h	49
6.13 src/Request.h File Reference	50
6.14 Request.h	50
6.15 src/Schedule.h File Reference	50
6.16 Schedule.h	51
6.17 src/student.h File Reference	51
6.18 student.h	51
6.19 src/studentGroup.h File Reference	52
6.20 studentGroup.h	52
6.21 src/SwitchRequest.h File Reference	53
6.22 SwitchRequest.h	53
Index	55

Chapter 1

Schedule Management System

Project made by:

- Henrique Fernandes 202204988
- José Sousa 202208817
- Leandro Martins 202208001

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ControlUnit	12
lesson	23
lessonTime	25
Menu	28
Request	32
AddRequest	9
RemoveRequest	30
SwitchRequest	40
Schedule	34
Student	34
studentGroup	38

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AddRequest		
Request of type add	9
ControlUnit		
Class used to handle the core functions of the program	12
lesson		
Class used to represent a lesson from a course	23
lesstime		
Class used to represent time	25
Menu		
Class used to represent the menu the user uses to navigate	28
RemoveRequest		
Request of type remove	30
Request		
Class used to represent a generic request	32
Schedule		
Class used to display a schedule	34
Student		
Class used to represent a student	34
studentGroup		
Class used to represent a class (group of students)	38
SwitchRequest		
Request of type switch	40

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

src/ AddRequest.h	43
src/ ControlUnit.h	44
src/ lesson.h	46
src/ lesstime.h	47
src/ Menu.h	48
src/ RemoveRequest.h	49
src/ Request.h	50
src/ Schedule.h	50
src/ student.h	51
src/ studentGroup.h	52
src/ SwitchRequest.h	53

Chapter 5

Class Documentation

5.1 AddRequest Class Reference

[Request](#) of type add.

```
#include <AddRequest.h>
```

Inheritance diagram for AddRequest:

Collaboration diagram for AddRequest:

Public Member Functions

- [AddRequest](#) (const std::string &studentID, const std::string &ucCode, const std::string &classCode)
Parameterized constructor.
- std::string [getStudentID](#) () const
Gets the student ID.
- std::string [getUCCode](#) () const
Gets the course code.
- std::string [getClassCode](#) () const
Gets the class code.
- void [setStudentID](#) (const std::string &studentID)
Sets a new student ID.
- void [setUCCode](#) (const std::string &ucCode)
Sets a new course code.
- void [setClassCode](#) (const std::string &classCode)
Sets a new class code.

Additional Inherited Members

5.1.1 Detailed Description

[Request](#) of type add.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AddRequest()

```
AddRequest::AddRequest (
    const std::string & studentID,
    const std::string & ucCode,
    const std::string & classCode )
```

Parameterized constructor.

Parameters

<i>studentID</i>	String representing the student ID.
<i>ucCode</i>	String representing the course code.
<i>classCode</i>	String representing the class code.

5.1.3 Member Function Documentation

5.1.3.1 getClassCode()

```
std::string AddRequest::getClassCode ( ) const
```

Gets the class code.

Returns

A string representing the class code.

Here is the caller graph for this function:

5.1.3.2 getStudentID()

```
std::string AddRequest::getStudentID ( ) const
```

Gets the student ID.

Returns

A string representing the studentID.

Here is the caller graph for this function:

5.1.3.3 getUCCode()

```
std::string AddRequest::getUCCode ( ) const
```

Gets the course code.

Returns

A string representing the course code.

Here is the caller graph for this function:

5.1.3.4 setClassCode()

```
void AddRequest::setClassCode (
    const std::string & classCode )
```

Sets a new class code.

Parameters

<i>classCode</i>	The new class code.
------------------	---------------------

5.1.3.5 setStudentID()

```
void AddRequest::setStudentID (
    const std::string & studentID )
```

Sets a new student ID.

Parameters

<i>studentID</i>	The new student ID.
------------------	---------------------

5.1.3.6 setUCCode()

```
void AddRequest::setUCCode (
    const std::string & ucCode )
```

Sets a new course code.

Parameters

<i>ucCode</i>	The new course code.
---------------	----------------------

The documentation for this class was generated from the following files:

- src/AddRequest.h
- src/AddRequest.cpp

5.2 ControlUnit Class Reference

Class used to handle the core functions of the program.

```
#include <ControlUnit.h>
```

Public Member Functions

- void **LoadCSV** (string studentFilename)
Loads all the csv files.
- void **LoadClassesCSV** ()
Loads the classes.csv file (which has all the lessons).
- void **LoadClassesPerUcCSV** ()
Loads the classes_per_uc.csv file (which has all the courses and classes).
- void **LoadStudentsClassesCSV** ()
Load students_classes.csv or student_classes_updated.csv, depending on the option chose.
- vector< vector< **lesson** > > **formatConflicts** (vector< **lesson** > &lessons)
Deals with overlaps in a schedule.
- int **maxSgSize** ()
Returns maximum size of all classes.
- void **setCap** (int n)
Set the class capacity.
- void **saveChanges** ()
Saves the changes made, updating the file students_classes_updated.csv.
- void **DisplayStudentSchedule** ()
Displays the schedule of a student.
- void **DisplayClassSchedule** ()
Displays the schedule of a class.
- int **StudentsInAtLeastNUcs** (int n)
Displays the students enrolled in at least N courses.
- int **StudentsInAtMostNUcs** (int n)
Displays the students enrolled in at most N courses.
- int **StudentsInExactNUcs** (int n)
Displays the students enrolled in exactly N courses.
- void **courseStudents** (string courseCode, function< bool(**Student**, **Student**)> func)
Displays the students enrolled in a specific course.
- void **yearStudents** (char year, function< bool(**Student**, **Student**)> func)
Displays the students from a specific year.
- void **classStudents** (string classCode, function< bool(**Student**, **Student**)> func)
Displays the students from a specific class.
- void **UCWithMostStudents** ()
Displays all the courses starting with the one with the most student.
- int **NumBalanced** (vector< **studentGroup** >, map< MainKey, int >)

- Checks the balance of the classes.*

 - bool `IsThereConflict` (vector< `lesson` >)

Detects conflicts in a schedule.
- bool `processRequest` (`Request` *request, bool bypassStack=false)

Processes a request.
- void `processAddRequest` (`AddRequest` *addRequest)

Processes a request of type add.
- void `processRemoveRequest` (`RemoveRequest` *removeRequest)

Processes a request of type remove.
- void `processSwitchRequest` (`SwitchRequest` *switchRequest)

Processes a request of type switch.
- void `processAllRequests` ()

Processes all the requests awaiting to be processed.
- void `removeLastPendingRequest` ()

Removes the most recent request that hasn't been applied.
- void `undoRequest` (int n)

Undoes the N most recent applied request.
- void `createAdd` ()

Creates a request of type add.
- void `createRemove` ()

Creates a request of type remove.
- void `createSwitch` ()

Creates a request of type switch.
- bool `CheckAdd` (`AddRequest` *addrq)

Checks if the request is possible.
- bool `CheckRemove` (`RemoveRequest` *remrq)

Checks if the request is possible.
- bool `CheckSwitch` (`SwitchRequest` *swrq)

Checks if the request is possible.
- string `getClassInUc` (string studentID, string ucCode)

Gets the class of a student knowing the course.
- void `clearMemory` ()

Frees all the dynamic memory.

5.2.1 Detailed Description

Class used to handle the core functions of the program.

