SEP1Y Final Project

Time Scheduling at VIA Made Easy

Daniel Lopes Adrião, 315274

Dragos Daniel Bonaparte, 315261

Laura do Bem Rebelo, 315174

Matas Armonaitis, 315263

Supervisors:

Steffen Andersen (SVA), Mona Andersen (MWA)

REPORT: 35523 characters

Software Technology Engineering

1st Semester

15.12.2021

Table of contents

Abstract

The aim of this project is to provide VIA's timetable manager with a tool to aid his job: scheduling classes for students and teachers at VIA.

In order to achieve this, a Java application with a graphical user interface was developed. This application is capable of retrieving and processing information about the current semester from comma-separated files, thus enabling the scheduling of sessions by the timetable manager according to real information. The sessions booked through the program are saved in files after every change so that no changes are lost. The product also includes a responsive website that can display any timetable that has been generated inside of the Java application.

The outcome of the project was positive: both the desired application and website were developed with satisfactory functionality and pleasing visuals. Although not all of the initially set requirements were met, the highest-priority ones were accomplished, meaning that the program fulfils its purpose without any major problems.

1 Introduction

VIA's timetable manager is currently unable to satisfactorily produce timetables for his institution. This is due to the lack of a tool that can automatically factor everything that needs to be taken into consideration when scheduling sessions for a large university such as VIA. These considerations are mainly the overlapping of timeframes and the vacancy of classrooms, because a student or teacher cannot attend two sessions at once, and two sessions cannot be held in the same classroom at the same time.

The method presently adopted by the scheduler is lengthy and prone to error: before the semester starts, the Head of Department provides him with a list of every course, student, class, and teacher. With that information, he makes use of spreadsheets to manually create a timetable for each class. After the manager has produced the first draft, a "test week" is conducted and the timetables are uploaded to a website where they can be accessed by students and teachers. Not only is this procedure extremely time-consuming, it is also unreasonable for one person to manually check for the availability of classrooms whenever they schedule a session.

Moreover, in higher educational institutions, classes are often cancelled for unpredictable reasons. This leaves a vacant classroom and a gap in students and teachers' schedules. The lack of a system that properly displays this information to whoever is scheduling the classes may lead to teachers not knowing that there is an available room and giving up a class due to the inaccessibility of this information.

A possible application for the manager to use would be Google Calendars, which is easy to sync with multiple devices. However, it lacks a feature that is imperative when booking classes for VIA. The University's new Campus has a shortage of classrooms due to the high number of students currently enrolled, and therefore it is difficult for a timetable manager to know which classrooms are available to book for a certain class. (What Can You Do with Calendar? - Google Workspace Learning Center, 2021)

In summary, the existing problem is the inefficiency of the time schedule manager in producing and providing well-organized timetables to students and teachers.

With this in mind, it is reasonable to infer the need for a tool that can aid the scheduler by automatically checking for timeframe and classroom overlaps, and also graphically display a timetable for easy access for students and teachers, making it so that there is no need to manually create them through spreadsheets.

This tool will:

- Enable the scheduler's work to be less time-consuming and intellectually demanding.
- Prevent VIA's timetables to have overlaps by mistake.
- Enable the reduced number of existing classrooms in the current campus to be used to their full potential.

Delimitation-wise, the tool referred to in this report does not support timetables for teachers, does not feature a login feature, and does not take students with credits into consideration. This means that the only timetables that will be generated are for whole classes and never for individual students.

In order for the tool to meet its purpose, a few mandatory functionalities must be implemented. These will be analyzed in the following chapters.

2 Analysis

Having fully understood the problem, it was possible to infer what the requirements for the desired tool were. They are as follows:

Requirements

CRITICAL PRIORITY

1. As a timetable manager, I want to schedule a session so that teachers and students know when to get together for a session.
2. As a timetable manager, I want students and teachers to be assigned to the courses that they learn or teach so that these appear on their timetables.
3. As a timetable manager, I want to book classrooms for sessions so that students and teachers have a place to get together.
4. As a timetable manager, I want to be able to cancel sessions so that there are more classrooms available for other sessions.
5. As a timetable manager, the access to the planning tool should be limited to me so that I am the only one able to modify schedules.
6. As a timetable manager, I want to see only classrooms which are available when scheduling a class so that there are no overlaps.
7. As a student, I want to see my timetable so that I know when and where I have sessions.

HIGH PRIORITY

8. As a timetable manager, I want to reschedule sessions so that unforeseen events (illness, personal problems, events) do not get in the way of the intended number of lessons for each course.
9. As a timetable manager, I want the changes made to the schedules to be immediately available for students and teachers to see, so that they are always up to date with their timetables.
10. As a timetable manager, I want to remove students from a class so that I can assign them to another class if they have requested a switch.
11. As a timetable manager, I want to add students to a class so that they can have access to a new timetable if they have requested to switch classes.
12. As a timetable manager, I want to be able to assign two teachers to the same course when the course requires it.
13. As a timetable manager, I want to remove students from a course so that it will not appear in their timetable.
14. I want to be able to edit enrollment in a course so that I can add individual students to specific courses.

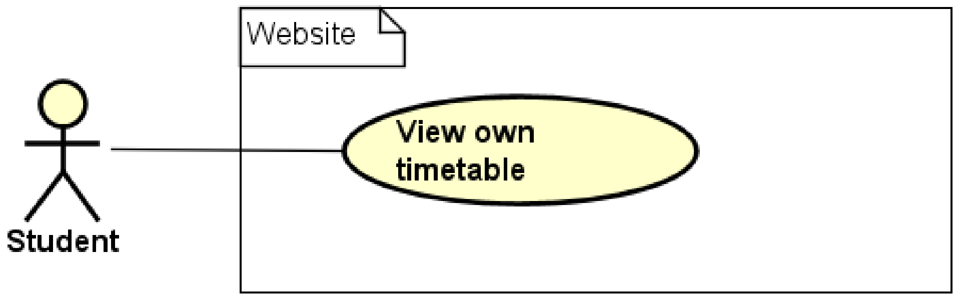
LOW PRIORITY

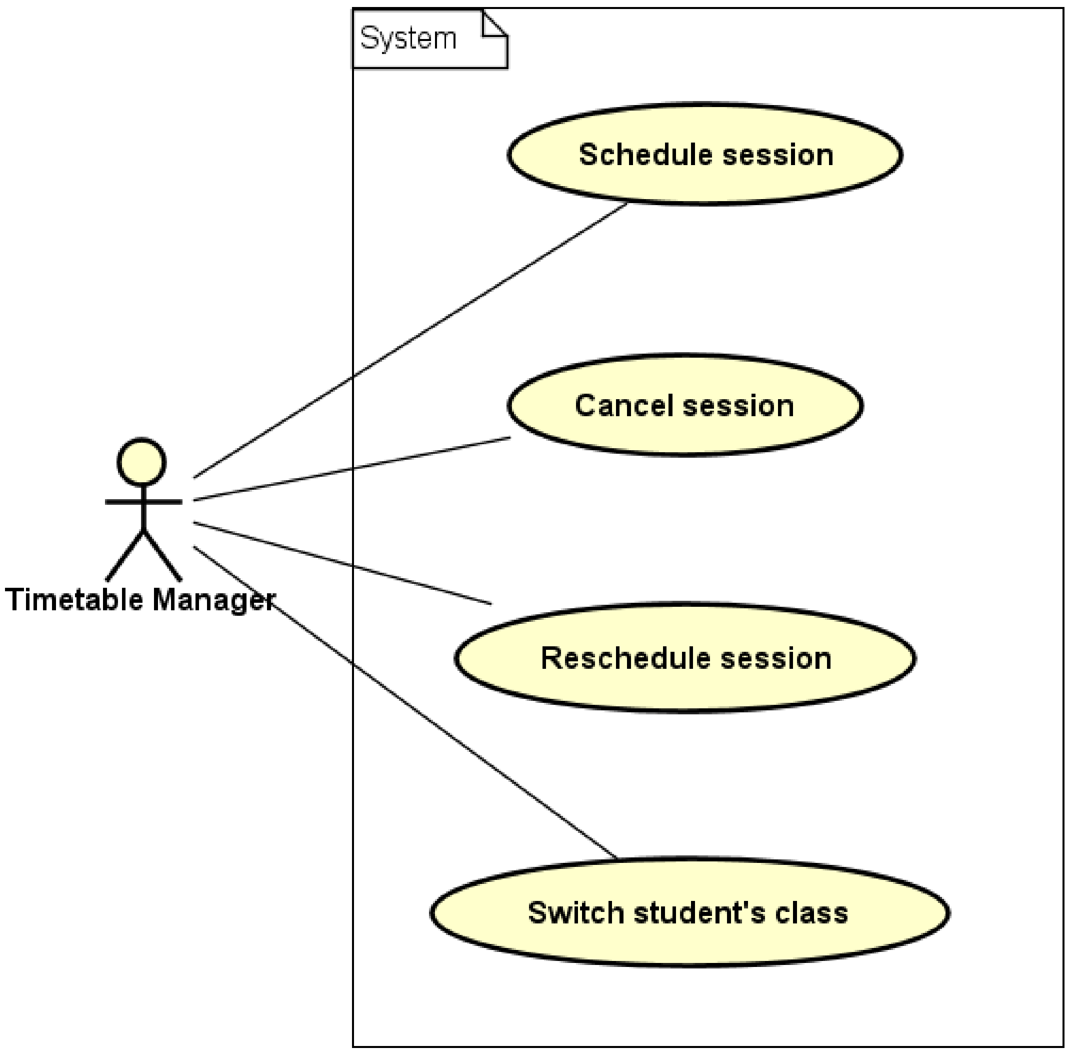
15. As a timetable manager, I want to be able to double the capacity of a room and merge two classes if it has a foldable wall.
16. As a user of this system, I want to see my schedule on a weekly format, so that I can see the sessions for each day.
17. As a user of this system, I want to be warned when sudden changes appear in the schedules.

NON-FUNCTIONAL REQUIREMENTS:

18. As a timetable manager, when assigning a student with credits to a course from another semester, I want the system to check if a timetable without overlaps is available for this student.

Given these requirements, the following use cases were formulated:





What follows is a table illustrating the relationship between the requirements and the use cases. The numbers in bold represent critical requirements.

Use cases	Covered requirements
Schedule session	**1**, **3**, **6**, 9
Reschedule session	8, 9
Cancel session	**4**, 9
Switch student's class	10, 11
View own timetable	**7**, 16, 17
No use case in particular	**2**, **5**

When it is stated that two requirements were met but not by any use case in particular, it is because:

- **Requirement number 2** states that students and teachers should be automatically assigned to the courses that they learn or teach. This is fulfilled not through a use case, but rather by the system itself as soon as it runs, so long as it is given the necessary information inside of a file.

- **Requirement number 5** states that the access to the planning tool should be limited to the timetable manager. This is achieved not through any of the use cases, but rather through the fact that the only person who can make changes to the timetables is the one in possession of the program, and that is intended to be the scheduler.

The use case that will serve as an example throughout this report will be the one that meets the most requirements: **Schedule session**. This use case is the most crucial because the other ones cannot be initiated without it happening first.

Here follows its use case description:

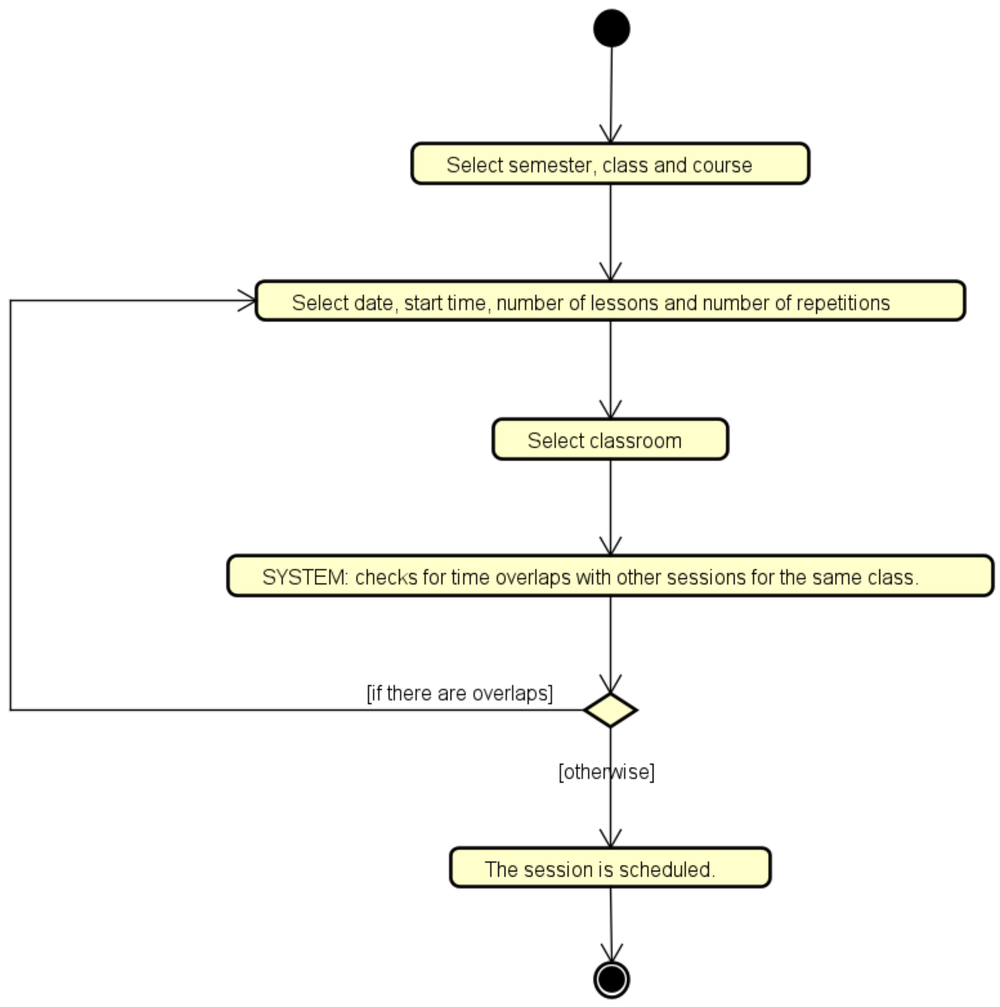
Use case	Schedule session
Summary	Schedule a session for a certain timeframe and book a classroom for that timeframe.
Actor	Timetable Manager
Precondition	The courses, students and classrooms must be registered in the system, given the information from the head of department.
Postcondition	The timetable is updated for the students affected by the change. The classroom associated with the session is marked as booked for that timeframe.
Base Sequence	1. Select semester, class, course. 2. Select date, start time, number of lessons and number of repetitions (how many copies of the session will be created for future weeks). 3. Select classroom. 4. SYSTEM: checks for time overlaps with other sessions for the same class. If there are time overlaps, go back to 2. 5. The session is scheduled.
Note	After changes have been made, the schedules are updated and whoever accesses the website will see the new changes. The requirements met by this Use Case are 1, 3, 6, 9

The reason why it the base sequence is not very lengthy and why there are very few checks for the validity of user input is because, for example, when prompted to select a semester, class or course, the only choices available are the ones retrieved from the Head of Department file, meaning that they will all be valid.

