

**IFB299 – IT Project Design and Development**  
**Queensland University of Technology**  
**Semester 1, 2018**

**Sprint 2 Personal Portfolio**



**Student:** John Santias (N9983244)

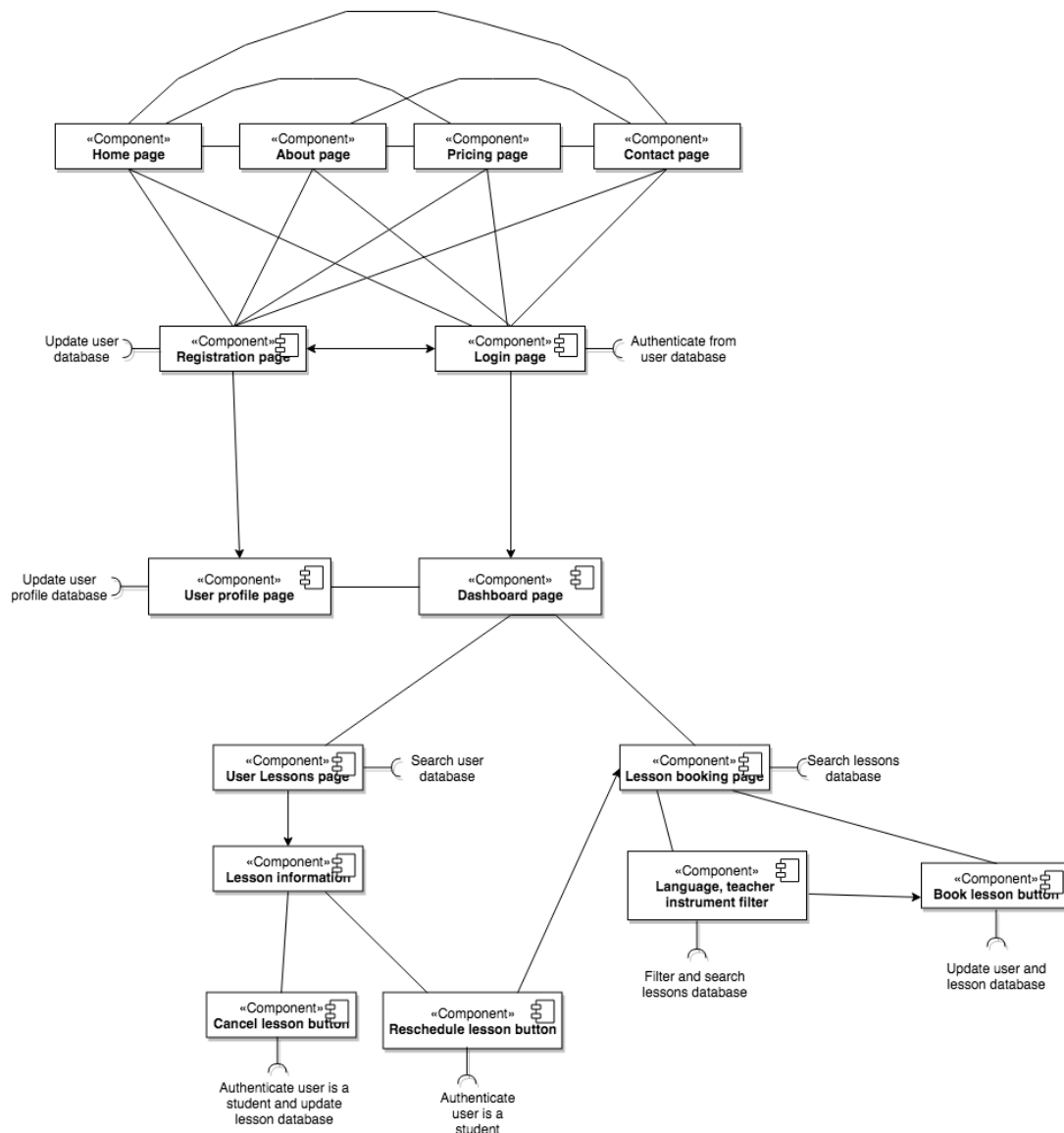
**Group:** 15 – Pink Spoon

**BitBucket:** <https://bitbucket.org/ifb299group15/>

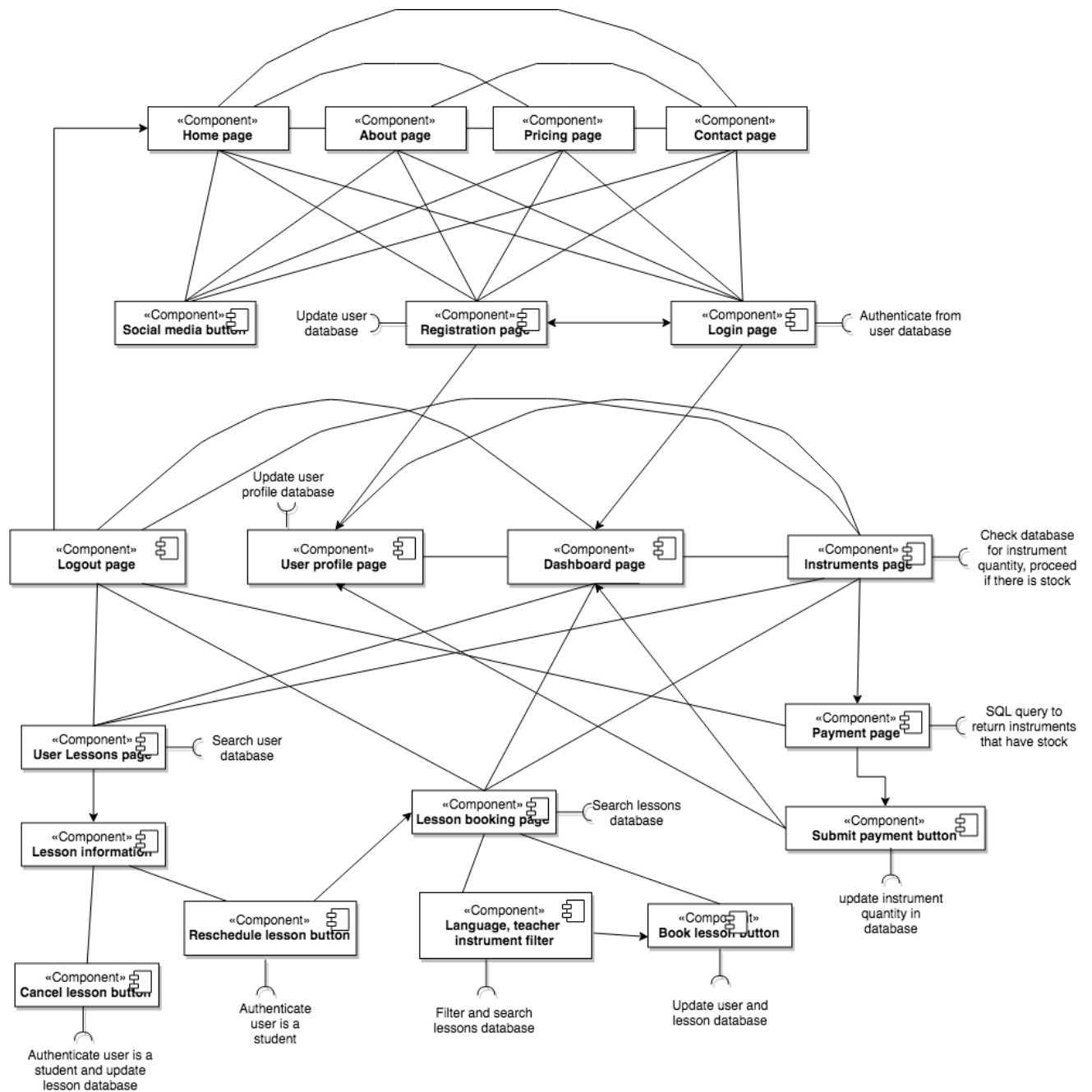
## Artefact 1 – (Component Diagram)

The component diagram describes how our whole website functions. In sprint 2, we included a couple of new components to our website. These include social media buttons, the instruments, payment page, logging out process, linking up pages, well as explaining their processes with the MySQL database. The diagram was used to guide the developing team during development of the project. For example, the instruments page had a 'hire here' button to click on to hire the instrument. So a developer had to implement a function following this diagram to redirect the user to the payment page, then update the instrument quantity after accepting the payment. The developers were able to follow this diagram making sure users did not end up getting an error message or going to an unknown page when going through the website. This diagram did help save time throughout the development preventing them from constantly contacting the product owner and changing links.