5.2.2 Member Function Documentation

5.2.2.1 CheckAdd()

```
bool ControlUnit::CheckAdd (
    AddRequest * addrq )
```

Checks if the request is possible.

Complexity is $O(\log n * k + m)$ where n is the amount of students, k is the amount of classes of that student and m is the amount of classes related to the course.

Parameters

<i>addrq</i>	Request to be analysed.
--------------	---

Returns

Boolean representing if the request is possible or not.

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.2 CheckRemove()

```
bool ControlUnit::CheckRemove (
    RemoveRequest * remrq )
```

Checks if the request is possible.

Complexity is $O(\log n * k)$ where n is the amount of students and k is the amount of classes of that student.

Parameters

<i>remrq</i>	Request to be analysed.
--------------	---

Returns

Boolean representing if the request is possible or not.

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.3 CheckSwitch()

```
bool ControlUnit::CheckSwitch (
    SwitchRequest * swrq )
```

Checks if the request is possible.

Complexity is $O(\log n)$ where n is the amount of students.

Parameters

<i>swrq</i>	Request to be analysed.
-------------	---

Returns

Boolean representing if the request is possible or not.

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.4 classStudents()

```
void ControlUnit::classStudents (
    string classCode,
    function< bool(Student, Student)> func )
```

Displays the students from a specific class.

Complexity is $O(n * k)$ where n is the number of students and k is the number of classes for each student

Parameters

<i>classCode</i>	String representing the class code.
<i>func</i>	Boolean function used to sort students.

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.5 clearMemory()

```
void ControlUnit::clearMemory ( )
```

Frees all the dynamic memory.

Complexity is $O(n)$ where n is the amount of classes plus the amount of lessons. Here is the caller graph for this function:

5.2.2.6 courseStudents()

```
void ControlUnit::courseStudents (
    string courseCode,
    function< bool(Student, Student)> func )
```

Displays the students enrolled in a specific course.

Complexity is $O(n * k)$ where n is the number of students and k is the number of classes for each student

Parameters

<i>courseCode</i>	String representing the course code.
<i>func</i>	Boolean function used to sort students.

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.7 createAdd()

```
void ControlUnit::createAdd ( )
```

Creates a request of type add.

Complexity is $O(\log n)$ where n is the amount of students. Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.8 createRemove()

```
void ControlUnit::createRemove ( )
```

Creates a request of type remove.

Complexity is $O(\log n)$ where n is the amount of students. Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.9 createSwitch()

```
void ControlUnit::createSwitch ( )
```

Creates a request of type switch.

Complexity is $O(\log^2(n) * k)$ where n is the amount of students and k is the amount of classes of that student. Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.10 DisplayClassSchedule()

```
void ControlUnit::DisplayClassSchedule ( )
```

Displays the schedule of a class.

Getting the schedule is $O(n)$ where n is the total number of studentGroups. Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.11 DisplayStudentSchedule()

```
void ControlUnit::DisplayStudentSchedule ( )
```

Displays the schedule of a student.

Getting the schedule is $O(\log n + m * k)$ where n is the total number of students, m is the number of student groups each student has k is the lessons in a student group. Basically $O(\log n)$ as the other input is almost constant for each student and very small compared to n . Additionally after getting the schedule it has to be displayed and that takes time which is not taken into consideration given its small impact on the overall performance. Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.12 formatConflicts()

```
vector< vector< lesson > > ControlUnit::formatConflicts (
    vector< lesson > & lessons )
```

Deals with overlaps in a schedule.

Complexity is $O(n^2)$ where n is the number of lessons given as input.

Parameters

<i>lessons</i>	Vector with all the lessons.
----------------	------------------------------

Returns

A 2d vector with the conflicts.

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.13 getClassInUc()

```
string ControlUnit::getClassInUc (
    string studentID,
    string ucCode )
```

Gets the class of a student knowing the course.

Complexity is $O(\log n * k)$ where n is the amount of students and k is the amount of classes of that student.

Parameters

<i>studentID</i>	String representing the student ID.
<i>ucCode</i>	String representing the course code.

Returns

String representing the class code.

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.14 IsThereConflict()

```
bool ControlUnit::IsThereConflict (
    vector< lesson > lessons )
```

Detects conflicts in a schedule.

Complexity is $O(n^2)$ where n is the number of lessons given as input

Returns

Boolean that represents the existence of conflicts.

Here is the caller graph for this function:

5.2.2.15 LoadCSV()

```
void ControlUnit::LoadCSV (
    string studentFilename )
```

Loads all the csv files.

Parameters

<i>studentFilename</i>	A string that represents the student csv, it can either be the original version or the updated version.
------------------------	---

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.16 maxSgSize()

```
int ControlUnit::maxSgSize ( )
```

Returns maximum size of all classes.

Complexity is $O(n)$ where n is the number of student groups.

Returns

Integer representing the maximum size of a [studentGroup](#).

Here is the caller graph for this function:

5.2.2.17 NumBalanced()

```
int ControlUnit::NumBalanced (
    vector< studentGroup > groups,
    map< MainKey, int > myMap )
```

Checks the balance of the classes.

Complexity is $O(n)$ where n is the student groups given as input

Returns

Returns the maximum difference between the amount of students in each class.

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.18 processAddRequest()

```
void ControlUnit::processAddRequest (
    AddRequest * addRequest )
```

Processes a request of type add.

Complexity is $O(\log n)$ where n is the amount of students.

Parameters

<i>addRequest</i>	The request to be processed.
-------------------	------------------------------

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.19 processAllRequests()

```
void ControlUnit::processAllRequests ( )
```

Processes all the requests awaiting to be processed.

Complexity is $O(\log n * k)$ where n is the amount of students and k is the amount of requests awaiting to be processed.. Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.20 processRemoveRequest()

```
void ControlUnit::processRemoveRequest (
    RemoveRequest * removeRequest )
```

Processes a request of type remove.

Complexity is $O(\log n)$ where n is the amount of students.

Parameters

<i>removeRequest</i>	The request to be processed.
----------------------	------------------------------

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.21 processRequest()

```
bool ControlUnit::processRequest (
    Request * request,
    bool bypassStack = false )
```

Processes a request.

Parameters

<i>request</i>	Request to be processed.
<i>bypassStack</i>	A boolean that states if the request should bypass the stack (true if the request is an undo of a previous request).

Returns

Boolean that represents if the request was processed successfully.

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.22 processSwitchRequest()

```
void ControlUnit::processSwitchRequest (
    SwitchRequest * switchRequest )
```

Processes a request of type switch.

Complexity is $O(\log n)$ where n is the amount of students.

Parameters

<i>switchRequest</i>	The request to be processed.
----------------------	------------------------------

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.23 removeLastPendingRequest()

```
void ControlUnit::removeLastPendingRequest ( )
```

Removes the most recent request that hasn't been applied.

Complexity is $O(n)$ where n is the amount of requests awaiting to be processed.. Here is the caller graph for this function:

5.2.2.24 saveChanges()

```
void ControlUnit::saveChanges ( )
```

Saves the changes made, updating the file students_classes_updated.csv.

Complexity is $O(n * k)$ where n is the number of students and k is the number of classes for each student Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.25 setCap()

```
void ControlUnit::setCap (
    int n )
```

Set the class capacity.

Parameters

<i>n</i>	Integer representing the new maximum capacity.
----------	--

Here is the caller graph for this function:

5.2.2.26 StudentsInAtLeastNUcs()

```
int ControlUnit::StudentsInAtLeastNUcs (
```

```
int n )
```

Displays the students enrolled in at least N courses.