As soon as a session is scheduled, all of the changes are saved.

The rest of the use case descriptions can be found in APPENDIX 1B - ANALYSIS DOCUMENT.

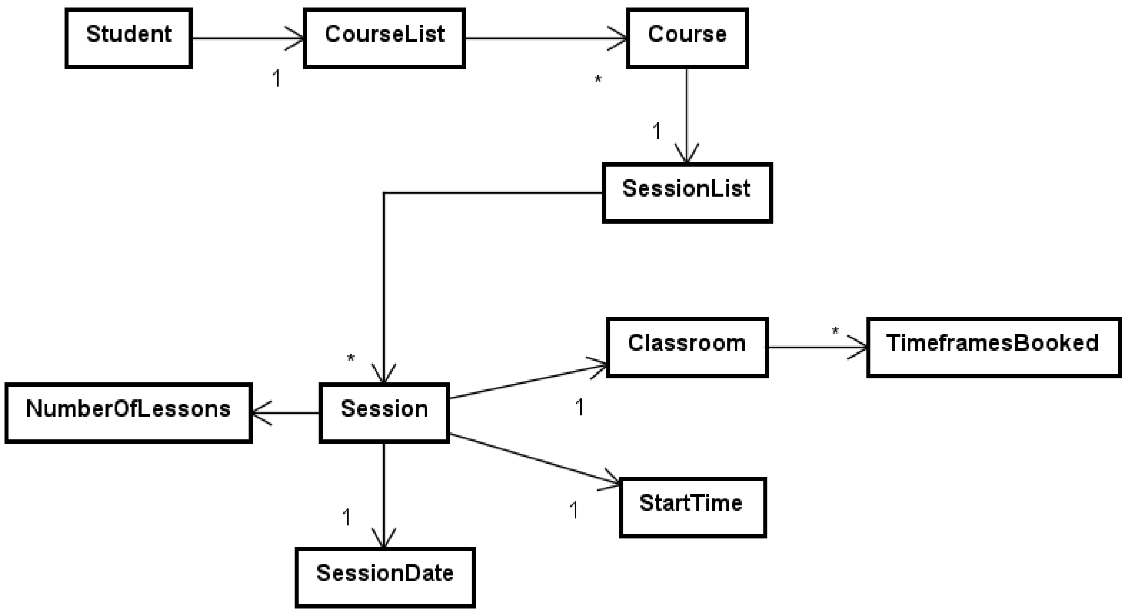
The activity diagram for this use case is as follows:


Select semester, class and course
Select date, start time, number of lessons and number of repetitions
Select classroom
SYSTEM: checks for time overlaps with other sessions for the same class.
[if there are overlaps]
[otherwise]
The session is scheduled.

The reason why a classroom can only be selected after a date, start time and number of lessons have been picked is because before prompting the user with classroom options, the system checks which classrooms are available for a certain timeframe. Therefore, the timeframe needs to be provided beforehand.

The rest of the activity diagrams can be found in APPENDIX 1B - ANALYSIS DOCUMENT.

Below can be found the domain model for this system:



When it is stated that a Student has a Course List, it means that they will be assigned to the Course List corresponding to a class. For example:

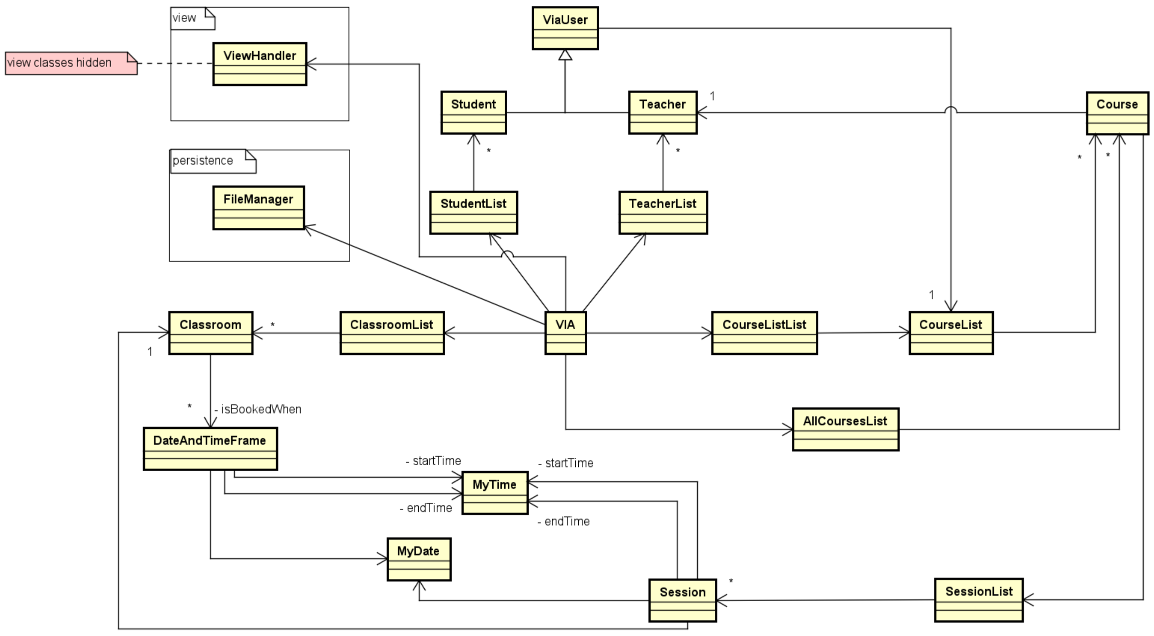
If Student 123456 is assigned to the Course List corresponding to Class **1X**, he will have the courses SDJ**1X**, DMA**1X**, RWD**1X** and SEP**1X** in his Course List.

Each Course, in turn, has a Session List. This way, whenever a timetable for a student is fetched, it will look through the courses in the student's Course List, fetch each Course's sessions and arrange them in a timetable.

The Session, in turn, is the center piece of the Domain model. It contains a Session Date, Start Time and Number of Lessons – these variables define a timeframe for the Session, which enables a check for overlaps inside of the system. Then, each session has a classroom, which has a list of Timeframes Booked. The existence of this list is also crucial: it will make it possible for the availability of a classroom in a certain timeframe to be checked and displayed whenever appropriate.

3 Design

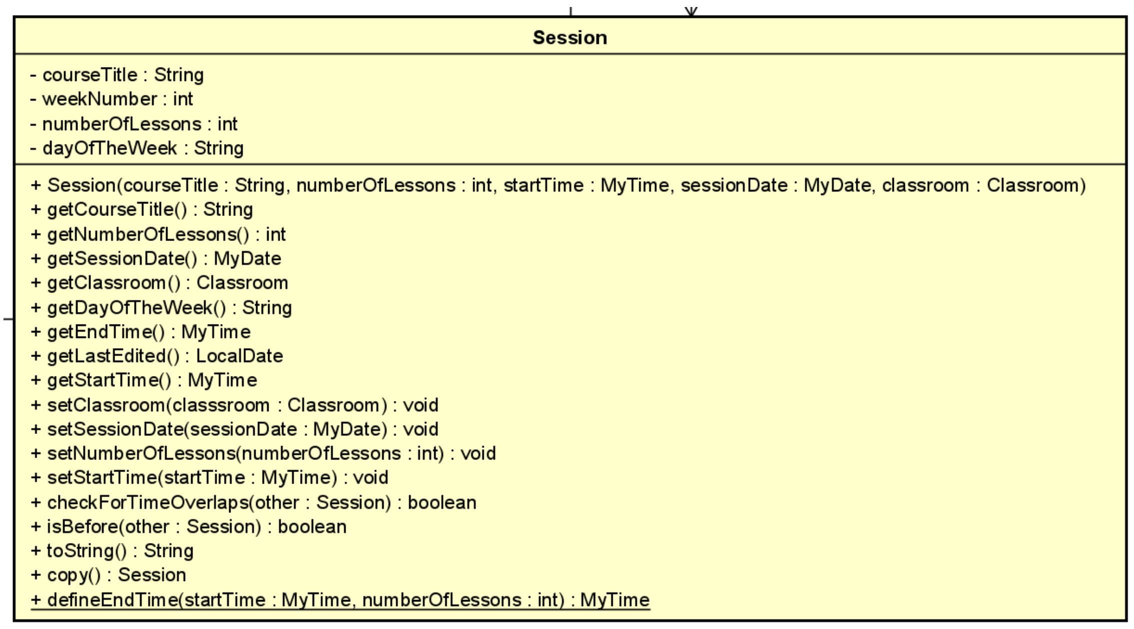
After analyzing the problem and its desired use cases, it was possible for a Class Diagram to be designed.



The central piece of the Class Diagram, VIA, which can also be referred to as Model Manager, has an instance variable of a variety of container classes. Inside it is stored all of the information for the current semester: all Classrooms, Students, Teachers, Courses and Course Lists (grouped by Classes (1X,1Y,2X…). The existence of a Model Manager enables the possibility to implement methods that retrieve information from anywhere inside of the program.

A full class diagram can be found in APPENDIX 2B – CLASS DIAGRAM.

A class that is worth diving into detail regarding the Schedule Session use case is the `Session` class. Its class diagram is in the following page.



```
Session
- courseTitle : String
- weekNumber : int
- numberOfLessons : int
- dayOfTheWeek : String
+ Session(courseTitle : String, numberOfLessons : int, startTime : MyTime, sessionDate : MyDate, classroom : Classroom)
+ getCourseTitle() : String
+ getNumberOfLessons() : int
+ getSessionDate() : MyDate
+ getClassroom() : Classroom
+ getDayOfTheWeek() : String
+ getEndTime() : MyTime
+ getLastEdited() : LocalDate
+ getStartTime() : MyTime
+ setClassroom(classsroom : Classroom) : void
+ setSessionDate(sessionDate : MyDate) : void
+ setNumberOfLessons(numberOfLessons : int) : void
+ setStartTime(startTime : MyTime) : void
+ checkForTimeOverlaps(other : Session) : boolean
+ isBefore(other : Session) : boolean
+ toString() : String
+ copy() : Session
+ defineEndTime(startTime : MyTime, numberOfLessons : int) : MyTime
```

Although it is not shown in this picture, `Session` also has:

- Two instance variables of type `MyTime` (startTime and endTime);
- One instance variable of type `Classroom` (classroom);
- Two instance variables of type `MyDate` (sessionDate and lastEdited);

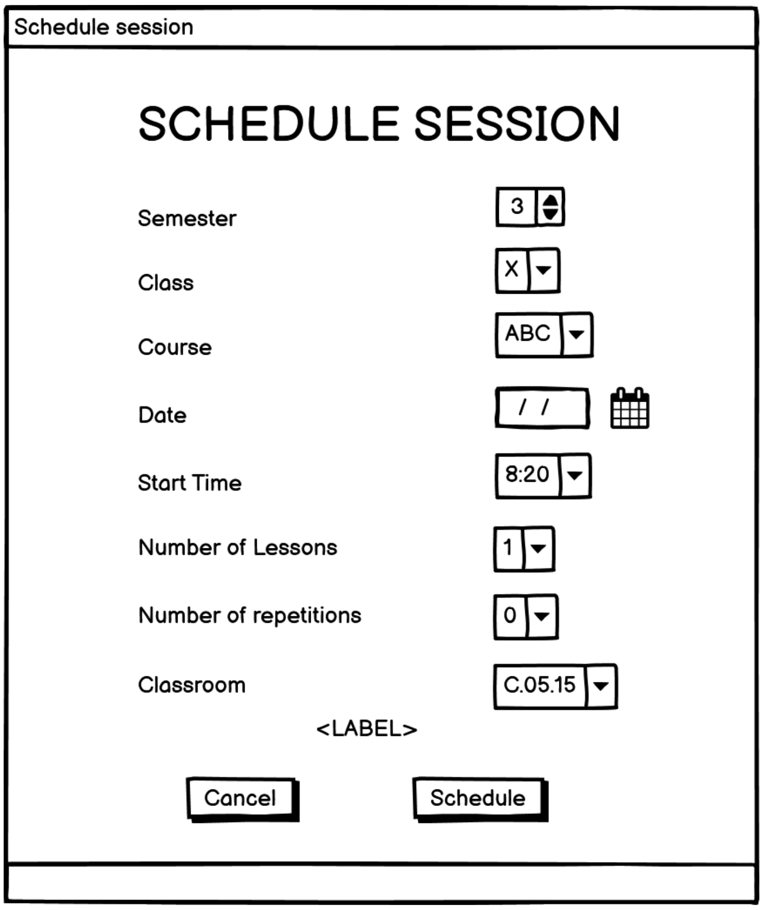
This is a class with many instance variables due to the fact that it requires storing a lot of different information. There is no method inside of it that directly regards scheduling, rescheduling or cancelling, but this is because those methods are, instead, implemented inside of the model manager class, VIA. The reason why is that VIA has access to every class in the diagram, meaning it can add a `Session` to a `SessionList`, for example, while a `Session` cannot schedule itself.

One method worth noting in the `Session` class, though, is "`checkForTimeOverlaps`". This method is relevant to the use case "Schedule Session" in a sense that two sessions cannot be scheduled for the same class within the same time period. So, what the method does is compare the start and end time of both sessions and see if they overlap. If they overlap, it will return true, and so the system will know not to allow the scheduling of these two sessions for the same class.

It could also be relevant to talk about "`defineEndTime`". As is observable, the constructor for Session does not take "endTime" as an argument, but rather an int numberOfLessons and a `MyTime` variable called startTime. What the constructor does in order to initialize the "endTime" variable is call `defineEndTime`, which will use both "startTime" and "numberOfLessons" to determine the appropriate end time according to the default start and end times at VIA.

Moreover, another instance variable worth noticing is “`weekNumber`”. This variable also does not appear in the constructor as an argument, but is fetched through a static method inside of the `MyDate` class, called “`getWeekNumber`” for a certain date. The need for this instance variable arose from the necessity to display timetables in a weekly manner inside of the website. With the sessions having a “week number” variable, it would then be possible to fetch this information using JavaScript and then navigate from week to week inside of the website without overlaps or conflicts.