### Sprint 1 Component Diagram:



## Sprint 2 Component Diagram:



**Artefact 2** – (Action/meeting minutes)

Meeting discussions and decisions were all recorded on a document, these files are called action or meeting minutes. Scribing each team meeting was very important because it allowed the team to look back on what was discussed, which task is assigned to who, seeing who showed up, and reviewing the progression of the project. An example of a time these action minute documents were very useful was when a team member forgot what was discussed during the team meeting and couldn't remember what the product owner wanted on a particular page. The member would look through the action minutes to double check on the decisions made. The benefit of having these action/meeting minute documents certainly clears uncertainty during development and saves repeated questions for the product owner. Examples of the action minutes are on the next page.

## Action minutes of meeting #20

### Action Minutes

Author: John Santias



Meeting Date:	Meeting Time:	Meeting Place:
14/5/18	9-10pm	Discord

#### Members Present:

Member Name	Present
James Uprichard	Yes
Michael Bell	Yes
Emily-Jane Deering	Yes
John Santias	Yes

#### Decisions:

- Remove the contact teacher on the navigation bar and move it to the dashboard instead.
- Product owner wants the team make the navigation bar simpler and make ease of use better. Maybe move the logout button to the bottom of the page or try to simplify the navigation bar.
- Michael to get story 17 from release 3 and complete it.

#### Assigned tasks:

Team Member	Tasks Assigned on Meeting Date	Expected Completion Date
Emily-Jane Deering	Statistics	16 May
James Uprichard	Hiring instruments	16 May
Michael Bell	Customer review	16 May

#### Project Progress:

Project Component	Status of Component	Delivery Date
Contacting the school	Completed	7 May
Admin controls	Completed	7 May
Viewing lesson types	Complete	11 May
Cancellation/Rescheduling lessons	Complete	11 May

Contacting Teacher	Complete	14 May
Hiring instruments	Incomplete	13 May
Statistics	Incomplete	13 May
Customer review	Incomplete	14 May

There were no reported issues during this meeting.

## Action minutes of meeting #22

### Action Minutes

Author: John Santias

Meeting Date:	Meeting Time:	Meeting Place:
18/5/18	11am	Discord

### Members Present:

Member Name	Present
James Upprichard	Yes
Michael Bell	Yes
Emily-Jane Deering	No
John Santias	Yes

### Decisions:

- Each team member to start writing the retrospective.
- Continue working on tasks
- Product owner wants website to auto generate lessons within the next 7 days instead of 14 days as there are too many lessons displaying on the page.

### Assigned tasks:

Team Member	Tasks Assigned on Meeting Date	Expected Completion Date
Emily-Jane Deering	Reaching out	21 May
James Upprichard	Hiring instruments	21 May
Michael Bell	Customer review	21 May
John Santias	Hiring instruments database	21 May
James upprichard & Michael Bell	User testing	23 May

### Project Progress:

Project Component	Status of Component	Delivery Date
Contacting the school	Completed	7 May
Admin controls	Completed	7 May

Viewing lesson types	Complete	11 May
Cancellation/Rescheduling lessons	Complete	11 May
Contacting Teacher	Complete	14 May
Statistics	Complete	16 May
Hiring instruments	Incomplete	13 May
Customer review	Incomplete	18 May
Reaching out	Incomplete	21 May

There were no reported issues during this meeting.

## Action minutes of meeting #23

### Action Minutes

Author: John Santias

Meeting Date:	Meeting Time:	Meeting Place:
21/5/18	8pm	Discord

#### Members Present:

Member Name	Present
James <del>Uprichard</del>	Yes
Michael Bell	No
Emily-Jane <del>Deering</del>	Yes
John Santias	Yes

#### Decisions:

- Emily to complete user story 18 asap.
- James & Michael to get user stories done asap.
- Polish website, fix grammar/spelling errors.
- Product owner wants website to automatically fill sql database tables such as instrument quantity and teachers So she doesn't have to fill in database manually through MySQLworkbench.

#### Assigned tasks:

Team Member	Tasks Assigned on Meeting Date	Expected Completion Date
Emily-Jane <del>Deering</del>	Reaching out	21 May
James <del>uprichard</del> & Michael Bell	User testing	23 May

#### Project Progress:

Project Component	Status of Component	Delivery Date
Contacting the school	Completed	7 May
Admin controls	Completed	7 May
Viewing lesson types	Complete	11 May
Cancellation/Rescheduling lessons	Complete	11 May
Contacting Teacher	Complete	14 May
Statistics	Complete	16 May
Hiring instruments	Complete	21 May
Customer review	Complete	21 May
Reaching out	Incomplete	21 May
User Testing	Incomplete	23 May

There were no reported issues during this meeting.