Complexity is $O(n)$ where n is the total number of students.

Parameters

n	Integer representing the minimum amount of courses.
-----	---

Returns

Integer representing the amount of students enrolled in at least N courses.

Here is the caller graph for this function:

5.2.2.27 StudentsInAtMostNUcs()

```
int ControlUnit::StudentsInAtMostNUcs (  
    int n )
```

Displays the students enrolled in at most N courses.

Complexity is $O(n)$ where n is the total number of students.

Parameters

n	Integer representing the maximum amount of courses.
-----	---

Returns

Integer representing the amount of students enrolled in at most N courses.

Here is the caller graph for this function:

5.2.2.28 StudentsInExactNUcs()

```
int ControlUnit::StudentsInExactNUcs (  
    int n )
```

Displays the students enrolled in exactly N courses.

Complexity is $O(n)$ where n is the total number of students.

Parameters

n	Integer representing the amount of courses.
-----	---

Returns

Integer representing the amount of students enrolled in N courses.

Here is the caller graph for this function:

5.2.2.29 UCWithMostStudents()

```
void ControlUnit::UCWithMostStudents ( )
```

Displays all the courses starting with the one with the most student.

Complexity is $O(n * k)$ where n is the number of students and k is the number of classes for each student Here is the caller graph for this function:

5.2.2.30 undoRequest()

```
void ControlUnit::undoRequest (
    int n )
```

Undoes the N most recent applied request.

Complexity is $O(\log n * k)$ where n is the amount of students and k is the amount of requests that need to be undone..

Parameters

<i>n</i>	Integer representing how many requests should be undone.
----------	--

Here is the call graph for this function: Here is the caller graph for this function:

5.2.2.31 yearStudents()

```
void ControlUnit::yearStudents (
    char year,
    function< bool(Student, Student)> func )
```

Displays the students from a specific year.

Complexity is $O(n * k)$ where n is the number of students and k is the number of classes for each student

Parameters

<i>year</i>	Char representing the year,
<i>func</i>	Boolean function used to sort students.

Here is the call graph for this function: Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- src/[ControlUnit.h](#)
- src/[ControlUnit.cpp](#)

5.3 lesson Class Reference

Class used to represent a lesson from a course.

```
#include <lesson.h>
```

Public Member Functions

- [lesson](#) (const std::string &ucCode, const std::string &[studentGroup](#), const std::string &weekday, double startTime, double duration, const std::string &type)
Parameterized Constructor.
- const std::string & [getWeekday](#) () const
Gets the lesson's weekday.
- const [lessonime](#) & [getStartTime](#) () const
Gets the time the lesson starts.
- const [lessonime](#) & [getDuration](#) () const
Gets the duration of the lesson.
- const [lessonime](#) & [getEndTime](#) () const
Gets the time the lesson ends.
- const string & [getUccode](#) () const
Gets the course code of the lesson.
- const std::string & [getType](#) () const
Gets the type of the lesson.

5.3.1 Detailed Description

Class used to represent a lesson from a course.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 [lesson\(\)](#)

```
lesson::lesson (  
    const std::string & ucCode,  
    const std::string & studentGroup,  
    const std::string & weekday,  
    double startTime,  
    double duration,  
    const std::string & type )
```

Parameterized Constructor.

Parameters

<i>ucCode</i>	String representing the course.
<i>studentGroup</i>	String representing the class.
<i>weekday</i>	String representing the weekday.
<i>startTime</i>	The time the lesson starts.
<i>duration</i>	The duration of the lesson.
<i>type</i>	The type of the lesson.

5.3.3 Member Function Documentation**5.3.3.1 getDuration()**

```
const lesstime & lesson::getDuration ( ) const
```

Gets the duration of the lesson.

Returns

The duration of the lesson.

Here is the caller graph for this function:

5.3.3.2 getEndTime()

```
const lesstime & lesson::getEndTime ( ) const
```

Gets the time the lesson ends.

Returns

The time the lesson ends.

5.3.3.3 getStartTime()

```
const lesstime & lesson::getStartTime ( ) const
```

Gets the time the lesson starts.

Returns

The time the lesson starts.

Here is the caller graph for this function:

5.3.3.4 getType()

```
const std::string & lesson::getType ( ) const
```

Gets the type of the lesson.

Returns

A string representing the type of the lesson.

Here is the caller graph for this function:

5.3.3.5 getUcode()

```
const std::string & lesson::getUcode ( ) const
```

Gets the course code of the lesson.

Returns

A string representing the course code.

Here is the caller graph for this function:

5.3.3.6 getWeekday()

```
const std::string & lesson::getWeekday ( ) const
```

Gets the lesson's weekday.

Returns

A string representing the weekday.

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- [src/lesson.h](#)
- [src/lesson.cpp](#)

5.4 lesstime Class Reference

Class used to represent time.

```
#include <lesstime.h>
```

Public Member Functions

- [lesstime](#) (double time)
Copy constructor.
- **lesstime** ()
Default constructor (00:00)
- [lesstime](#) (int hour, int minutes)
Parameterized constructor.
- string [displayHourFormat](#) () const
Converts the time to a string.
- int [getHour](#) () const
Hour getter.
- int [getMinute](#) () const
Minutes getter.

5.4.1 Detailed Description

Class used to represent time.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 lesstime() [1/2]

```
lesstime::lesstime (
    double time ) [explicit]
```

Copy constructor.

Parameters

<i>time</i>	
-------------	--

5.4.2.2 lesstime() [2/2]

```
lesstime::lesstime (
    int hour,
    int minutes )
```

Parameterized constructor.

Parameters

<i>hour</i>	
<i>minutes</i>	

5.4.3 Member Function Documentation

5.4.3.1 displayHourFormat()

```
std::string lessontime::displayHourFormat ( ) const
```

Converts the time to a string.

Returns

A string representing the time.

Here is the caller graph for this function:

5.4.3.2 getHour()

```
int lessontime::getHour ( ) const
```

Hour getter.

Returns

An integer representing the hour.

Here is the caller graph for this function:

5.4.3.3 getMinute()

```
int lessontime::getMinute ( ) const
```

Minutes getter.

Returns

An integer representing the minutes.

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- [src/lessontime.h](#)
- [src/lessontime.cpp](#)

5.5 Menu Class Reference

Class used to represent the menu the user uses to navigate.

```
#include <Menu.h>
```

Public Member Functions

- void **createMenu** ()
Creates the menu.
- void **SeeStudentSchedule** ()
Displays the schedule of a student.
- void **SeeClassSchedule** ()
Displays the schedule of a class.
- void **SeeNumStudentsInExactNUCs** ()
Displays the student enrolled in exactly N courses.
- void **SeeNumStudentsAtLeastNUCs** ()
Displays the students enrolled in at least N courses.
- void **SeeNumStudentsAtMostNUCs** ()
Displays the student enrolled in at most N courses.
- void **SeeUcFromMostStudents** ()
Displays all the courses starting with the one with the most student.
- void **SeeNumStudentsInNUCs** ()
Enters the submenu for listing the students in courses.
- void **listingMenu** ()
Enters the listing menu, which allows the user to list students, see schedules etc.
- void **requestMenu** ()
Enters the request menu, which allows the user to create, delete and manage requests.
- void **scheduleMenu** ()
Enters the schedule menu, which allows the user to see the schedule for a student or a class.
- void **studentMenu** ()
Enters the student menu, which allows the user to see all students from a year, course or class.
- void **SeeStudentsInUc** (function< bool([Student](#), [Student](#))> comp)
Lists all the students in a specific course.
- void **SeeStudentsInYear** (function< bool([Student](#), [Student](#))> comp)
Lists all the students in a specific year.
- void **SeeStudentsInClass** (function< bool([Student](#), [Student](#))> comp)
Lists all the students in a specific class.
- void **createRequest** ()
Enters the menu for creating request, allowing users to add, remove or switch classes.
- function< bool([Student](#), [Student](#))> **optionStudentMenu** ()
Allows the user to select different sorting options for displaying the students.