The GUI design for the window related to the use case at hand is as follows:

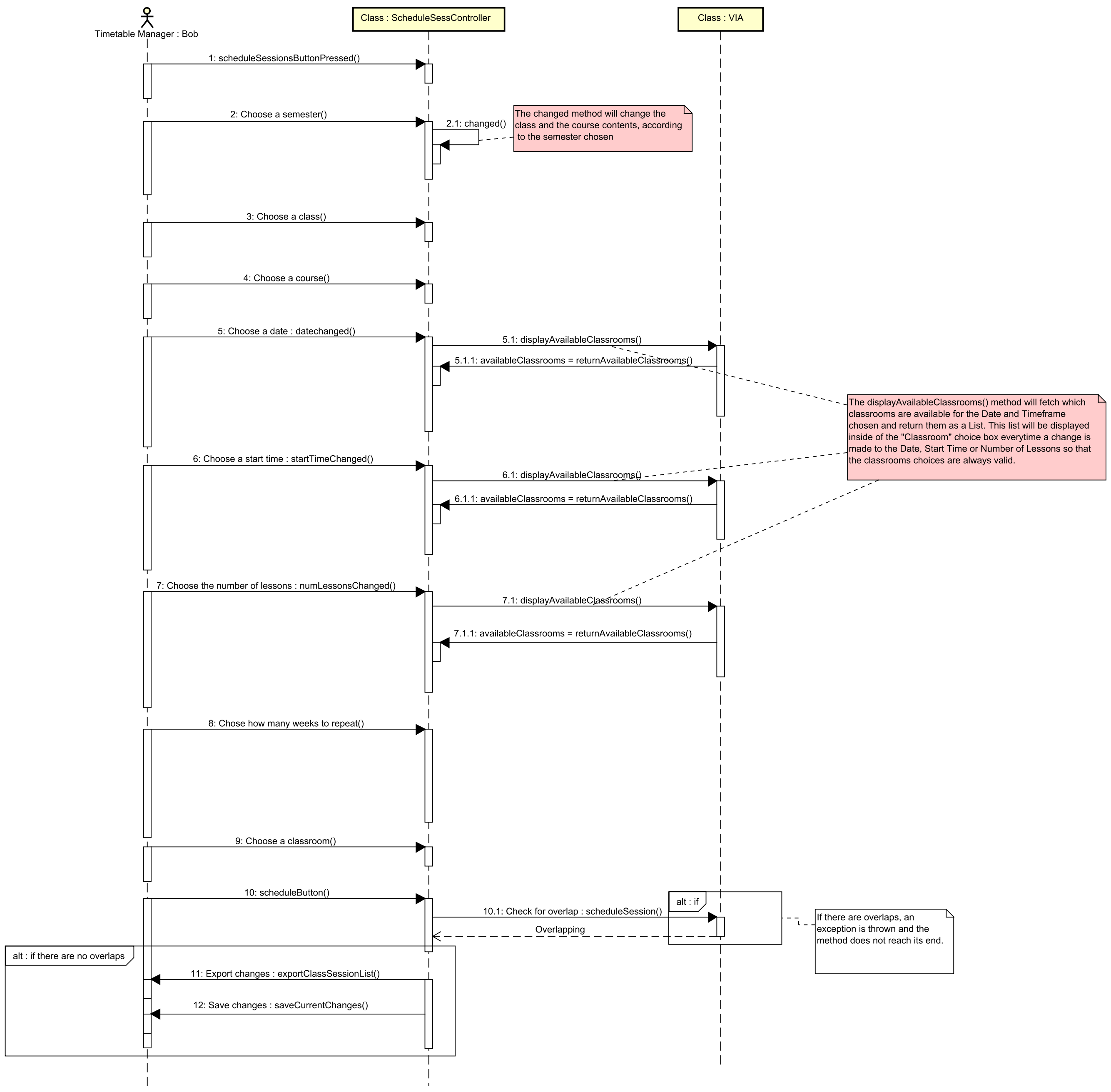


In summary, it starts by requesting the information for a class (semester and class), and afterwards, all of the arguments requested by a `Session` object constructor. This is because, in practice, when scheduling a session, a `Session` object will be created inside of a `SessionList` corresponding to a `Course` of a certain class (e.g., `SessionList` of SEP1Y).

The label at the bottom of the window is used to display errors.

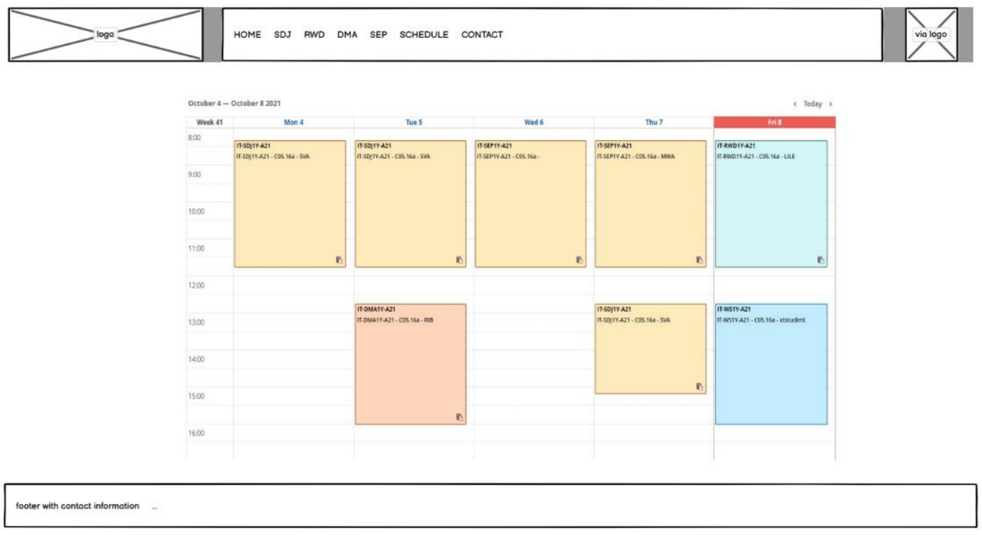
The rest of the Balsamiq Wireframes designs can be found in APPENDIX FOLDER 3 – BALSAMIQ.

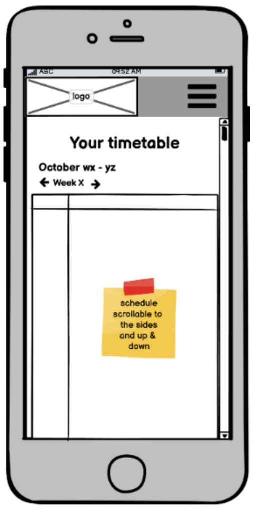
What follows is a sequence diagram for Scheduling a Session. To put it briefly, the Timetable Manager fills out fields with information about the Session at hand, the system checks for overlaps and, if everything is in order, a Session is scheduled.


Timetable Manager : Bob
Class : ScheduleSessController
Class : VIA
1: scheduleSessionsButtonPressed()
2: Choose a semester()
2.1: changed()
The changed method will change the class and the course contents, according to the semester chosen
3: Choose a class()
4: Choose a course()
5: Choose a date : datechanged()
5.1: displayAvailableClassrooms()
5.1.1: availableClassrooms = returnAvailableClassrooms()
The displayAvailableClassrooms() method will fetch which classrooms are available for the Date and Timeframe chosen and return them as a List. This list will be displayed inside of the "Classroom" choice box everytime a change is made to the Date, Start Time or Number of Lessons so that the classrooms choices are always valid.
6: Choose a start time : startTimeChanged()
6.1: displayAvailableClassrooms()
6.1.1: availableClassrooms = returnAvailableClassrooms()
7: Choose the number of lessons : numLessonsChanged()
7.1: displayAvailableClassrooms()
7.1.1: availableClassrooms = returnAvailableClassrooms()
8: Chose how many weeks to repeat()
9: Choose a classroom()
10: scheduleButton()
alt : if
10.1: Check for overlap : scheduleSession()
Overlapping
If there are overlaps, an exception is thrown and the method does not reach its end.
alt : if there are no overlaps
11: Export changes : exportClassSessionList()
12: Save changes : saveCurrentChanges()

The website also required designing before being implemented. As for the part of it that is most relevant to this project, the Schedule page, two layouts were set out for different screen sizes: Desktop and Mobile. On Desktop, the timetable shows in full extension, while on the phone, it is scrollable to the sides due to the small size of the screen. This does not hinder navigability, however, as the timetable cells maintain a considerable size as opposed to shrinking unreasonably.

The designs produced on Balsamiq Wireframes are below. They can be found in more detail in APPENDIX FOLDER 3 – BALSAMIQ.

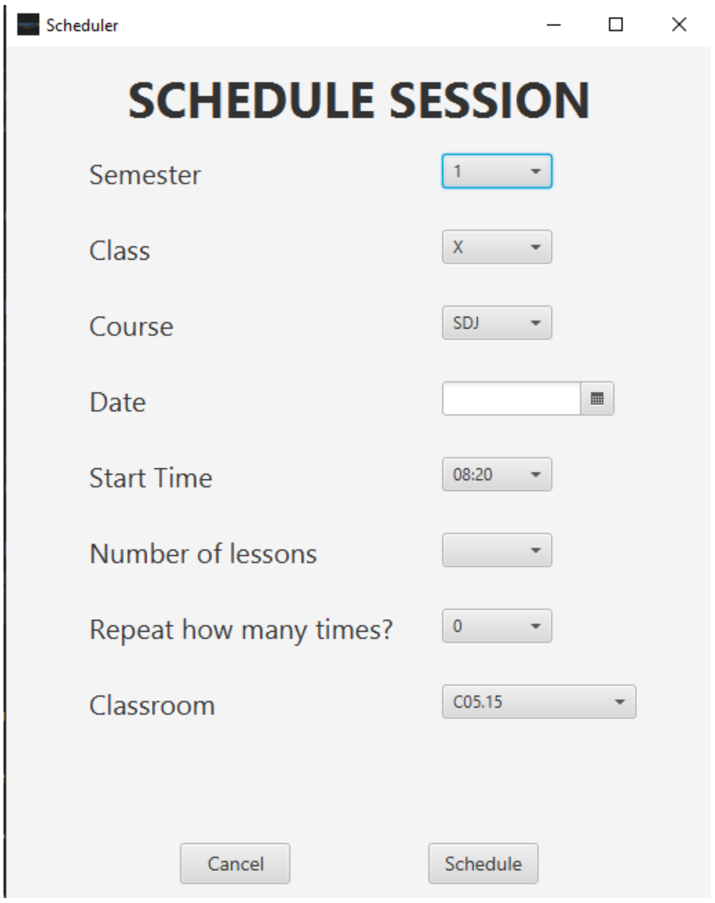

logo
HOME
SDJ
RWD
DMA
SEP
SCHEDULE
CONTACT
via logo
October 4 — October 8 2021
‹ Today ›
Week 41
Mon 4
Tue 5
Wed 6
Thu 7
Fri 8
8:00
9:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
IT-SDJ1Y-A21
IT-SDJ1Y-A21 - C05.16a - SVA
IT-SDJ1Y-A21
IT-SDJ1Y-A21 - C05.16a - SVA
IT-SEP1Y-A21
IT-SEP1Y-A21 - C05.16a -
IT-SEP1Y-A21
IT-SEP1Y-A21 - C05.16a - MWA
IT-RWD1Y-A21
IT-RWD1Y-A21 - C05.16a - LILE
IT-DMA1Y-A21
IT-DMA1Y-A21 - C05.16a - RIB
IT-SDJ1Y-A21
IT-SDJ1Y-A21 - C05.16a - SVA
IT-WS1Y-A21
IT-WS1Y-A21 - C05.16a - xtstudent
footer with contact information ...



4 Implementation

Implementation was conducted in a structured manner:

In the first place, the GUI windows were created in Scene Builder according to the design initially proposed. The window for "Schedule Session" turned out to be as follows:



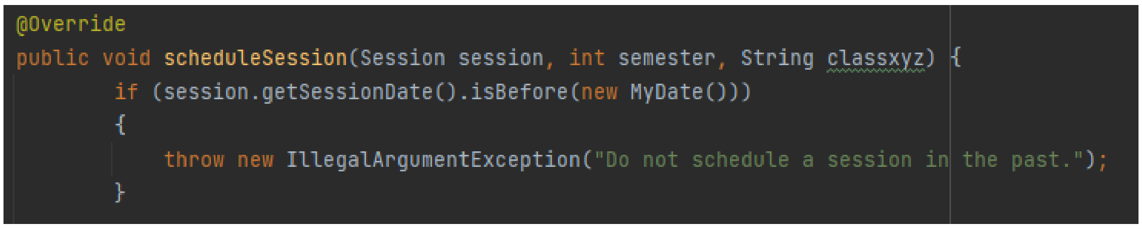
Scheduler

SCHEDULE SESSION

Semester	1
Class	X
Course	SDJ
Date	
Start Time	08:20
Number of lessons	
Repeat how many times?	0
Classroom	C05.15

This window makes use of a multiplicity of Choice Boxes that display and retrieve the information relevant for scheduling a session, one Date Picker and two Buttons: one for going back and one for confirming the Scheduling.

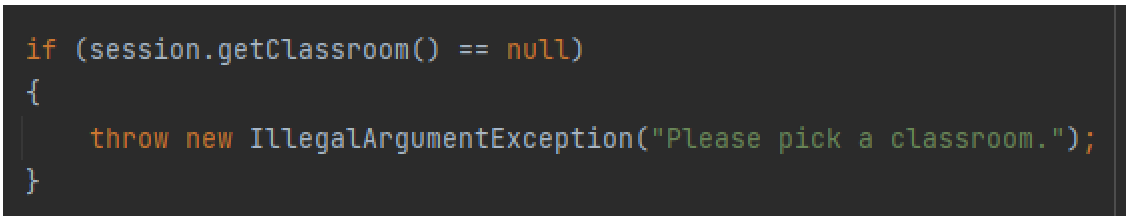
The next step was to implement the Model classes. This was done inside of IntelliJ. They were all implemented according to the Class Diagram shown in the Design chapter. A method that could be worth discussing in detail could be "`scheduleSession`" inside of the VIA class.



```
@Override
public void scheduleSession(Session session, int semester, String classxyz) {
        if (session.getSessionDate().isBefore(new MyDate()))
        {
            throw new IllegalArgumentException("Do not schedule a session in the past.");
        }
```

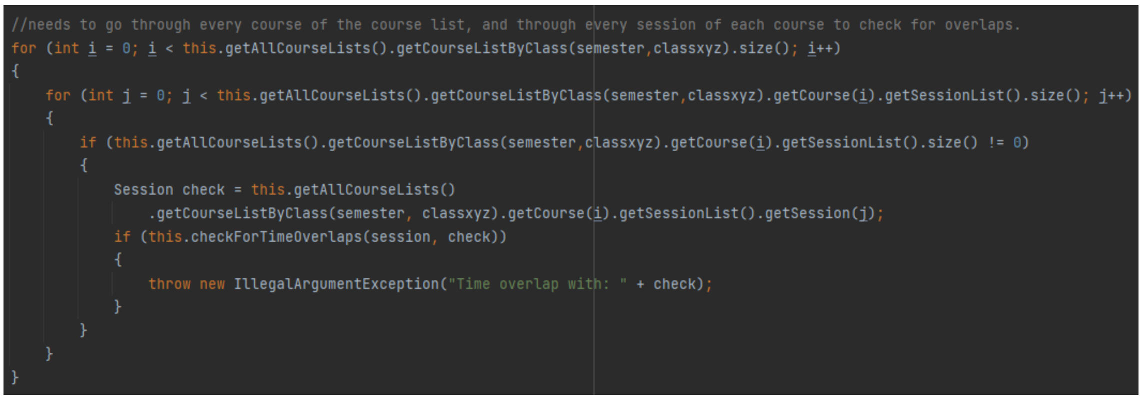
The method takes as an argument a Session object, an int for the semester and a String for the class (x,y,z...). This data is all fetched inside of the Schedule Session window, as mentioned before.

The first thing it does is check whether the date trying to be scheduled is in the past. If so, the program will throw an exception warning the user not to do this.



```
if (session.getClassroom() == null)
{
    throw new IllegalArgumentException("Please pick a classroom.");
}
```

Next, it checks whether the Classroom field is empty. Whenever the Start Time, Number of Lessons or Date are changed, the Classroom field is updated and set to empty for the user to pick a classroom once again. This is why this check was necessary.

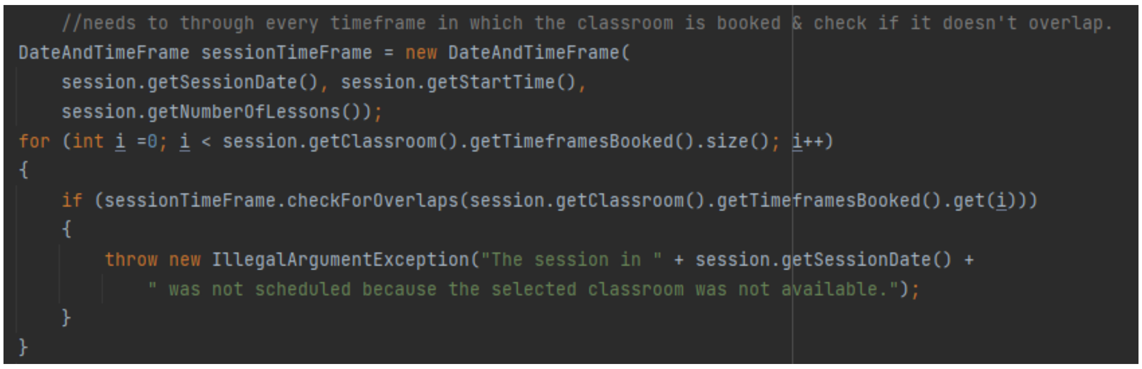


```
//needs to go through every course of the course list, and through every session of each course to check for overlaps.
for (int i = 0; i < this.getAllCourseLists().getCourseListByClass(semester,classxyz).size(); i++)
{
    for (int j = 0; j < this.getAllCourseLists().getCourseListByClass(semester,classxyz).getCourse(i).getSessionList().size(); j++)
    {
        if (this.getAllCourseLists().getCourseListByClass(semester,classxyz).getCourse(i).getSessionList().size() != 0)
        {
            Session check = this.getAllCourseLists()
                .getCourseListByClass(semester, classxyz).getCourse(i).getSessionList().getSession(j);
            if (this.checkForTimeOverlaps(session, check))
            {
                throw new IllegalArgumentException("Time overlap with: " + check);
            }
        }
    }
}
```

What follows is a lengthier check: in order for there not to be time overlaps, the method goes through every Session featured in that class's `CourseList` to check if any of them overlap with the Session trying to be booked. If they do overlap, an exception will be thrown telling the user which session is overlapping with the one at hand.

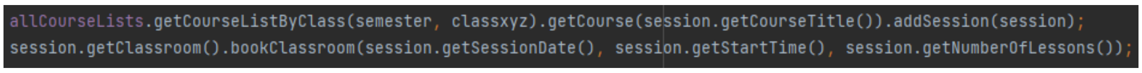
The **time complexity** of this method is O(n*m), being "n" the number of `Courses` inside of the class's `CourseList`, and "m" the amount of `Sessions` inside of each `Course's SessionList`. The reason why it is O(n*m) is because there are two nested for loops. There are two nested "if" statements inside of the nested "for" loop, but they only represent one time unit each, so each of them will happen, at most, (n*m) times. This does not affect the final time complexity.

Lastly, the method goes through all of the “timeframes booked” for the picked classroom to see if it is available for the timeframe in which the session is trying to be booked. If it is unavailable, an exception will be thrown.



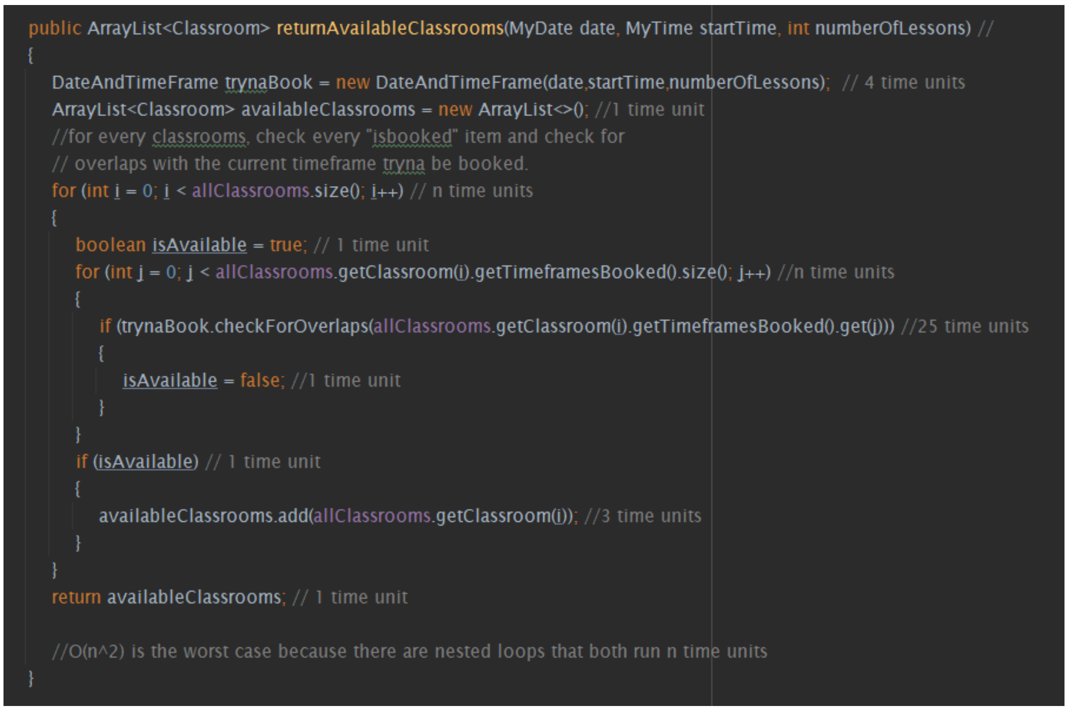
```java
    //needs to through every timeframe in which the classroom is booked & check if it doesn't overlap.
DateAndTimeFrame sessionTimeFrame = new DateAndTimeFrame(
    session.getSessionDate(), session.getStartTime(),
    session.getNumberOfLessons());
for (int i =0; i < session.getClassroom().getTimeframesBooked().size(); i++)
{
    if (sessionTimeFrame.checkForOverlaps(session.getClassroom().getTimeframesBooked().get(i)))
    {
        throw new IllegalArgumentException("The session in " + session.getSessionDate() +
            " was not scheduled because the selected classroom was not available.");
    }
}
```

And finally, the session is booked: it is added to the chosen `Course’s SessionList` and the according `Classroom` is booked for the timeframe picked.



```java
allCourseLists.getCourseListByClass(semester, classxyz).getCourse(session.getCourseTitle()).addSession(session);
session.getClassroom().bookClassroom(session.getSessionDate(), session.getStartTime(), session.getNumberOfLessons());
```

Another method whose time complexity could be analysed is "`returnAvailableClassrooms`", due to the fact that this is a method that is ran every time anything related to time is changed inside of the "Schedule Session" window.



```
public ArrayList<Classroom> returnAvailableClassrooms(MyDate date, MyTime startTime, int numberOfLessons) //
{
  DateAndTimeFrame trynaBook = new DateAndTimeFrame(date,startTime,numberOfLessons);  // 4 time units
  ArrayList<Classroom> availableClassrooms = new ArrayList<>(); //1 time unit
  //for every classrooms, check every "isbooked" item and check for
  // overlaps with the current timeframe tryna be booked.
  for (int i = 0; i < allClassrooms.size(); i++) // n time units
  {
    boolean isAvailable = true; // 1 time unit
    for (int j = 0; j < allClassrooms.getClassroom(i).getTimeframesBooked().size(); j++) //n time units
    {
      if (trynaBook.checkForOverlaps(allClassrooms.getClassroom(i).getTimeframesBooked().get(j))) //25 time units
      {
        isAvailable = false; //1 time unit
      }
    }
    if (isAvailable) // 1 time unit
    {
      availableClassrooms.add(allClassrooms.getClassroom(i)); //3 time units
    }
  }
  return availableClassrooms; // 1 time unit

  //O(n^2) is the worst case because there are nested loops that both run n time units
}
```

This method returns the available classrooms and it has big O(n^2) time complexity because of the nested loops.

First, the local variable `trynaBook` is created and initialized, which takes 4 time units.

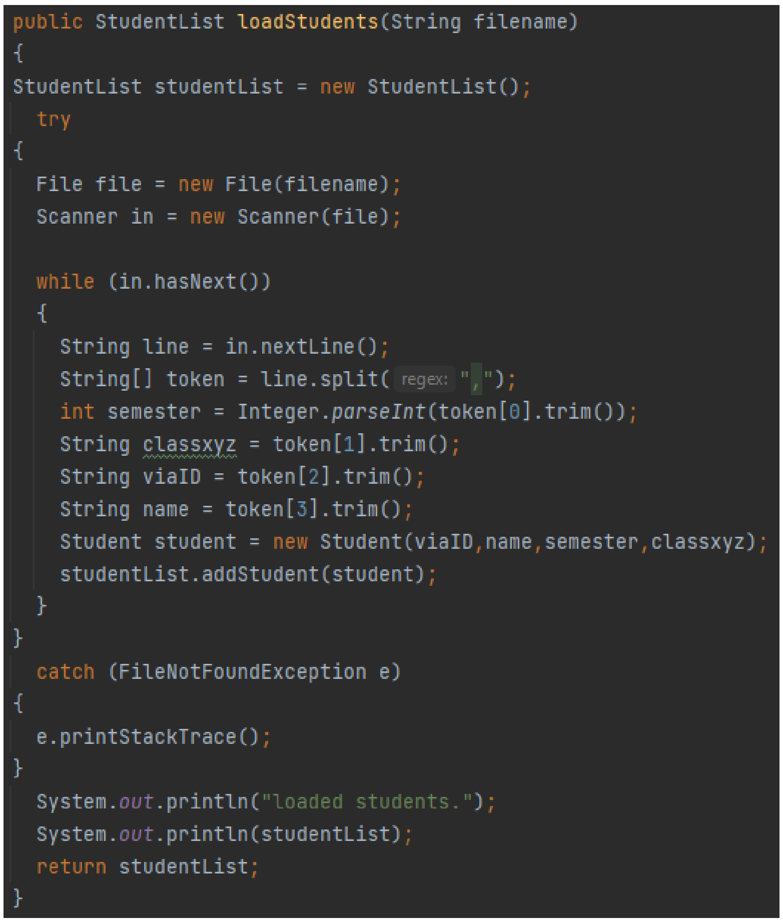
Afterwards, `ArrayList` is initialized, which takes 1 time unit.

The first "for" loop will run `allClassrooms.size()` times, will be simplified as being said to run "n" times. It will first initialize `isAvailable` with the value true, and then it is going to enter another for loop which will run `allClassrooms.getClassroom(i).getTimeFramesBooked().size()` times. In order to simplify as much as possible, and because the for-loop unit times are changing every increment of the first for loop, this loop is also going to be said to run for "n" time units.

The second loop will run an "if" statement **n** times, each time generating 25 time units in the background, because this is the runtime of `checkForOverlaps`. If `isAvailable` remains as it is, it is going to be added to the local variable `availableClassrooms`. If not, then it will not be added.

At the end, the method is going to return the `availableClassrooms` ArrayList.

It could also be relevant to talk about how the information is retrieved from the external .txt files and incorporated inside of the program.



```
public StudentList loadStudents(String filename)
{
StudentList studentList = new StudentList();
  try
{
  File file = new File(filename);
  Scanner in = new Scanner(file);

  while (in.hasNext())
  {
    String line = in.nextLine();
    String[] token = line.split( regex: ",");
    int semester = Integer.parseInt(token[0].trim());
    String classxyz = token[1].trim();
    String viaID = token[2].trim();
    String name = token[3].trim();
    Student student = new Student(viaID,name,semester,classxyz);
    studentList.addStudent(student);
  }
}
  catch (FileNotFoundException e)
{
  e.printStackTrace();
}
  System.out.println("loaded students.");
  System.out.println(studentList);
  return studentList;
}
```

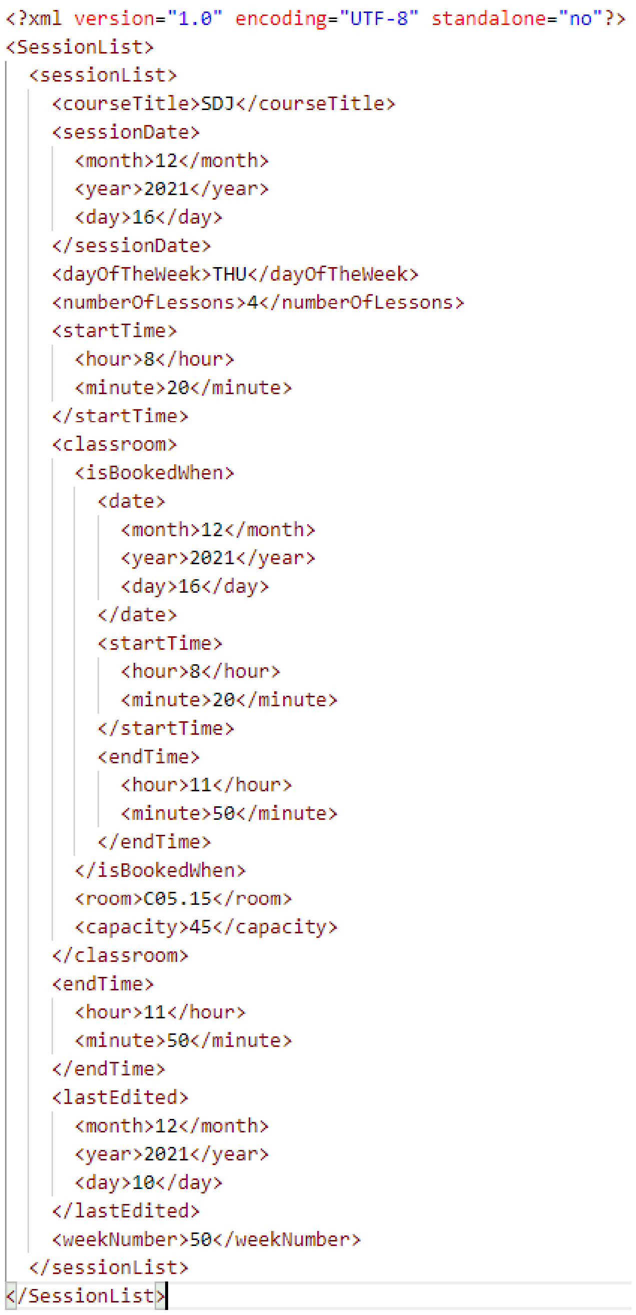
On the picture above is a method called “`loadStudents`”. This method exists inside of the class “`FileManager`”, part of the “`persistence`” package, which is responsible for the exchange of information between the program and the .txt external files.