5.5.1 Detailed Description

Class used to represent the menu the user uses to navigate.

5.5.2 Member Function Documentation

5.5.2.1 optionStudentMenu()

```
function< bool(Student, Student)> Menu::optionStudentMenu ( )
```

Allows the user to select different sorting options for displaying the students.

Returns

A boolean function the compares students.

Here is the call graph for this function: Here is the caller graph for this function:

5.5.2.2 SeeStudentsInClass()

```
void Menu::SeeStudentsInClass (
    function< bool(Student, Student)> comp )
```

Lists all the students in a specific class.

Parameters

<i>comp</i>	A boolean function that compares the students, allowing the program to list the students in different ways.
-------------	---

Here is the call graph for this function: Here is the caller graph for this function:

5.5.2.3 SeeStudentsInUc()

```
void Menu::SeeStudentsInUc (
    function< bool(Student, Student)> comp )
```

Lists all the students in a specific course.

Parameters

<i>comp</i>	A boolean function that compares the students, allowing the program to list the students in different ways.
-------------	---

Here is the call graph for this function: Here is the caller graph for this function:

5.5.2.4 SeeStudentsInYear()

```
void Menu::SeeStudentsInYear (
    function< bool(Student, Student)> comp )
```

Lists all the students in a specific year.

Parameters

<i>comp</i>	A boolean function that compares the students, allowing the program to list the students in different ways.
-------------	---

Here is the call graph for this function: Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- src/Menu.h
- src/Menu.cpp

5.6 RemoveRequest Class Reference

[Request](#) of type remove.

```
#include <RemoveRequest.h>
```

Inheritance diagram for RemoveRequest:

Collaboration diagram for RemoveRequest:

Public Member Functions

- [RemoveRequest](#) (const std::string &studentID, const std::string &ucCode, const std::string &classCode)
Parameterized constructor.
- std::string [getStudentID](#) () const
Gets the student ID.
- std::string [getUCCode](#) () const
Gets the course code.
- std::string [getClassCode](#) () const
Gets the class code.
- void [setStudentID](#) (const std::string &studentID)
Sets a new student ID.
- void [setUCCode](#) (const std::string &ucCode)
Sets a new course code.
- void [setClassCode](#) (const std::string &classCode)
Sets a new class code.

Additional Inherited Members

5.6.1 Detailed Description

[Request](#) of type remove.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 RemoveRequest()

```
RemoveRequest::RemoveRequest (
    const std::string & studentID,
    const std::string & ucCode,
    const std::string & classCode )
```

Parameterized constructor.

Parameters

<i>studentID</i>	String representing the student ID.
<i>ucCode</i>	String representing the course code.
<i>classCode</i>	String representing the class code.

5.6.3 Member Function Documentation

5.6.3.1 getClassCode()

```
std::string RemoveRequest::getClassCode ( ) const
```

Gets the class code.

Returns

A string representing the class code.

Here is the caller graph for this function:

5.6.3.2 getStudentID()

```
std::string RemoveRequest::getStudentID ( ) const
```

Gets the student ID.

Returns

A string representing the studentID.

Here is the caller graph for this function:

5.6.3.3 getUCCode()

```
std::string RemoveRequest::getUCCode ( ) const
```

Gets the course code.

Returns

A string representing the course code.

Here is the caller graph for this function:

5.6.3.4 setClassCode()

```
void RemoveRequest::setClassCode (
    const std::string & classCode )
```

Sets a new class code.

Parameters

<i>classCode</i>	The new class code.
------------------	---------------------

5.6.3.5 setStudentID()

```
void RemoveRequest::setStudentID (
    const std::string & studentID )
```

Sets a new student ID.

Parameters

<i>studentID</i>	The new student ID.
------------------	---------------------

5.6.3.6 setUCCode()

```
void RemoveRequest::setUCCode (
    const std::string & ucCode )
```

Sets a new course code.

Parameters

<i>ucCode</i>	The new course code.
---------------	----------------------

The documentation for this class was generated from the following files:

- [src/RemoveRequest.h](#)
- [src/RemoveRequest.cpp](#)

5.7 Request Class Reference

Class used to represent a generic request.

```
#include <Request.h>
```

Inheritance diagram for Request:

Public Member Functions

- [Request](#) (std::string type)
Parameterized constructor.
- std::string [getType](#) () const
Gets the type of the request.

Static Public Member Functions

- static void **resetCount** ()
Resets the request counter.

5.7.1 Detailed Description

Class used to represent a generic request.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Request()

```
Request::Request (
    std::string type ) [inline]
```

Parameterized constructor.

Parameters

<i>type</i>	String representing the request type (add/remove/switch).
-------------	---

5.7.3 Member Function Documentation

5.7.3.1 getType()

```
std::string Request::getType ( ) const [inline]
```

Gets the type of the request.

Returns

A string representing the type of the request.

Here is the caller graph for this function:

The documentation for this class was generated from the following file:

- [src/Request.h](#)

5.8 Schedule Class Reference

Class used to display a schedule.

```
#include <Schedule.h>
```

Public Member Functions

- [Schedule](#) (vector< [lesson](#) > lessons)
Parameterized constructor.
- void **display** ()
Displays the schedule.

5.8.1 Detailed Description

Class used to display a schedule.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Schedule()

```
Schedule::Schedule (  
    vector< lesson > lessons )
```

Parameterized constructor.

Parameters

<i>lessons</i>	Vector with all the lessons to be displayed.
----------------	--

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- src/[Schedule.h](#)
- src/Schedule.cpp

5.9 Student Class Reference

Class used to represent a student.

```
#include <student.h>
```


Public Member Functions

- **Student** ()=default
Default constructor.
- **Student** (string studentId, string name, list< [studentGroup](#) > group)
Parameterized constructor.
- std::string **getStudentID** () const
Gets the student ID.
- list< [studentGroup](#) > **getStudentGroups** () const
Gets all the classes the student belongs to.
- std::string **getName** () const
Gets the name of the student.
- void **setName** (const std::string &newName)
Sets the newName of the student.
- void **setStudentID** (const std::string &studentId)
Sets the student ID.
- void **addStudentGroup** (const [studentGroup](#) &GroupToAdd)
Adds a new class to the student.
- void **removeGroup** (const [studentGroup](#) &GroupToRemove)
Removes a class from the student.
- bool **isInUC** (const string &uc) const
Detects if the student is enrolled in a certain course.
- bool **isInClass** (const string &ucCode, const string &[studentGroup](#)) const
Detects if the student is enrolled in a certain class from a course.

5.9.1 Detailed Description

Class used to represent a student.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Student()

```
Student::Student (
    string studentId,
    string name,
    list< studentGroup > group )
```

Parameterized constructor.

Parameters

<i>studentId</i>	String representing the student ID.
<i>name</i>	String representing the name of the student.
<i>group</i>	A list with the classes the student has.