What this method does is go through every line of the .txt file, parse information from there, create a `Student` object for each line and add it to a `StudentList` that is returned at the end of the method. This method is useful for the constructor of the model manager, VIA. Inside of the constructor, VIA’s `StudentList` is initialized to the one that is returned by this method from the file.

The implementation about how the website is able to display information about schedules created inside of the program is also worth analysing in detail.

Inside of the Java application, there is a function that exports the current Sessions scheduled for a class to an XML file. This XML file is tailored to be read and displayed in the website by JavaScript.

An example of one of these XML files could look something like this:



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SessionList>
  <sessionList>
    <courseTitle>SDJ</courseTitle>
    <sessionDate>
      <month>12</month>
      <year>2021</year>
      <day>16</day>
    </sessionDate>
    <dayOfTheWeek>THU</dayOfTheWeek>
    <numberOfLessons>4</numberOfLessons>
    <startTime>
      <hour>8</hour>
      <minute>20</minute>
    </startTime>
    <classroom>
      <isBookedWhen>
        <date>
          <month>12</month>
          <year>2021</year>
          <day>16</day>
        </date>
        <startTime>
          <hour>8</hour>
          <minute>20</minute>
        </startTime>
        <endTime>
          <hour>11</hour>
          <minute>50</minute>
        </endTime>
      </isBookedWhen>
      <room>C05.15</room>
      <capacity>45</capacity>
    </classroom>
    <endTime>
      <hour>11</hour>
      <minute>50</minute>
    </endTime>
    <lastEdited>
      <month>12</month>
      <year>2021</year>
      <day>10</day>
    </lastEdited>
    <weekNumber>50</weekNumber>
  </sessionList>
</SessionList>
```

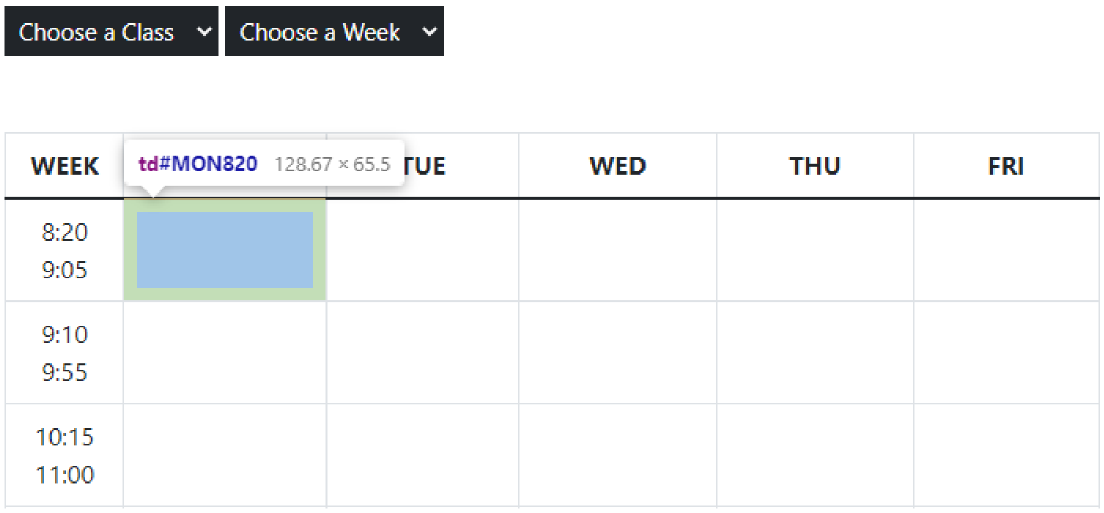
The XML file in the previous page contains only one session. Inside of each session is stored a lot of information:

- Session Date
- Start Time
- Day of the Week
- Number of Lessons
- Classroom
- End Time
- Last Edited Date
- Week Number

All of these variables will be read by JavaScript and displayed inside of the timetable.

In order for them to be displayed accurately, all of the table cells of the timetable inside of the website were given a unique ID.

For example, the table cell that is the lesson on Monday at 8:20 was given the ID "#MON820".



Choose a Class | Choose a Week

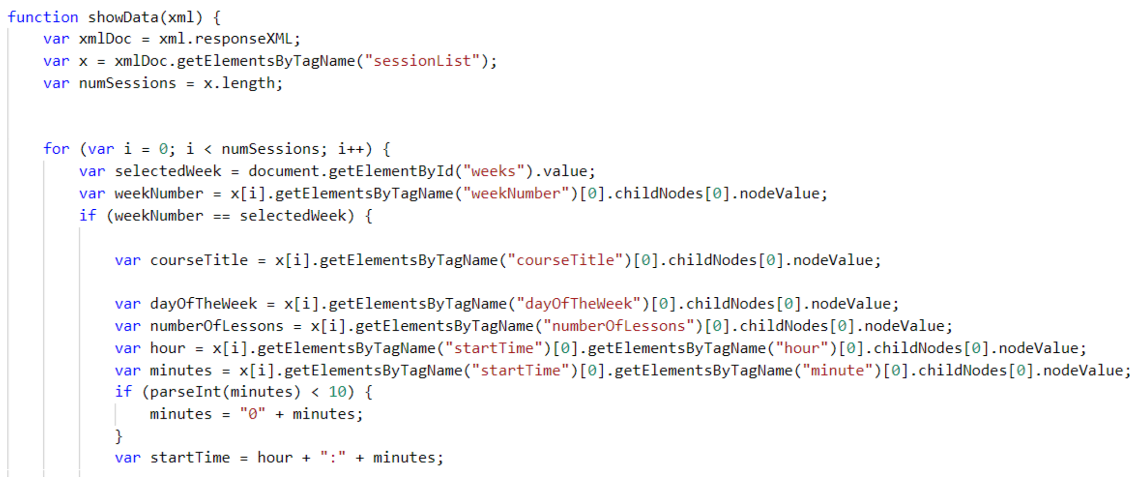
td#MON820 128.67 × 65.5

WEEK		TUE	WED	THU	FRI
8:20 9:05					
9:10 9:55					
10:15 11:00					

With these ID's, it was possible to infer where each session should be placed inside of the table. However, some sessions may have the length of more than one lesson. For this to be displayed in the timetable, the solution found was to:

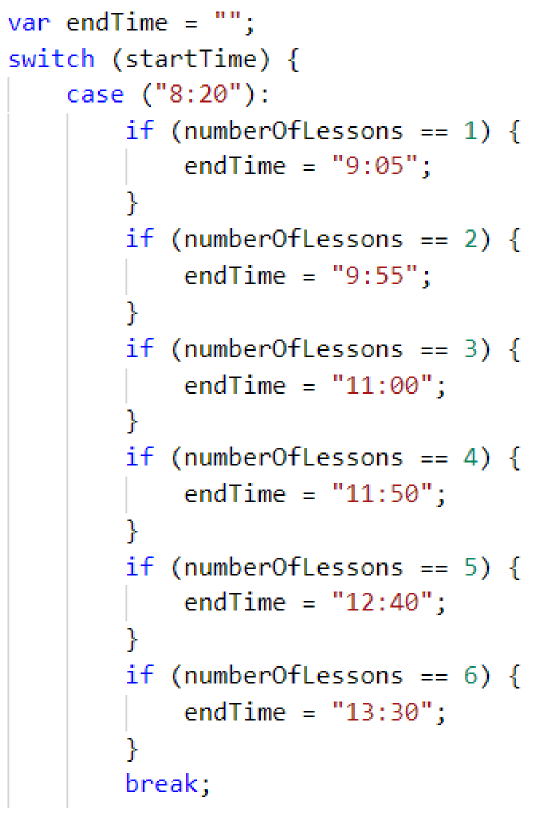
- Set the "rowspan" attribute of the table cell to the number of lessons. This way, the cell would be taller the longer the session.
- Delete the table cells that were "overridden" by the longer lesson. This was necessary because otherwise, they would be shifted to the right side and the timetable would become inaccurate.

The following screenshot portrays how the information was fetched for each session:



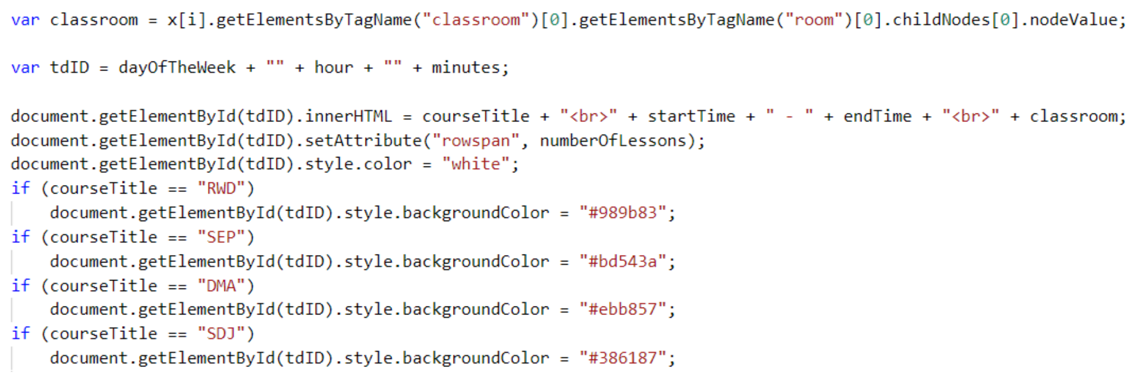
```
function showData(xml) {
    var xmlDoc = xml.responseXML;
    var x = xmlDoc.getElementsByTagName("sessionList");
    var numSessions = x.length;


    for (var i = 0; i < numSessions; i++) {
        var selectedWeek = document.getElementById("weeks").value;
        var weekNumber = x[i].getElementsByTagName("weekNumber")[0].childNodes[0].nodeValue;
        if (weekNumber == selectedWeek) {

            var courseTitle = x[i].getElementsByTagName("courseTitle")[0].childNodes[0].nodeValue;

            var dayOfTheWeek = x[i].getElementsByTagName("dayOfTheWeek")[0].childNodes[0].nodeValue;
            var numberOfLessons = x[i].getElementsByTagName("numberOfLessons")[0].childNodes[0].nodeValue;
            var hour = x[i].getElementsByTagName("startTime")[0].getElementsByTagName("hour")[0].childNodes[0].nodeValue;
            var minutes = x[i].getElementsByTagName("startTime")[0].getElementsByTagName("minute")[0].childNodes[0].nodeValue;
            if (parseInt(minutes) < 10) {
                minutes = "0" + minutes;
            }
            var startTime = hour + ":" + minutes;
```

The end time was fetched through a “switch” statement due to the fact that JavaScript was not accurately fetching its value.



```
var endTime = "";
switch (startTime) {
    case ("8:20"):
        if (numberOfLessons == 1) {
            endTime = "9:05";
        }
        if (numberOfLessons == 2) {
            endTime = "9:55";
        }
        if (numberOfLessons == 3) {
            endTime = "11:00";
        }
        if (numberOfLessons == 4) {
            endTime = "11:50";
        }
        if (numberOfLessons == 5) {
            endTime = "12:40";
        }
        if (numberOfLessons == 6) {
            endTime = "13:30";
        }
        break;
```

The screenshot below portrays how the information came to be inside of the correct table cell.



```
var classroom = x[i].getElementsByTagName("classroom")[0].getElementsByTagName("room")[0].childNodes[0].nodeValue;

var tdID = dayOfTheWeek + "" + hour + "" + minutes;

document.getElementById(tdID).innerHTML = courseTitle + "<br>" + startTime + " - " + endTime + "<br>" + classroom;
document.getElementById(tdID).setAttribute("rowspan", numberOfLessons);
document.getElementById(tdID).style.color = "white";
if (courseTitle == "RWD")
    document.getElementById(tdID).style.backgroundColor = "#989b83";
if (courseTitle == "SEP")
    document.getElementById(tdID).style.backgroundColor = "#bd543a";
if (courseTitle == "DMA")
    document.getElementById(tdID).style.backgroundColor = "#ebb857";
if (courseTitle == "SDJ")
    document.getElementById(tdID).style.backgroundColor = "#386187";
```

For example, if the session starts on Monday at 8:20:

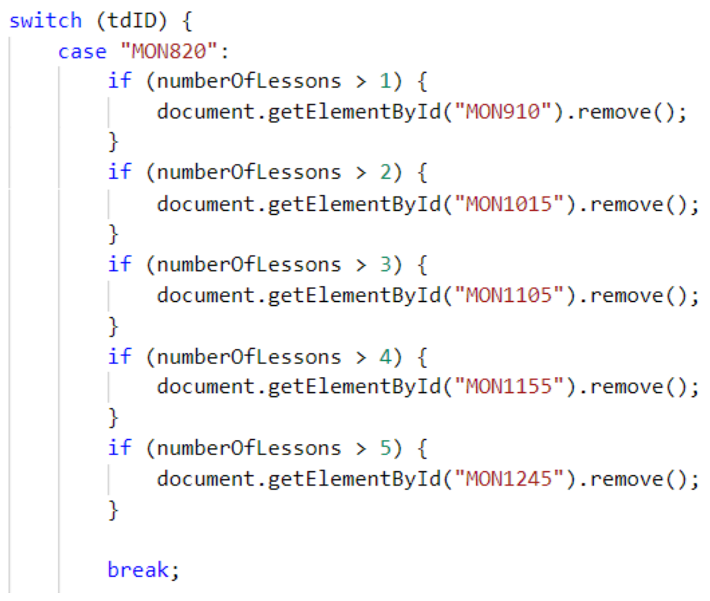
A variable called “tdID” will be created with the day of the week (“MON”), the starting hour (“8”) and the starting minutes (“20”).

Therefore, its final value will be “MON820”.

It was mentioned before that each table cell has its unique ID, and they are also in this format. Therefore, “tdID” is the method’s way of finding in which cell to insert the information for the current session through its unique ID.

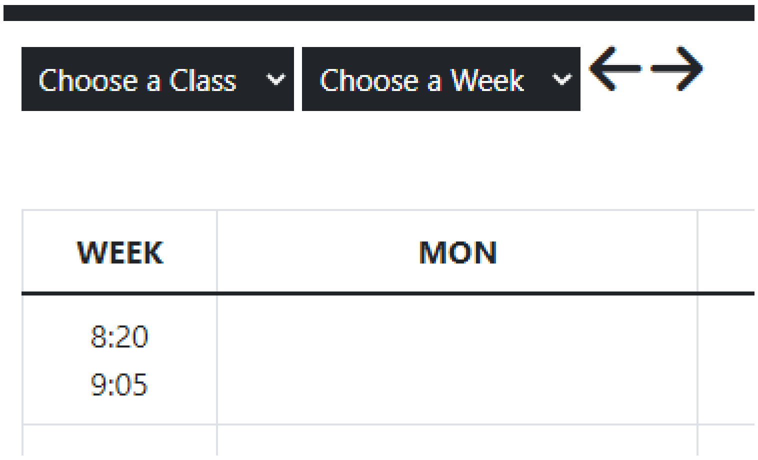
After that, the information is put inside of the table cell, its text color is set to white and the background color is set differently depending on the course. The only courses taken into account were the ones included in the “sample” .txt files.

Lastly, the removing of the “extra” table cells:



```
switch (tdID) {
    case "MON820":
        if (numberOfLessons > 1) {
            document.getElementById("MON910").remove();
        }
        if (numberOfLessons > 2) {
            document.getElementById("MON1015").remove();
        }
        if (numberOfLessons > 3) {
            document.getElementById("MON1105").remove();
        }
        if (numberOfLessons > 4) {
            document.getElementById("MON1155").remove();
        }
        if (numberOfLessons > 5) {
            document.getElementById("MON1245").remove();
        }

        break;
```

The Schedule page of the website also includes two dropdown menus and two buttons designed for a better navigability of the timetables.

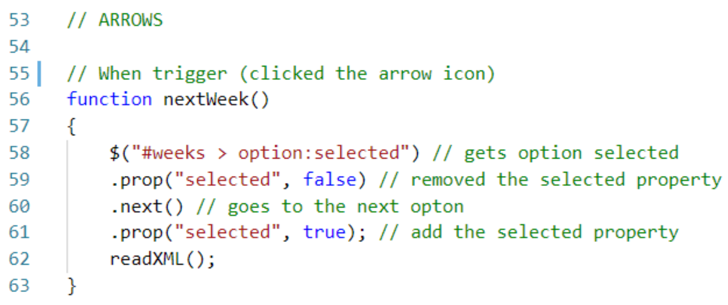


The dropdown menu "Choose a Class" gets its options from an XML file that is also generated inside of the application and tailored for this purpose: it solely contains the existing classes: e.g., 1X , 1Y, 1Z… When a class is picked, the Session XML file that is read is the one corresponding to it (e.g., "1X.xml").

```
//this function filters the information of the xml
function readClasses(xml) {
    var xmlDoc = xml.responseXML; // gets the response
    var Classs = $(xmlDoc).find("Class"); // finds the "Class" Object
    var classxyz = $("#classxyz"); // Finds where to add that information, in this case it's a select box
    Classs.each(function () { // For every found "Class" will add an option for the select
        var option = $("<option />");
        option.html($(this).find("ClassName").text());
        classxyz.append(option);
    });
}
```

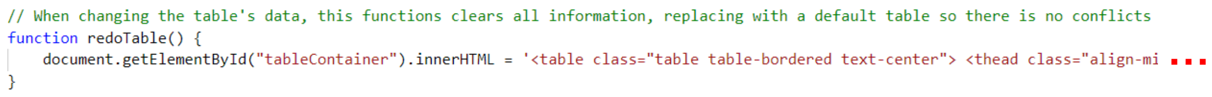
Additionally, the dropdown menu "Choose a Week" has the purpose of preventing the overlapping of Sessions from different weeks. For example, if there were two sessions: one on the 7/12/21 (TUE) and 22/12/21 (WED) and this option did not exist, it would seem in the timetable that they existed in the same week, when in reality, they are two weeks apart. In order to achieve this, the variable "`weekNumber`" inside of `Session` was used. So, when the selected week corresponds to a Session's week number, it will be displayed inside of the timetable. This way, it is possible to navigate through weeks without overlaps.

The two arrow buttons exist also to facilitate navigation: instead of having to open the dropdown menu every time, the user can just go to the previous or next week with one click.



```
// ARROWS

// When trigger (clicked the arrow icon)
function nextWeek()
{
    $("#weeks > option:selected") // gets option selected
    .prop("selected", false) // removed the selected property
    .next() // goes to the next opton
    .prop("selected", true); // add the selected property
    readXML();
}
```

Lastly, another need that arose was to “redo” the table every time a new option was selected. This is because some schedules delete table cells, as mentioned before, so when another schedule was selected, it might not have all of the cells to its disposal and could appear inaccurately. In order to prevent this, a function “`redoTable()`” was created, which clears the content of the container table and recreates all of the table cells.



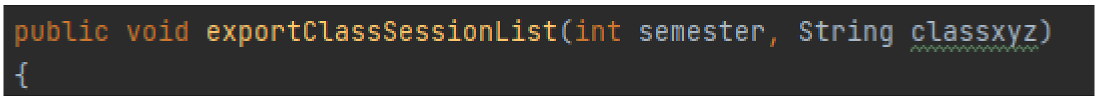
```
// When changing the table's data, this functions clears all information, replacing with a default table so there is no conflicts
function redoTable() {
    document.getElementById("tableContainer").innerHTML = '<table class="table table-bordered text-center"> <thead class="align-mi ...
}
```

To wrap up the Implementation chapter, it might be interesting to talk about how a high priority requirement related to both the “Schedule Session” use case and the website implementation:

9. As a timetable manager, I want the changes made to the schedules to be immediately available for students and teachers to see, so that they are always up to date with their timetables.

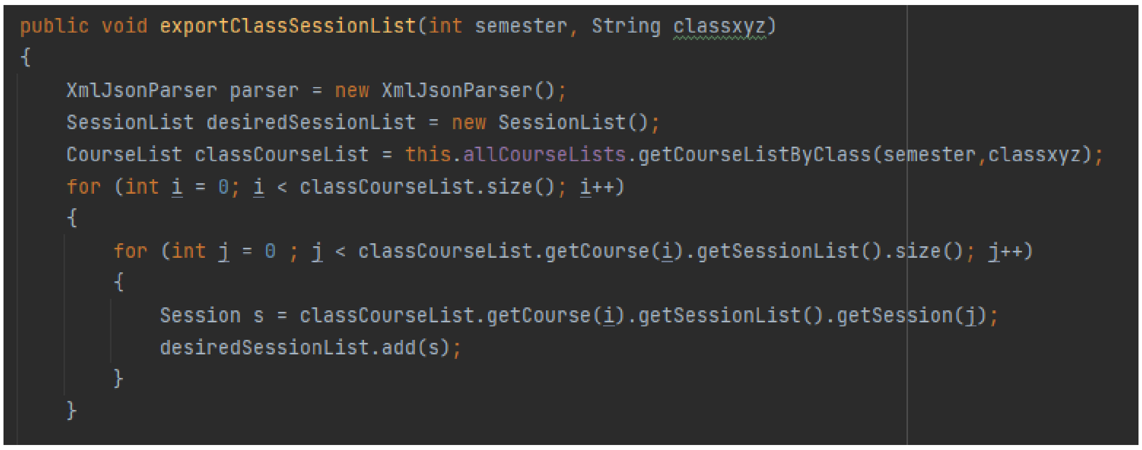
It was achieved in the following way:

Every time any changes are made to the schedule, this method is called:



```
public void exportClassSessionList(int semester, String classxyz)
{
```

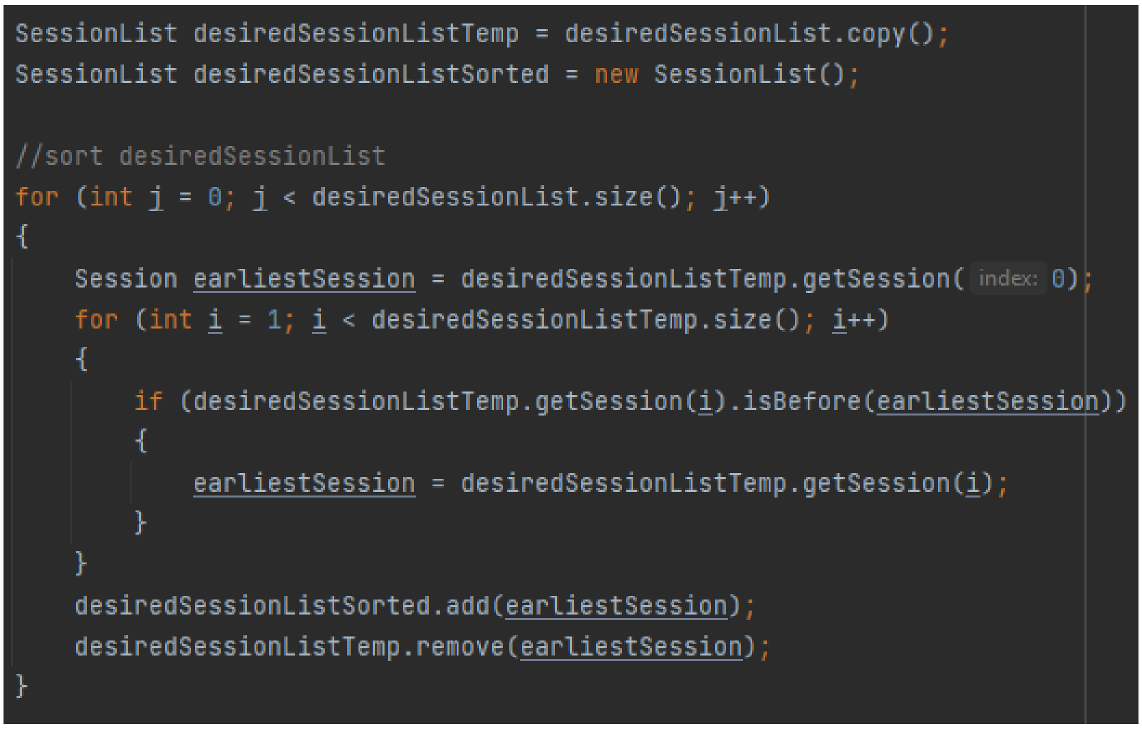
What this method does is <u>access the website directory</u> and export an XML file there. The XML filename depends on the semester and class at hand: for example, if a session was scheduled for class 1X, the XML file edited would be “1X.xml”. The direct consequence of this is that the next time the website is opened, the changes made are immediately available, and the requirement at hand is met. Code snippets for the method will be presented on the following page:



```java
public void exportClassSessionList(int semester, String classxyz)
{
    XmlJsonParser parser = new XmlJsonParser();
    SessionList desiredSessionList = new SessionList();
    CourseList classCourseList = this.allCourseLists.getCourseListByClass(semester,classxyz);
    for (int i = 0; i < classCourseList.size(); i++)
    {
        for (int j = 0 ; j < classCourseList.getCourse(i).getSessionList().size(); j++)
        {
            Session s = classCourseList.getCourse(i).getSessionList().getSession(j);
            desiredSessionList.add(s);
        }
    }
```

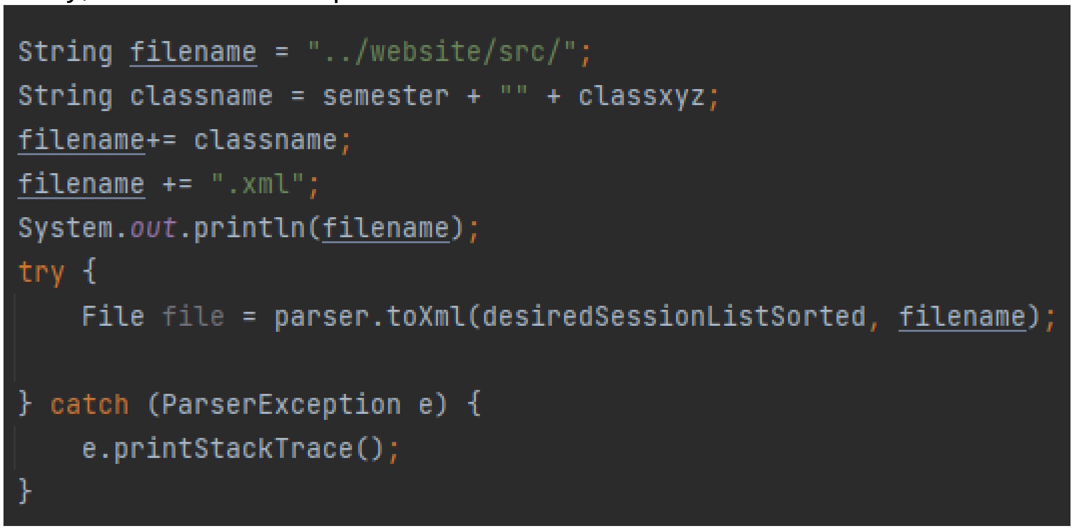
Firstly, all of the Sessions for the desired class are put inside of a "`desiredSessionList`" that will be exported to XML at the end of the method.

Then, they are sorted from earliest to latest and put inside of a sorted version of `desiredSessionList`.



```java
SessionList desiredSessionListTemp = desiredSessionList.copy();
SessionList desiredSessionListSorted = new SessionList();

//sort desiredSessionList
for (int j = 0; j < desiredSessionList.size(); j++)
{
    Session earliestSession = desiredSessionListTemp.getSession( index: 0);
    for (int i = 1; i < desiredSessionListTemp.size(); i++)
    {
        if (desiredSessionListTemp.getSession(i).isBefore(earliestSession))
        {
            earliestSession = desiredSessionListTemp.getSession(i);
        }
    }
    desiredSessionListSorted.add(earliestSession);
    desiredSessionListTemp.remove(earliestSession);
}
```

Lastly, the sorted list is exported to an XML file inside of the source folder for the website.