5.9.3 Member Function Documentation

5.9.3.1 addStudentGroup()

```
void Student::addStudentGroup (
    const studentGroup & GroupToAdd )
```

Adds a new class to the student.

Parameters

<i>GroupToAdd</i>	
-------------------	--

Here is the caller graph for this function:

5.9.3.2 getName()

```
string Student::getName ( ) const
```

Gets the name of the student.

Returns

A string representing the name of the student.

Here is the caller graph for this function:

5.9.3.3 getStudentGroups()

```
list< studentGroup > Student::getStudentGroups ( ) const
```

Gets all the classes the student belongs to.

Returns

A list of classes that the student belongs to.

Here is the caller graph for this function:

5.9.3.4 getStudentID()

```
string Student::getStudentID ( ) const
```

Gets the student ID.

Returns

A string representing the student ID.

Here is the caller graph for this function:

5.9.3.5 isInClass()

```
bool Student::isInClass (
    const string & ucCode,
    const string & studentGroup ) const
```

Detects if the student is enrolled in a certain class from a course.

Complexity is $O(n)$ where n is the amount of class the student has.

Parameters

<i>ucCode</i>	String representing a course.
<i>studentGroup</i>	String representing a class.

Returns

Returns true if the student is enrolled in a certain class form a course.

5.9.3.6 isInUC()

```
bool Student::isInUC (
    const string & uc ) const
```

Detects if the student is enrolled in a certain course.

Parameters

<i>uc</i>	String representing a course.
-----------	-------------------------------

Returns

Returns true if the student is enrolled in a certain course.

5.9.3.7 removeGroup()

```
void Student::removeGroup (
    const studentGroup & GroupToRemove )
```

Removes a class from the student.

Parameters

<i>GroupToRemove</i>	
----------------------	--

Here is the call graph for this function: Here is the caller graph for this function:

5.9.3.8 setName()

```
void Student::setName (
    const std::string & newName )
```

Sets the newName of the student.

Parameters

<i>newName</i>	A string representing the newName of the student
----------------	--

5.9.3.9 setStudentID()

```
void Student::setStudentID (
    const std::string & studentId )
```

Sets the student ID.

Parameters

<i>studentId</i>	A string representing the new student ID.
------------------	---

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- [src/student.h](#)
- [src/student.cpp](#)

5.10 studentGroup Class Reference

Class used to represent a class (group of students).

```
#include <studentGroup.h>
```

Public Member Functions

- **studentGroup** ()=default
Default constructor.
- **studentGroup** (const std::string &ucCode, const std::string &classCode)
Parameterized constructor.
- const std::string & **getClassCode** () const
Gets the class code.
- const std::string & **getUcCode** () const
Gets the course code.

5.10.1 Detailed Description

Class used to represent a class (group of students).

5.10.2 Constructor & Destructor Documentation

5.10.2.1 studentGroup()

```
studentGroup::studentGroup (
    const std::string & ucCode,
    const std::string & classCode )
```

Parameterized constructor.

Parameters

<i>ucCode</i>	String representing the course code.
<i>classCode</i>	String representing the class code.

5.10.3 Member Function Documentation

5.10.3.1 getClassCode()

```
const std::string & studentGroup::getClassCode ( ) const [inline]
```

Gets the class code.

Returns

A string representing the class code.

Here is the caller graph for this function:

5.10.3.2 getUcCode()

```
const std::string & studentGroup::getUcCode ( ) const [inline]
```

Gets the course code.

Returns

A string representing the course code.

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- [src/studentGroup.h](#)
- [src/studentGroup.cpp](#)

5.11 SwitchRequest Class Reference

[Request](#) of type switch.

```
#include <SwitchRequest.h>
```

Inheritance diagram for SwitchRequest:

Collaboration diagram for SwitchRequest:

Public Member Functions

- [SwitchRequest](#) (const std::string &studentID, const std::string &ucCode1, const std::string &ucCode2, const std::string &classCode1, const std::string &classCode2)
Parameterized constructor.
- std::string [getStudentID](#) () const
Gets the student ID.
- std::string [getUCCode1](#) () const
Gets the current course code.
- std::string [getUCCode2](#) () const
Gets the new course code.
- std::string [getClassCode1](#) () const
Gets the current class code.
- std::string [getClassCode2](#) () const
Gets the new class code.

Additional Inherited Members

5.11.1 Detailed Description

[Request](#) of type switch.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 SwitchRequest()

```
SwitchRequest::SwitchRequest (
    const std::string & studentID,
    const std::string & ucCode1,
    const std::string & ucCode2,
    const std::string & classCode1,
    const std::string & classCode2 )
```

Parameterized constructor.

Parameters

<i>studentID</i>	String representing the student ID.
<i>ucCode1</i>	String representing the current course code.
<i>ucCode2</i>	String representing the new course code.
<i>classCode1</i>	String representing the current class code.
<i>classCode2</i>	String representing the new class code.

5.11.3 Member Function Documentation

5.11.3.1 getClassCode1()

```
std::string SwitchRequest::getClassCode1 ( ) const
```

Gets the current class code.

Returns

A string representing the current class code.

Here is the caller graph for this function:

5.11.3.2 getClassCode2()

```
std::string SwitchRequest::getClassCode2 ( ) const
```

Gets the new class code.

Returns

A string representing the new class code.

Here is the caller graph for this function:

5.11.3.3 getStudentID()

```
std::string SwitchRequest::getStudentID ( ) const
```

Gets the student ID.

Returns

A string representing the student ID.

Here is the caller graph for this function:

5.11.3.4 getUCCode1()

```
std::string SwitchRequest::getUCCode1 ( ) const
```

Gets the current course code.

Returns

A string representing the current course code.

Here is the caller graph for this function:

5.11.3.5 getUCCode2()

```
std::string SwitchRequest::getUCCode2 ( ) const
```

Gets the new course code.

Returns

A string representing the new course code.

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- [src/SwitchRequest.h](#)
- [src/SwitchRequest.cpp](#)

Chapter 6

File Documentation

6.1 src/AddRequest.h File Reference

```
#include <string>
#include "Request.h"
Include dependency graph for AddRequest.h:
```

6.2 AddRequest.h

[Go to the documentation of this file.](#)

```
1
2 #ifndef PROJAED_ADDREQUEST_H
3 #define PROJAED_ADDREQUEST_H
4
5 #include <string>
6 #include "Request.h"
7
11 class AddRequest : public Request {
12 private:
13     std::string studentID;
14     std::string ucCode;
15     std::string classCode;
16
17
18 public:
25     AddRequest(const std::string &studentID, const std::string &ucCode, const std::string &classCode);
26
31     std::string getStudentID() const;
32
37     std::string getUCCode() const;
38
43     std::string getClassCode() const;
44
49     void setStudentID(const std::string &studentID);
50
55     void setUCCode(const std::string &ucCode);
56
61     void setClassCode(const std::string &classCode);
62 };
63
64 #endif //PROJAED_ADDREQUEST_H
```

6.3 src/ControlUnit.h File Reference

```
#include <vector>
#include <string>
#include "studentGroup.h"
#include <map>
#include "student.h"
#include <set>
#include <list>
#include <queue>
#include <stack>
#include <functional>
#include "lesson.h"
#include "Request.h"
#include "AddRequest.h"
#include "RemoveRequest.h"
#include "SwitchRequest.h"
```

Include dependency graph for ControlUnit.h: This graph shows which files directly or indirectly include this file:

Classes

- class [ControlUnit](#)

Class used to handle the core functions of the program.