```
String filename = "../website/src/";
String classname = semester + "" + classxyz;
filename+= classname;
filename += ".xml";
System.out.println(filename);
try {
    File file = parser.toXml(desiredSessionListSorted, filename);

} catch (ParserException e) {
    e.printStackTrace();
}
```

So if any changes were made to the 1X Session List, whoever opened the website and picked 1X would see the new changes in the timetable.

5 Test

In order for the implemented system to be tested, the approach chosen was to go through each use case, define an expected end result, test it and document whether the desired end result was met or not.

The results can be observed in the table below.

The "Reschedule Session table" is referenced frequently because it is a way to see information about the scheduled sessions.

Use Case	Expected Result	Works as expected?
Schedule Session	The session scheduled appears in the "Reschedule Session" table when correctly searched for, meaning it was successfully created.	Yes
Reschedule Session	The session rescheduled has its information updated according to what was rescheduled. This can be seen in the "Reschedule Session" table.	Yes
Cancel Session	The session cancelled does no longer appear in the "Reschedule Session" table.	Yes
Switch student's class	The student's class is updated to the new one, which can be observed in the "Manage Students" window.	Yes
View own timetable	The sessions scheduled appear in the website when the right class and week are selected.	Yes

6 Results and Discussion

When contemplating the final product, it was possible to determine which initially set requirements were met.

Requirement	Met?	Description
		CRITICAL PRIORITY
1	Yes	As a timetable manager, I want to schedule a session so that teachers and students know when to get together for a session.
2	Yes	As a timetable manager, I want students and teachers to be assigned to the courses that they learn or teach so that these appear on their timetables
3	Yes	As a timetable manager, I want to book classrooms for sessions so that students and teachers have a place to get together.
4	Yes	As a timetable manager, I want to be able to cancel sessions so that there are more classrooms available for other sessions.
5	Yes	As a timetable manager, the access to the planning tool should be limited to me so that I am the only one able to modify schedules.
6	Yes	As a timetable manager, I want to see only which classrooms are available when scheduling a class so that there are no overlaps
7	Yes	As a student, I want to see my timetable so that I know when and where I have sessions.
		HIGH PRIORITY
8	Yes	As a timetable manager, I want to reschedule sessions so that unforeseen events (illness, personal problems, events) do not get in the way of the intended number of lessons for each course.
9	Yes	As a timetable manager, I want the changes made to the schedules to be immediately available for students and teachers to see, so that they are always up to date with their timetables.
10	Yes	As a timetable manager, I want to remove students from a class so that I can assign them to another class if they have requested a switch.
11	Yes	As a timetable manager, I want to add students to a class so that they can have access to a new timetable if they have requested to switch classes.
12	No	As a timetable manager, I want to be able to assign two teachers to the same course when the course requires it.

13	No	As a timetable manager, I want to remove students from a course so that it will not appear in their timetable.
14	No	I want to be able to edit enrollment in a course so that I can add individual students to specific courses.
LOW PRIORITY		
15	No	As a timetable manager, I want to be able to double the capacity of a room and merge two classes if it has a foldable wall.
16	Yes	As a user of this system, I want to see my schedule on a weekly format, so that I can see the sessions for each day.
17	Yes	As a user of this system, I want to be able to filter only the schedules that correspond to my class.
18	No	As a user of this system, I want to be warned when sudden changes appear in the schedules.
NON-FUNCTIONAL REQUIREMENTS		
19	No	As a timetable manager, when assigning a student with credits to a course from another semester, I want the system to check if a timetable without overlaps is available for this student.

Although not all initially set requirements were met, the end result was still satisfactory. By this, it is meant that every critical requirement was fulfilled, and also that the final product is a good foundation for a possible future version of the program in which all other requirements can be implemented.

Looking back at it critically, a few aspects could have been implemented in more convenient ways:

- The `Session` class could have had a `DateAndTimeFrame` instance variable instead of two `MyTime` and one `MyDate` variable to define the timeframe in which it is booked.
- Instead of the `Courses` having a `SessionList`, there could have been a class for each Class (e.g., 1X, 1Y…), and only inside that class would there be a `SessionList`. It was implemented in this non-optimal way because the first plan was for each student to have an individual timetable, in which their `Sessions` would be fetched from the `Courses` to which the `Student` was assigned.

However, given the way the system ended up, `Sessions` could have been booked for directly for each Class instead of for each `Course`.

- The use of files for persistence is far from optimal. Every time any change is made to the system, a new .json file and new .xml are created with all of the information, overriding the previous files, as opposed to simply appending to what was previously there, no matter how little the change.

7 Conclusions

In summary, each step of the process of developing the project at hand build on top of the previous one, eventually leading to a satisfactory end product. It was first necessary to formulate the Problem Description for the client's present situation, extract the requirements for the system during the Analysis, and only then was it possible to become solution-oriented and start designing how to implement these use cases through a Java Application and Website that came to existence during the Implementation phase. Then, in possession of the product, a test phase was conducted, and the results were positive: both application and website functioned as intended, and enough requirements were fulfilled by the project for the solution to be considered viable.

8 Sources and References

Flaticon. 2021. Detailed Flat Circular Icon Style / Flat - 39,741 vector icons available in SVG, EPS, PNG, PSD files and Icon Font.. [online] Available at: <https://www.flaticon.com/authors/detailed-flat-circular/flat> [Accessed 14 December 2021].

Gandy, D., 2021. Font Awesome. [online] Fontawesome.com. Available at: <https://fontawesome.com/> [Accessed 14 December 2021].

Miranda, R., 2021. Javafx Cascading dropdown based on selection. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/20794359/javafx-cascading-dropdown-based-on-selection> [Accessed 14 December 2021].

Stack Overflow. 2021. JavaFX- how to save the selected from ChoiceBox. [online] Available at: <https://stackoverflow.com/questions/31605585/javafx-how-to-save-the-selected-from-choicebox> [Accessed 14 December 2021].

Stack Overflow. 2021. Making a search bar in javafx. [online] Available at: <https://stackoverflow.com/questions/47559491/making-a-search-bar-in-javafx> [Accessed 14 December 2021].

Gaddis, T., 2015. Starting Out with Java: Early Objects. 5th ed. Boston [etc.]: Pearson.

Duckett., J., 2011. HTML & CSS: Design and Build Websites. John Wiley & Sons Incorporated.

Duckett, J., Ruppert, G. and Moore, J., 2014. JavaScript & jQuery.

LaGrone, B., 2013. HTML5 and CSS3 Responsive Web Design Cookbook.

Weiss, M., 2012. Data Structures and Algorithm Analysis in Java. Boston, Mass: Addison-Wesley.

Support.google.com. 2021. What can you do with Calendar? - Google Workspace Learning Center. [online] Available at:

<https://support.google.com/a/users/answer/9302892?hl=en> [Accessed 8 October 2021].

W3schools.com. 2021. W3Schools Online Web Tutorials. [online] Available at: <https://www.w3schools.com/>.

9 Appendices

Appendices can be found inside of the APPENDICES folder inside of the handed-in .zip file.

The ones that were referenced in this report will be highlighted in yellow.

They are structured by folders in the following manner:

APPENDIX FOLDER 1 – DOCUMENTS

- APPENDIX 1A – Project Description
- APPENDIX 1B – Analysis Document
- APPENDIX 1C – Logbook
- APPENDIX 1D – User Guide
- APPENDIX 1E – Installation Guide

APPENDIX FOLDER 2 – ASTAH

- APPENDIX SUBFOLDER 2A – ACTIVITY DIAGRAM
  - APPENDIX 2Aa – Schedule Session Activity Diagram
  - APPENDIX 2Ab – Reschedule Session Activity Diagram
  - APPENDIX 2Ac – Cancel Session Activity Diagram
  - APPENDIX 2Ad – Switch Student’s Class Activity Diagram
  - APPENDIX 2Ae – View Own Timetable Activity Diagram
- APPENDIX SUBFOLDER 2B – CLASS DIAGRAM
  - APPENDIX 2Ba – Simplified Class Diagram
  - APPENDIX 2Bb – Complete Class Diagram
- APPENDIX SUBFOLDER 2C – DOMAIN MODEL
  - APPENDIX 2Ca – Domain Model
- APPENDIX SUBFOLDER 2D – USE CASE DIAGRAM
  - APPENDIX 2Da – Use Case Diagram
- APPENDIX SUBFOLDER 2E – SEQUENCE DIAGRAM
  - Appendix 2Ea – Sequence Diagram

APPENDIX FOLDER 3 – BALSAMIQ WIREFRAMES

- APPENDIX 3A – Website for Desktop Design
- APPENDIX 3B – Website for Mobile Design

- APPENDIX 3C – GUI Design

APPENDIX FOLDER 4 – SOURCE CODE

- EXTERNAL JAR FILES
- *SEP1Ygroup6* (IntelliJ Project Folder)
- *website* (Website directory)

APPENDIX FOLDER 5 – JAVADOC

- Open "index.html" to access Javadoc.

SEP1Y Group 6 Final Project

PROCESS REPORT

Daniel Lopes Adrião, 315274

Dragos Daniel Bonaparte, 315261

Laura do Bem Rebelo, 315174

Matas Armonaitis, 315263

Supervisors:

Steffen Andersen (SVA), Mona Andersen (MWA)

VIA University College

doNotFailCourse()

30.310 characters

Software Technology Engineering

1st Semester

15.12.2021

Table of content

1 Introduction

Group 6 from class Y, composed of Daniel Lopes Adrião, Dragos Daniel Bonaparte, Laura do Bem Rebelo and Matas Armonaitis worked together throughout their first semester, autumn 2021, for the course SEP1.

A recollection of their meetings throughout the Project Period can be assessed in the following table:

29/11	- **START OF PROJECT PERIOD.**
30/11	- Met from 13:00 – 17:00.
1/12	- Met from 10:00 – 15:00.
2/12	- Met from 13:00 – 16:00. -
3/12	- Met from 13:00 – 16:30.
4/12	WEEKEND
5/12	WEEKEND
6/12	- Met from 13:00 – 17:00.
7/12	- Met from 11:00 – 16:00.
8/12	- Met from 10:00 – 15:00.
9/12	- Met from 10:00 – 15:00.
10/12	- Met from 12:00 – 16:00.
11/12	WEEKEND
12/12	WEEKEND
13/12	- Met from 13:00 – 17:00.
14/12	- Met from 10:00 – 12:00, 14:00 – 16:00.
15/12	- Met from 10:00 – 16:00.

The logbook for this group can be found in more detail in APPENDIX 1C LOGBOOK. As for the tuition period, it contains information about what assignments and parts of SEP were done in each week, and as for Project Period, it contains more detailed information about what was accomplished on each day in terms of Implementation and Documentation of the Semester Project.

Only one supervisor meeting was held on the 10th of December at 14:20 with Mona Andersen, of short duration (about 20 minutes).

2 Group Description

For a four-person group, our group has a fair amount of diversity, both culturally and experience-wise. We feature two students from Portugal: Daniel and Laura, one from Romania: Dragos, and one from Lithuania: Matas, all with different programming backgrounds.

DANIEL LOPES ADRIÃO

NATIONALITY: Portuguese

PRIOR EXPERIENCE: Three-year programming course, internships related to that subject, some personal projects

DRAGOS DANIEL BONAPARTE

NATIONALITY: Romanian

PRIOR EXPERIENCE: C++ in High School

LAURA DO BEM REBELO

NATIONALITY: Portuguese

PRIOR EXPERIENCE: 1 semester of Python

MATAS ARMONAITIS

NATIONALITY: Lithuanian

PRIOR EXPERIENCE: C++ in High School

3 Project Initiation

As for how it all began, our group was formed quite occasionally: we were two pairs of people: Daniel & Laura and Dragos & Matas. Both pairs were looking for another two people to team up with in order to achieve a group of four members, so it was purely by chance that we came to be <DNFC/>.

<DNFC/> is the name of our group. It stands for "Do Not Fail Course" – which started out as a joke, but we gradually became so fond of it that it became our group name. However, it is not to be misunderstood: our goal is not merely to pass the course, but to pass it with a high grade by bringing out the best in each other and presenting high quality work.

The prompt for this project was a case presented to us, for which we had to produce a solution. The case in question is explained in detail in the Project Description – APPENDIX 1A, but in summary: VIA's timetable manager was looking for a tool that would facilitate his job of creating timetables, and we had to develop a Java application and design a website to fulfill that need.

Regarding "planning" in order to execute our project work, our planning was often light and casual. Initially, we merely followed what was asked of us: write a Project Description, make an Analysis Document, design our program, etc. But as soon as we were on our own, the one dogma that we followed was solely to set our priorities straight and make sure to make progress each time we met. This way, if we kept meeting regularly, we could make sure that our project would keep moving forward, and that it would be finished sooner or later. By "setting our priorities straight", what we mean is that it we made it our number one priority to get down the most critical requirements, as opposed to getting stuck on details that would not make that much difference in the final product. Only after the "foundations" had been concluded did we allow ourselves to dive into detail on what could be improved – polish our work. This approach proved itself to be successful: although the project period was stressful and demanding, we were always able to stay on track and did not struggle to meet the deadline. This is extremely positive, because working in a rush could have led us to sacrifice the quality of our project in order to “save time”, and the fact that we had a comfortable time margin made it possible for us to excel in our performance.

4 Project Description

It would be a lie to say that our Project Description was a smooth start. As first semester students, we were not familiar with neither Waterfall nor Report Writing. This meant that as soon as we were presented with the case, we started thinking about “how to implement it”, instead of having a structured approach to project work. Of course, as soon as we became aware of what was expected of us, we stopped, slowed down and took the time to write an appropriate Project Description. This meant focusing exclusively on the problem and not on the solution. We then realized that it is crucial to first understand the problem before considering how to approach it and solve it. Moreover, the fact that SEP encourages so much autonomous work is directly related to Problem-Based Learning: by giving us so much responsibility, this course promotes independency, is highly engaging and feels extremely rewarding (What is Problem-Based Learning (PBL) | Hun School of Princeton, 2021).

At the beginning, we set out different goals: the bare minimum was to pass the course, as our group name would suggest. However, that would not be enough for us to be satisfied: so, when drawing up a group contract, we collectively agreed that we would work hard in order to achieve a high, fulfilling grade, in order to start the Software Technology Engineering Programme on the right foot.

On a more technical note, the goals that we set out for our program were not realistic at the start - we were a bit too ambitious with what we could do with the amount of time and knowledge that we had in our possession. For example, we wanted the system to be able to provide timetables individually for each student, but it ended up only supporting timetables for whole classes (1X, 1Y…). Still, the broader goal was to successfully develop a working Java application featuring a responsive website that could read from an XML file, two things we managed to achieve. This will be discussed in more detail in the following chapter.

5 Project Execution

In order for our project to consistently move forward, it was required for us to follow a methodology. As first semester students, we were expected to vaguely follow the Waterfall method, so we made that our starting point. We began by drafting up a Project Description, an Analysis Document, followed up by designing our program carefully, and only then allowed ourselves to dive into Implementation. However, it was solely at the latter stages that we realized the flaws of this method: it is impossible for one to accurately predict how further stages of development will progress, meaning that when we got to the end of the Implementation phase, our Analysis and Design drafts had become outdated – therefore, it was necessary for us to redo them in order for our Project Report to be coherent.

Nevertheless, the step-by-step approach suggested by Waterfall was a good start. Although not all of the initial ideas came through in the end, our Analysis and Design documents were a good foundation for the product developed, and for this reason do we consider that following the Waterfall methodology was successful. Besides, redoing Analysis in a "backwards" manner was not hard at all. When all the use cases had been implemented, it was easy to describe them and draw up their activity diagrams.

On a more personal note, if we are to assess our conflict escalations by *Glasl's Nine-Stage Model Of Conflict Escalation*, we would say that we never got past stage 1: *Hardening*. This is because the threshold for stage 2 is when "one or both counterparties lose faith in solving the problem through straight and fair discussions" (flixabout.com, 2021). It never got to this point for us – our discussions were always constructive, respectful and fair, and the involved counterparties never doubted each other – only disagreed on how to approach the issue at hand.

As for how we chose to work together, the work style selected by our group was similar to the fourth one from the *"How to work together.mp4"* video in the Effective Teams learning path inside of the SEP1Y Course, *itsLearning (Wærn, 2021)*. The work style described in the video is based on working as sub-teams to complete different tasks. This way, it is possible to learn from different people and simultaneously get a lot of different work done. Our approach relied a lot on pair work. The pairs formed were often Daniel and Laura, and Dragos and Matas. Still, although we were working separately, we were all in the same room for group meetings and could ask any group member for help at any time.

6 Personal Reflection

Laura Rebelo

This project taught me a lot. During my past education, every task assigned to us was always done individually and rarely ever as a group. However, in SEP, we were forced to break out of our shell, learn to work as a team, and place our trust on others. This was a bit of a challenge for me, as I tend to like to have everything my way. In fact, this is something that I wish to improve on my next Semester Project: throughout SEP1, I sometimes worked outside of "meeting hours" in ways that made the project move forward. This alone is good - but what was not so positive was that, afterwards, I did not always make sure that I went through every detail of what I had done with the rest of my group members. This is negative in a sense that, if not everyone has knowledge of everything that has been done, further progress can be hindered. Next semester, I hope that I can be more transparent with my work in order for my group members to be able to build on top of what I have worked on, instead of me inadvertently assuming the "monopoly" of a certain aspect of the program only because I have chosen to work overtime on it.

As for motivation, I believe our group felt motivated throughout the whole project period. No one ever skipped meetings without a critical reason, and we always made great progress every time we met. Group members also frequently seeked out to do useful tasks on their own, which never failed to bring the project closer to completion.

Personally, my only "demotivator" was just the overall feeling of burning out after working on the project after two, three weeks. Nearing the end of the project period, I started often feeling tired and finding myself wishing that it would be over as soon as possible. During this stage, my motivation shifted from doing it because it was enjoyable and stimulating to doing it because there was a deadline to meet and I wanted to pass the course. In other words, it became a lot less intrinsic and a lot more extrinsic due to exhaustion.

On another note, I would say that the group contract did not play a big role in defining our team dynamics. Since all of us were motivated towards the same goal and rarely ever slacked off, there was never the need to ever bring up the contract. Still, we did live up to its contents: respected each other, helped each other and always made the project move forward to the best of each of our abilities.

After SEP, I believe more than ever that group work and problem-based learning are exceptional ways of learning. Through group work, we are able to learn from each other and teach each other – two things that are really effective at making knowledge sink in deeply and not just superficially (Fiorella and Mayer, 2021). As for problem-based learning, there is much that can be said about it. It is not easy. For someone who is used to sitting back, listening to lectures and merely doing notebook exercises, it is a big leap and a big change that one might not always be ready to handle. However, I believe that it is when we struggle that we are pressured to learn the most (Boaler, 2021), and I can confirm that amidst the struggle felt throughout this Project Period, I inherently learned a lot about teamwork, about programming and about JavaScript.

Dragos-Daniel Bonaparte

This project just proved to me that even tough I have some experience in programming, things might still get complicated and weird. The most difficult part was to create de GUI part of the program because it was a newly acquired knowledge. The content of the group contract was pretty much respected throughout the project and everybody did whatever it was necessary to achieve a common objective. Throughout the project I in cooperation with my groupmate Matas, we created all the GUI methods and everything that was needed for the GUI to display and to make everything work the way that is intended. After everything was displaying and working Laura helped us implement the logic into the GUI, we made the buttons functional and started to think about how to schedule everything based on the input from the GUI. Basically, me and Matas worked on the creation of the GUI and on the translation of the user input to data that can be processed and used to create timetables.

The group contract had a huge success because whenever we were getting stuck in a situation, maybe one of us knew what was the problem and that helped us resolve the problems on the go without wasting too much time on the problem. Everybody worked on the same thing at the same time, for example when we were designing the program everybody was giving suggestions and everybody was designing a window. The strategy of working together on the same thing worked like a charm and helped us surpass checkpoints after checkpoints as the day passed by.

I suggest that we first create a modular checklist where we all can see what it’s needed for the project and we assign a plan on how to achieve all of this. Organizing by days or in another way. For example, let’s say that we want to create the module of the program, we create together the Astah diagram and to assign to each of us a number of methods to complete until the next day. This way, everybody knows what to do and will know how to use those classes. Currently the module was primarily made by Laura and because of that, me and Matas found it difficult at some times to access some classes or variables because we didn't know a lot about them, which resulted in us needing to study the module first then start create functionality for the GUI.

Nobody worked more than others, we worked an equal number of hours and we valued our skills which resulted of us managing to move forward fast. As me and Matas had experience in C++ we started with the GUI and as Laura had experience in phyton she started with the module, Daniel helped us a lot in HTML because of his experience with using javascript and JQuery we managed to create a very responsive website that can read our XML files and display them and a very well-made table.

The thing that motivated the group most was that we wanted to prove to ourselves that we are capable of creating a desktop application that is of good quality and easy to use. The thing that demotivated us from my perspective was the lack of a task list, we knew what we did but sometimes you'll get confuse and don't know what exactly needs to be done in order to advance with the project.

I learned to be out of my comfort zone for an extended amount of time and I managed to grow a system in which I can tell to my group what bothers me about the project. I learned to communicate more and to ask for help whenever I need it. The biggest challenge I had was to get used of working with others, I am used of coding alone in my house and thinking to everything alone. At first I just couldn't get the hang of it, I was feeling uncomfortable when wanting to look for tutorials and information around the internet on how to code something more complex because of a small anxiety I have.

I will try to stand out more the next group project we have and to get more involved in the group arguments.

The advantages of group work are that you don't need to pull everything off by yourself, the amount of stress you have is smaller and way more manageable. I did a lot of problem-based learning in high school so for me personally it didn't have as much of an impact as in others, but it is good that we don't bother to much of discussing the problem, but rather start diagnosing the problem and resolving it.

The disadvantages of group work are that you don't have the same thinking when it comes to coding something which means that you need to be very adaptive when it comes to coding style. I didn't find yet a disadvantage for me in problem-based learning.

I think that by creating a problem formulation you understand more what exactly needs to be done in order to resolve that problem. The con of creating a problem formulation is that it might also confuse you at some point and often make you misunderstand the problem.

By creating a project description you're writing everything you did in coding and how you thought the application in human language that everybody can understand. I think is a very good thing because it also makes the customer aware of what exactly is happening in the background. The con of creating a project description is that it takes a lot of brain power and a lot of time to create a quality one.

Matas Armonaitis

This project was an interesting experience and really made me learn a lot of new things. I had experience in programming in my high school, but we never had projects of this scale, also I was learning c++ not java. I learned a lot of new tricks and methods how to create a program and for that I am also grateful to my group members who helped me with some things I could not understand myself.

I did not have a lot of experience working in groups before this project so a lot of what we did was new to me. I really liked that we all agreed on when the group meetings should be, even though some of us sometimes are a little late we never address that being late is a problem which made things less stressful. We all had different backgrounds in the work we did, and I would say that’s a good thing since we could arrange our work accordingly. For example, Daniel is very experienced in web design, and he did most of the website related stuff, me and Dragos had some programming experience, even though it was not a lot, it really helped in creation of the program. Laura has very good English skills and did most of the writing in the documentation and a lot of the program writing as well. Even though we arrange our work depending on the background the work was not spread out evenly, especially in the program coding period, even though we worked the same number of hours in the group meetings. I feel that for the next time we should spread the coding more evenly since it is hard to help someone or to add something to the program without a lot of knowledge of what has been already done and how it is done.

Putting all the work-related stuff aside, the group environment I feel is good. We never had arguments, we had discussions with valid arguments and always came out with one way to do things or a compromise, that way we would not have any bad feelings in the group. We do breaks usually when we feel like we need a break which makes me feel very free in while working. During breaks we either talk about random things just to get our minds off the project or go to eat. We all have a similar goal to our project and that helps us not to push each other more than we want to. Even though we are from different countries and different cultures somehow it does not feel like it separates us. We usually have music playing in the background while we work and even though we listen to different genres of music, we have some that suits us all. Music itself made the working environment feel more laid back and we all kind of “vibe” together.

This project also made me learn a lot about myself. I learned how much work I can take, learned how much more productive I am in a friendly and focused environment, learned new ways to approach a problem and probably most importantly how to cooperate with my group. I learned that talking and discussing problems helps a lot to do the work faster.

Even though we cooperated a lot during project period, for the next time I think we should distribute the work more evenly instead of some of us taking more work than the others. Because this time I do not feel as I did as much work as the others, but not because of lack of motivation, but simply that some of the work was done when not on the group meetings.

Big advantages of group work are that with more people big projects can be done very quickly, also if someone gets stuck on some part, they cannot go solve by themselves, they can always receive help.

Disadvantages of group work is that work can sometimes be confusing, especially when trying to help someone with their problem. Also, if the group does not cooperate the work can be very unfocused and without the same path towards the objective.

Advantages of problem-based learning are that when we already know the problem, we can start discussing how to solve it head on instead of wandering what the problems could be.

The Disadvantage of problem-based learning is that some deeper things can be not addressed when solving something and starting from a problem.

Problem formulation is very important for the work since when already the problems having laid out, we can start solving them head on.

Project description is essential for big project since it gives the overall description of the work that is done or going to be done, also helps to keep the people working on the project to have the same goal.

Daniel Lopes Adrião

Even though I had some past experience with programming, this project taught me a lot especially about documentation and working on the group since until the start of this project, I almost only worked alone.

One of the most difficult parts to understand was the creation of the GUI because it was so new for us and also I was used to windows forms and I was expecting to be closer to that but it turns out it's different in a lot of aspects.

One of the top things about our group was that everyone helped everyone with everything. That made it so that everyone knows every single aspect of the project and how it was done. Another thing I really enjoyed about the group was the meetings because we could find time to focus but to laugh too and I'm sure that's one of the reasons we were always motivated for the next one.

About the group contract, it was very nicely written and fair but with time we didn't use that often since everyone had the same goals and motivation to work so we just did our best and did not follow the contract line by line what turns out as a great result after all.

Putting work aside, the group itself was really wonderful, we always understood each other needs and helped in every sense, also one of my favourite things was the background music because every time there was someone playing music to keep the flow and we always made a way that everyone likes the chosen songs and sometimes sing together too, also the breaks for lunch together were always very nice so we could have some free time to talk about us and our lives outside here what made us closer than before.

Even though I can say that everything was amazing there is the only thing it could be better but I'm sure it will be "fixed" for the next project, which was the distribution of work, there was some meeting that I felt lost or didn't have anything to work on and in that times I felt a bit useless but as I said it was only in some meetings.

After the first SEP, I can say that one of the advantages of problem-based learning is that when we go through a problem we really learn with it, why it happens and how to fix it and it really makes us understand better but in the same way, there is an advantage that sometimes we get a lot of time stuck in the same problem and can't seem to be fixed, of course, it eventually is but sometimes can be stressful.

The project description is a really important step on every project because it gives not only a summary about what it's done but how it was done and after a while, it's always important to understand we took a step and how and why we did it like that.

7 Supervision

Supervision was a crucial aid for us in developing this project. Although we did not frequently have meetings with our supervisors, we contacted them through e-mail quite often – almost every working day.

Our criteria for reaching out was to do it whenever we felt stuck on something that we could not figure out on our own. This would include questions about implementation in Java or JavaScript, solving errors for which we could not find a solution online, or documentation-specific requirements. We were always satisfied with the reply received from our supervisors – they were straight to the point and exactly what we were looking for, meaning they helped us overcome the obstacles that were hindering our progress. Were it not for them, we might not have been able to reach a satisfactory working end product.

8 Conclusions

If we are to sum up what we learned from this Project, the main takeaway was related to teamwork. Most of us were not used to working in teams or pairs before coming to Denmark, and this experience broadened our horizons and made it easier for us to trust others in a work-related environment. Additionally, by being presented with a problem and being prompted to solve it autonomously without a tutorial or guide to follow, we gained deep knowledge about programming in Java and about JavaScript.

Next semester, we are inclined to stay in the same group. Knowing this, and in order for everyone in the group to have a better SEP experience in a sense that no one feels overworked nor underworked, we will try to be more organized and transparent with our tasks. This could mean splitting tasks more evenly, and this way, the workload for each group member will be evened out.

9 Sources

Boaler, J., 2021. Why Struggle Is Essential for the Brain — and Our Lives - EdSurge News. [online] EdSurge. Available at: <https://www.edsurge.com/news/2019-10-28-why-struggle-is-essential-for-the-brain-and-our-lives> [Accessed 14 December 2021].

Fiorella, L. and Mayer, R., 2021. The relative benefits of learning by teaching and teaching expectancy.

Hunschool.org. 2021. What is Problem-Based Learning (PBL) | Hun School of Princeton. [online] Available at: <https://www.hunschool.org/resources/problem-based-learning> [Accessed 14 December 2021].

Wærn, M., 2021. How to work together. [video] Available at: <https://vimeo.com/558004312/bdb1c0229e> [Accessed 15 December 2021].

Appendices

APPENDIX 1C – Detailed Logbook

Time period	What was done
Week 40	- First draft of **Project Description**
Week 41	- Feedback from Partner Group & Supervisor**: POLISHED PROJECT DESCRIPTION** - First RWD Assignment (Unresponsive Website) - Fifth DMA Assignment (Algorithm Analysis and Big O)
Week 42	HOLIDAY
Week 43	- Started **Analysis** part 1 (Requirements & Use Cases Descriptions) - Fifth DMA Assignment (Algorithm Analysis and Big O)
Week 44	- Second RWD Assignment (Responsive Website) - Sixth DMA Assignment (Binary Trees) - Feedback from Partner Group & Supervisor: **POLISHED ANALYSIS DOCUMENT**
Week 45	- Seventh DMA Assignment (Sets, maps, hashing and sorting) - Started **Analysis** part 2 (Activity Diagram & Domain Model)
Week 46	- Seventh DMA Assignment (Sets, maps, hashing and sorting) - Finished & handed in Final Analysis Document.
Week 47	- Third RWD Assignment (JQuery) - Eighth DMA Assignment (Graphs) - Presentation about design. - Designed Class Diagram.
29/11	- **START OF PROJECT PERIOD.** - Had DMA class about Graphs.
30/11	- Met from 13:00 – 17:00. - Designed GUI.
1/12	- Met from 10:00 – 15:00. - Created GUI on Java FX.
2/12	- Met from 13:00 – 16:00. - Implemented most of the Model Classes inside Java Application. - Implemented View Controllers.
3/12	- Met from 13:00 – 16:30. - Polished classes: improved their “reusability” - Implemented extra classes - Got started on persistence (loading and saving, to and from txt and xml files)
4/12	WEEKEND
5/12	WEEKEND
6/12	- Met from 13:00 – 17:00. - Improved choice boxes inside Schedule Session window. - Implemented “Schedule Session” use case. - Implemented the Search function in “Reschedule Session” window. - Minor fixes.
7/12	- Met from 11:00 – 16:00. - Implemented the Search function in “Manage Students” window.

	- Minor fixes. - Changed persistence from XML to JSON. - Website can now read XML file with information about Sessions. - Website can now display timetable imported from XML file in a responsive table. - Implemented "Cancel Session" button.
8/12	- Met from 10:00 – 15:00. - Added a Pop-up to Switch a Student's class. - Started Javadoc. - Website nearly finished. - Implemented check for Classroom overlaps.
9/12	- Met from 10:00 – 15:00. - Continued Javadoc. - Made Algorithm Time Complexity Analysis. - Implemented Switch a Student's Class. - Finished implementing Rescheduling of a Selected Session. - Fixed bug in the website where End Time of Sessions was displayed incorrectly.
10/12	- Met from 12:00 – 16:00. - Program is basically working. Only minor fixes left. - Redid Analysis document according to existing system. - Outlined what needs to be done for Documentation.
11/12	- Got started on Project Report.
12/12	- Got started on Process Report.
13/12	- Met from 13:00 – 17:00 - Made User Guide. - Made major merge in code. Code is finished. - Minor changes to the website: added arrows for navigation between weeks. - Progress on Project Report and Process Report.
14/12	- Met from 10:00 – 12:00, 14:00 – 16:00 - Made Installation Guide. - Finished Project Report. - Finished Process Report. - Organized Appendices for Hand-in.