6.4 ControlUnit.h

[Go to the documentation of this file.](#)

```
1
2 #ifndef PROJAED_CONTROLUNIT_H
3 #define PROJAED_CONTROLUNIT_H
4
5 #include <vector>
6 #include <string>
7 #include "studentGroup.h"
8 #include <map>
9 #include "student.h"
10 #include <set>
11 #include <list>
12 #include <queue>
13 #include <stack>
14 #include <functional>
15 #include "lesson.h"
16 #include "Request.h"
17 #include "AddRequest.h"
18 #include "RemoveRequest.h"
19 #include "SwitchRequest.h"
20
21 class ControlUnit {
22 private:
23     struct MainKey {
24         string ucCode;
25         string ClassCode;
26
27         bool operator<(const MainKey &other) const {
28             if (ucCode != other.ucCode) {
29                 return ucCode < other.ucCode;
30             }
31             return ClassCode < other.ClassCode;
32         }
33     };
34
35     string filename;
36     set<Student> StudentSet;
37     vector<lesson> LessonVector;
38     list<studentGroup> StudentGroupList;
```

```

43     map<MainKey, studentGroup *> KeyToStudentGroup;
44     map<MainKey, set<lesson *> LessonMap;
45     map<MainKey, int> SizeMap;
46     queue<Request *> RequestsToProcess;
47     stack<Request *> ProcessedRequests;
48     int cap = 30;
49
50 public :
51
52     void LoadCSV(string studentFilename);
53
54     void LoadClassesCSV();
55
56     void LoadClassesPerUcCSV();
57
58     void LoadStudentsClassesCSV();
59
60     vector<vector<lesson>> formatConflicts(vector<lesson> &lessons);
61
62     int maxSgSize();
63
64     void setCap(int n);
65
66     void saveChanges();
67
68     void DisplayStudentSchedule();
69
70     void DisplayClassSchedule();
71
72     int StudentsInAtLeastNUcs(int n);
73
74     int StudentsInAtMostNUcs(int n);
75
76     int StudentsInExactNUcs(int n);
77
78     void courseStudents(string courseCode, function<bool(Student, Student)> func);
79
80     void yearStudents(char year, function<bool(Student, Student)> func);
81
82     void classStudents(string classCode, function<bool(Student, Student)> func);
83
84     void UCWithMostStudents();
85
86     int NumBalanced(vector<studentGroup>, map<MainKey, int>);
87
88     bool IsThereConflict(vector<lesson>);
89
90     bool processRequest(Request *request, bool bypassStack = false);
91
92     void processAddRequest(AddRequest *addRequest);
93
94     void processRemoveRequest(RemoveRequest *removeRequest);
95
96     void processSwitchRequest(SwitchRequest *switchRequest);
97
98     void processAllRequests();
99
100    void removeLastPendingRequest();
101
102    void undoRequest(int n); //this method removes last n applied request
103
104    void createAdd();
105
106    void createRemove();
107
108    void createSwitch();
109
110    bool CheckAdd(AddRequest *addrq);
111
112    bool CheckRemove(RemoveRequest *remrq);
113
114    bool CheckSwitch(SwitchRequest *swrq);
115
116    string getClassInUc(string studentID, string ucCode);
117
118    void clearMemory();
119 };
120
121 #endif //PROJAED_CONTROLUNIT_H

```

6.5 src/lesson.h File Reference

```
#include <string>
#include <ctime>
#include "lesstime.h"
#include <iostream>
#include <map>
```

Include dependency graph for lesson.h: This graph shows which files directly or indirectly include this file:

Classes

- class [lesson](#)

Class used to represent a lesson from a course.

6.6 lesson.h

[Go to the documentation of this file.](#)

```
1
2 #ifndef PROJAED_LESSON_H
3 #define PROJAED_LESSON_H
4
5
6 #include <string>
7 #include <ctime>
8 #include "lesstime.h"
9 #include <iostream>
10 #include <map>
11
12
13
14
15 class lesson {
16 public:
17     lesson(const std::string &ucCode, const std::string &studentGroup, const std::string &weekday, double
18         startTime,
19         double duration, const std::string &type);
20
21     const std::string &getWeekday() const;
22
23     const lesstime &getStartTime() const;
24
25     const lesstime &getDuration() const;
26
27     const lesstime &getEndTime() const;
28
29     const string &getUcode() const;
30
31     const std::string &getType() const;
32
33     friend std::ostream &operator<<(std::ostream &os, const lesson &lesson);
34
35     mutable std::map<std::string, int> dayMap = {{"Monday", 0},
36         {"Tuesday", 1},
37         {"Wednesday", 2},
38         {"Thursday", 3},
39         {"Friday", 4},
40         {"Saturday", 5},
41         {"Sunday", 6}};
42
43     bool operator<(const lesson &other) const {
44         if (dayMap[this->getWeekday()] < dayMap[other.getWeekday()]) {
45             return true;
46         } else if (dayMap[this->getWeekday()] == dayMap[other.getWeekday()]) {
47             if (this->getStartTime() < other.getStartTime()) {
48                 return true;
49             } else if (other.getStartTime() < this->getStartTime()) {
50                 return false;
51             } else if (this->getUcode() < other.getUcode()) {
52                 return true;
53             } else {
54                 return false;
55             }
56         }
57     }
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
```

```

90     }
91     return false;
92
93 }
94
95 private:
96
97     std::string studentGroup;
98     std::string UcCode;
99     std::string weekday;
100     lesstime startTime;
101     lesstime duration;
102     lesstime endTime;
103     std::string type;
104 };
105
106
107 #endif //PROJAED_LESSON_H

```

6.7 src/lesstime.h File Reference

```

#include <string>
#include <iostream>
#include <iomanip>

```

Include dependency graph for lesstime.h: This graph shows which files directly or indirectly include this file:

Classes

- class [lesstime](#)

Class used to represent time.

6.8 lesstime.h

[Go to the documentation of this file.](#)

```

1
2 #ifndef PROJAED_LESSTIME_H
3 #define PROJAED_LESSTIME_H
4
5
6 #include <string>
7 #include <iostream>
8 #include <iomanip>
9 #include <string>
10
11 using namespace std;
12
13 class lesstime {
14 public:
15     explicit lesstime(double time);
16
17     lesstime();
18
19     lesstime(int hour, int minutes);
20
21     string displayHourFormat() const;
22
23     int getHour() const;
24
25     int getMinute() const;
26
27     friend std::ostream &operator<<(std::ostream &os, const lesstime &t);
28
29     bool operator<(const lesstime &other) const {
30         // Compare two lesstime objects based on their hours and minutes
31         if (hour < other.hour) {
32             return true;
33         } else if (hour == other.hour && minute < other.minute) {
34             return true;
35         }
36     }
37 }

```

```

63
64     return false;
65 }
66
67 bool operator==(const lessontime &other) const {
68     // Compare two lessontime objects based on their hours and minutes
69     if (hour == other.getHour() && minute == other.getMinute()) {
70         return true;
71     }
72
73     return false;
74 }
75
76 bool operator<=(const lessontime &other) const {
77     // Compare two lessontime objects based on their hours and minutes
78     return (hour < other.hour) || (hour == other.hour && minute <= other.minute);
79 }
80
81 private:
82     int hour;
83     int minute;
84 };
85
86
87 #endif //PROJAED_LESSONTIME_H

```

6.9 src/Menu.h File Reference

#include "ControlUnit.h"
 Include dependency graph for Menu.h:

Classes

- class [Menu](#)
Class used to represent the menu the user uses to navigate.

6.10 Menu.h

[Go to the documentation of this file.](#)

```

1
2 #ifndef PROJAED_MENU_H
3 #define PROJAED_MENU_H
4
5 #include "ControlUnit.h"
6
10 class Menu {
11 public:
15     void createMenu();
20     void SeeStudentSchedule();
21
25     void SeeClassSchedule();
26
30     void SeeNumStudentsInExactNUCs();
31
35     void SeeNumStudentsAtLeastNUCs();
36
40     void SeeNumStudentsAtMostNUCs();
41
45     void SeeUcFromMostStudents();
46
50     void SeeNumStudentsInNUCs();
51
55     void listingMenu();
56
60     void requestMenu();
61
65     void scheduleMenu();
66
70     void studentMenu();

```

```

71
72     void SeeStudentsInUc(function<bool(Student, Student)> comp);
73
74     void SeeStudentsInYear(function<bool(Student, Student)> comp);
75
76     void SeeStudentsInClass(function<bool(Student, Student)> comp);
77
78     void createRequest();
79
80     function<bool(Student, Student)> optionStudentMenu();
81
82 private:
83     ControlUnit Control;
84
85 };
86
87 #endif //PROJAED_MENU_H

```

6.11 src/RemoveRequest.h File Reference

```

#include <string>
#include "Request.h"

```

Include dependency graph for RemoveRequest.h: This graph shows which files directly or indirectly include this file:

Classes

- class [RemoveRequest](#)
Request of type remove.

6.12 RemoveRequest.h

[Go to the documentation of this file.](#)

```

1
2 #ifndef PROJAED_REMOVEREQUEST_H
3 #define PROJAED_REMOVEREQUEST_H
4
5 #include <string>
6 #include "Request.h"
7
8 class RemoveRequest : public Request {
9 private:
10     std::string studentID;
11     std::string ucCode;
12     std::string classCode;
13
14 public:
15     RemoveRequest(const std::string &studentID, const std::string &ucCode,
16                 const std::string &classCode);
17
18     std::string getStudentID() const;
19
20     std::string getUCCode() const;
21
22     std::string getClassCode() const;
23
24     void setStudentID(const std::string &studentID);
25
26     void setUCCode(const std::string &ucCode);
27
28     void setClassCode(const std::string &classCode);
29 };
30
31 #endif //PROJAED_REMOVEREQUEST_H

```

6.13 src/Request.h File Reference

```
#include "student.h"
#include "studentGroup.h"
```

Include dependency graph for Request.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Request](#)

Class used to represent a generic request.

6.14 Request.h

[Go to the documentation of this file.](#)

```
1
2 #ifndef PROJAED_REQUEST_H
3 #define PROJAED_REQUEST_H
4
5 #include "student.h"
6 #include "studentGroup.h"
7
11 class Request {
12 private:
13     static int count; // Static variable for request counter.
14     int requestId; // ID for each request.
15
16     std::string type;
17
18 public:
22     void static resetCount() {
23         count = 0;
24     }
25
30     Request(std::string type) {
31         count++;
32         requestId = count;
33         cout << "request id is " << requestId << " and count is " << count << endl;
34
35         this->type = type;
36     }
37
42     std::string getType()const { return type; }
43
44     // Virtual function for allowing downcasting.
45     virtual void dummy() {}
46
47     virtual ~Request() {};
48 };
49
50 #endif // PROJAED_REQUEST_H
```

6.15 src/Schedule.h File Reference

```
#include <vector>
#include "lesson.h"
#include <map>
```

Include dependency graph for Schedule.h:

Classes

- class [Schedule](#)

Class used to display a schedule.

6.16 Schedule.h

[Go to the documentation of this file.](#)

```

1
2 #ifndef PROJAED_SCHEDULE_H
3 #define PROJAED_SCHEDULE_H
4
5 #include <vector>
6 #include "lesson.h"
7 #include <map>
8
9 using namespace std;
10
11 class Schedule {
12 private:
13     vector<lesson> lessons; //the lessons that go into the schedule
14     map<pair<int, int>, string> ScheduleMap; // a schedule is made up of 30 by 6 blocks
15 public:
16     Schedule(vector<lesson> lessons);
17
18     void display();
19 };
20
21 #endif //PROJAED_SCHEDULE_H

```

6.17 src/student.h File Reference

```

#include <set>
#include <tuple>
#include <string>
#include <list>
#include "studentGroup.h"

```

Include dependency graph for student.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Student](#)
Class used to represent a student.

6.18 student.h

[Go to the documentation of this file.](#)

```

1
2 #ifndef PROJAED_STUDENT_H
3 #define PROJAED_STUDENT_H
4
5 #include <set>
6 #include <tuple>
7 #include <string>
8 #include <list>
9 #include "studentGroup.h"
10
11 using namespace std;
12
13 class Student {
14 public:
15     Student() = default;
16
17     Student(string studentId, string name, list<studentGroup> group);
18
19     std::string getStudentID() const;
20
21     list<studentGroup> getStudentGroups() const;

```

```

42
43     std::string getName() const;
44
45     void setName(const std::string &newName);
46
47     void setStudentID(const std::string &studentID);
48
49     void addStudentGroup(const studentGroup &GroupToAdd);
50
51     void removeGroup(const studentGroup &GroupToRemove);
52
53     bool isInUC(const string &uc) const;
54
55     bool isInClass(const string &ucCode, const string &studentGroup) const;
56
57     bool operator<(const Student &other) const {
58         return studentID < other.studentID;
59     }
60
61     bool operator==(const Student &other) const {
62         return (this->studentID == other.studentID) && (this->name == other.name);
63     }
64
65     friend std::ostream &operator<<(std::ostream &os, const Student &student);
66
67 private:
68     std::string studentID;
69     std::string name;
70     std::list<studentGroup> StudentGroups;
71 };
72
73 #endif //PROJAED_STUDENT_H

```

6.19 src/studentGroup.h File Reference

```

#include <iostream>
#include <string>

```

Include dependency graph for studentGroup.h: This graph shows which files directly or indirectly include this file:

Classes

- class [studentGroup](#)

Class used to represent a class (group of students).

6.20 studentGroup.h

[Go to the documentation of this file.](#)

```

1
2 #ifndef STUDENTGROUP_H
3 #define STUDENTGROUP_H
4
5 #include <iostream>
6 #include <string>
7
11 class studentGroup {
12 public:
13     studentGroup() = default;
14
15     studentGroup(const std::string &ucCode, const std::string &classCode);
16
17     const std::string &getClassCode() const {
18         return classCode;
19     }
20
21     const std::string &getUcCode() const {
22         return UcCode;
23     }
24
25

```

```

41     bool operator<(const studentGroup &other) const {
42         // Define a comparison logic here based on your criteria.
43         // For example, you can compare based on class code or other fields.
44         return this->classCode + this->UcCode < other.classCode + other.UcCode;
45     }
46
47     friend std::ostream &operator<<(std::ostream &os, const studentGroup &group) {
48         os << "UcCode: " << group.UcCode << ", Class Code: " << group.classCode;
49         return os;
50     }
51
52
53 private:
54     std::string classCode;
55     std::string UcCode;
56
57
58
59 };
60
61 #endif

```

6.21 src/SwitchRequest.h File Reference

```
#include "Request.h"
```

```
#include <string>
```

Include dependency graph for SwitchRequest.h: This graph shows which files directly or indirectly include this file:

Classes

- class [SwitchRequest](#)
Request of type switch.

6.22 SwitchRequest.h

[Go to the documentation of this file.](#)

```

1
2 #ifndef PROJAED_SWITCHREQUEST_H
3 #define PROJAED_SWITCHREQUEST_H
4
5 #include "Request.h"
6 #include <string>
7
11 class SwitchRequest : public Request {
12 private:
13     std::string studentID;
14     std::string ucCode1;
15     std::string ucCode2;
16     std::string classCode1;
17     std::string classCode2;
18
19 public:
20     SwitchRequest(const std::string &studentID, const std::string &ucCode1, const std::string &ucCode2,
21                 const std::string &classCode1, const std::string &classCode2);
22
23     std::string getStudentID() const;
24
25     std::string getUCCode1() const;
26
27     std::string getUCCode2() const;
28
29     std::string getClassCode1() const;
30
31     std::string getClassCode2() const;
32 };
33
34 #endif // PROJAED_SWITCHREQUEST_H

```


Index

- AddRequest, 9
 - AddRequest, 10
 - getClassCode, 10
 - getStudentID, 10
 - getUCCode, 10
 - setClassCode, 11
 - setStudentID, 11
 - setUCCode, 11
- addStudentGroup
 - Student, 36
- CheckAdd
 - ControlUnit, 13
- CheckRemove
 - ControlUnit, 14
- CheckSwitch
 - ControlUnit, 14
- classStudents
 - ControlUnit, 14
- clearMemory
 - ControlUnit, 15
- ControlUnit, 12
 - CheckAdd, 13
 - CheckRemove, 14
 - CheckSwitch, 14
 - classStudents, 14
 - clearMemory, 15
 - courseStudents, 15
 - createAdd, 15
 - createRemove, 15
 - createSwitch, 16
 - DisplayClassSchedule, 16
 - DisplayStudentSchedule, 16
 - formatConflicts, 16
 - getClassInUc, 17
 - IsThereConflict, 17
 - LoadCSV, 17
 - maxSgSize, 18
 - NumBalanced, 18
 - processAddRequest, 18
 - processAllRequests, 19
 - processRemoveRequest, 19
 - processRequest, 19
 - processSwitchRequest, 19
 - removeLastPendingRequest, 20
 - saveChanges, 20
 - setCap, 20
 - StudentsInAtLeastNUcs, 20
 - StudentsInAtMostNUcs, 21
 - StudentsInExactNUcs, 21
 - UCWithMostStudents, 22
 - undoRequest, 22
 - yearStudents, 22
- courseStudents
 - ControlUnit, 15
- createAdd
 - ControlUnit, 15
- createRemove
 - ControlUnit, 15
- createSwitch
 - ControlUnit, 16
- DisplayClassSchedule
 - ControlUnit, 16
- displayHourFormat
 - lesstime, 27
- DisplayStudentSchedule
 - ControlUnit, 16
- formatConflicts
 - ControlUnit, 16
- getClassCode
 - AddRequest, 10
 - RemoveRequest, 31
 - studentGroup, 39
- getClassCode1
 - SwitchRequest, 41
- getClassCode2
 - SwitchRequest, 41
- getClassInUc
 - ControlUnit, 17
- getDuration
 - lesson, 24
- getEndTime
 - lesson, 24
- getHour
 - lesstime, 27
- getMinute
 - lesstime, 27
- getName
 - Student, 36
- getStartTime
 - lesson, 24
- getStudentGroups
 - Student, 36
- getStudentID
 - AddRequest, 10
 - RemoveRequest, 31
 - Student, 36

- SwitchRequest, 41
- getType
 - lesson, 24
 - Request, 33
- getUCCode
 - AddRequest, 10
 - RemoveRequest, 31
- getUcCode
 - studentGroup, 39
- getUccode
 - lesson, 25
- getUCCode1
 - SwitchRequest, 41
- getUCCode2
 - SwitchRequest, 42
- getWeekday
 - lesson, 25
- isInClass
 - Student, 36
- isInUC
 - Student, 37
- IsThereConflict
 - ControlUnit, 17
- lesson, 23
 - getDuration, 24
 - getEndTime, 24
 - getStartTime, 24
 - getType, 24
 - getUccode, 25
 - getWeekday, 25
 - lesson, 23
- lesstime, 25
 - displayHourFormat, 27
 - getHour, 27
 - getMinute, 27
 - lesstime, 26
- LoadCSV
 - ControlUnit, 17
- maxSgSize
 - ControlUnit, 18
- Menu, 28
 - optionStudentMenu, 29
 - SeeStudentsInClass, 29
 - SeeStudentsInUc, 29
 - SeeStudentsInYear, 29
- NumBalanced
 - ControlUnit, 18
- optionStudentMenu
 - Menu, 29
- processAddRequest
 - ControlUnit, 18
- processAllRequests
 - ControlUnit, 19
- processRemoveRequest
 - ControlUnit, 19
- processRequest
 - ControlUnit, 19
- processSwitchRequest
 - ControlUnit, 19
- removeGroup
 - Student, 37
- removeLastPendingRequest
 - ControlUnit, 20
- RemoveRequest, 30
 - getClassCode, 31
 - getStudentID, 31
 - getUCCode, 31
 - RemoveRequest, 30
 - setClassCode, 31
 - setStudentID, 32
 - setUCCode, 32
- Request, 32
 - getType, 33
 - Request, 33
- saveChanges
 - ControlUnit, 20
- Schedule, 34
 - Schedule, 34
- SeeStudentsInClass
 - Menu, 29
- SeeStudentsInUc
 - Menu, 29
- SeeStudentsInYear
 - Menu, 29
- setCap
 - ControlUnit, 20
- setClassCode
 - AddRequest, 11
 - RemoveRequest, 31
- setName
 - Student, 38
- setStudentID
 - AddRequest, 11
 - RemoveRequest, 32
 - Student, 38
- setUCCode
 - AddRequest, 11
 - RemoveRequest, 32
- src/AddRequest.h, 43
- src/ControlUnit.h, 44
- src/lesson.h, 46
- src/lesstime.h, 47
- src/Menu.h, 48
- src/RemoveRequest.h, 49
- src/Request.h, 50
- src/Schedule.h, 50, 51
- src/student.h, 51
- src/studentGroup.h, 52
- src/SwitchRequest.h, 53
- Student, 34
 - addStudentGroup, 36

- getName, [36](#)
- getStudentGroups, [36](#)
- getStudentID, [36](#)
- isInClass, [36](#)
- isInUC, [37](#)
- removeGroup, [37](#)
- setName, [38](#)
- setStudentID, [38](#)
- Student, [35](#)
- studentGroup, [38](#)
 - getClassCode, [39](#)
 - getUcCode, [39](#)
 - studentGroup, [39](#)
- StudentsInAtLeastNUcs
 - ControlUnit, [20](#)
- StudentsInAtMostNUcs
 - ControlUnit, [21](#)
- StudentsInExactNUcs
 - ControlUnit, [21](#)
- SwitchRequest, [40](#)
 - getClassCode1, [41](#)
 - getClassCode2, [41](#)
 - getStudentID, [41](#)
 - getUCCode1, [41](#)
 - getUCCode2, [42](#)
 - SwitchRequest, [40](#)
- UCWithMostStudents
 - ControlUnit, [22](#)
- undoRequest
 - ControlUnit, [22](#)
- yearStudents
 - ControlUnit, [22](#)