### Artefact 3 – (Assigning tasks)

The group was not able to complete all of the stories in sprint 1, reflecting on the processes of the previous sprint, I was assigning tasks not matching the member's skills set and assigning two or three of a user story's tasks. Assigning tasks this way took up more time to complete. Sometimes also getting two or more team members to help out with a difficult task. There were also some members doing the same unassigned tasks causing confusion and misunderstanding in the team meetings. In sprint 2, I wanted to be more clear with the assigned tasks and get the group more organised. Learning from the past, tasks were given to group members matching their skills and experience. One user story was assigned to one member. Towards the end of team meetings, I would ask each member what their tasks were so that way I could be sure that there wouldn't be any conflict later on. The meeting minutes also benefited the group to remind their selves of the task assigned. Trello's feature of putting a member's name on a story card was another helpful tool to remind the members. It was found that giving a user story to work on as a task and matching the member's skills/experience allowed the team to complete the tasks in a timely manner compared to sprint one.

### Trello's user assignment tools:

The left screenshot shows a Trello board with a list of stories. Each story card has a green checkmark indicating completion status and a member avatar assigned to it. The stories and their assigned members are:

- S18: Reaching Out {Could} - EXTRA STORY (Assigned to E)
- S17: Customer Review {Could} - EXTRA STORY (Assigned to M)
- Story 9: Contacting teacher {Should} (2.5) (Assigned to M)
- Story 2: Hiring Instruments {Must} (2) (Assigned to JU and J)
- Story 12: Statistics {Should} (0.5) (Assigned to E)
- Story 15: Viewing lessons types {Should} (2) (Assigned to J)
- Story 14: Cancellation/rescheduling lessons {Should} (3) (Assigned to J)

The right screenshot shows a Trello story card for S18: Reaching Out {Could} - EXTRA STORY. The card has a green checkmark indicating completion status. A dropdown menu is open, showing the 'Members' section with a search bar and a list of board members. The members listed are:

- John Santias (john98273387)
- ejdeering (ejdeering) (Selected with a checkmark)
- James Uprichard (jamesuprichard)
- Michael (michael29198838)

The dropdown menu also includes options for 'Edit Labels', 'Change Members', 'Move', 'Copy', 'Change Due Date', and 'Archive'.



## Assigning tasks on the action minute documents:

Assigned tasks:

Team Member	Tasks Assigned on Meeting Date	Expected Completion Date
Emily-Jane <del>Deering</del>	Reaching out	21 May
James <del>Uprichard</del>	Hiring instruments	21 May
Michael Bell	Customer review	21 May
John Santias	Hiring instruments database	21 May
James <del>uprichard</del> & Michael Bell	User testing	23 May



- 3) Database table filled with lessons after loading the *'Bookings'* page, booked lessons aren't removed.

id	Instrument	Date	Time	Lesson_id	Language	Booked	teacher_id
3018	Guitar	2018-06-02	13:00:00.000000	Guitar13:00-2...	English	YES	6
6378	Violin	2018-05-30	10:00:00.000000	Violin10:00-20...	English	NO	1
6379	Flute	2018-05-30	10:00:00.000000	Flute10:00-20...	English	NO	2
6380	Piano	2018-05-30	10:00:00.000000	Piano10:00-20...	English	NO	3
6381	Saxophone	2018-05-30	10:00:00.000000	Saxophone10:...	English	NO	4
6382	Drums	2018-05-30	10:00:00.000000	Drums10:00-2...	English	NO	5
6383	Guitar	2018-05-30	10:00:00.000000	Guitar10:00-2...	English	NO	6
6384	Violin	2018-05-30	10:30:00.000000	Violin10:30-20...	English	NO	1
6385	Flute	2018-05-30	10:30:00.000000	Flute10:30-20...	English	NO	2
6386	Piano	2018-05-30	10:30:00.000000	Piano10:30-20...	English	NO	3
6387	Saxophone	2018-05-30	10:30:00.000000	Saxophone10:...	English	NO	4
6388	Drums	2018-05-30	10:30:00.000000	Drums10:30-2...	English	NO	5
6389	Guitar	2018-05-30	10:30:00.000000	Guitar10:30-2...	English	NO	6
6390	Violin	2018-05-30	11:00:00.000000	Violin11:00-20...	English	NO	1
6391	Flute	2018-05-30	11:00:00.000000	Flute11:00-20...	English	NO	2
6392	Piano	2018-05-30	11:00:00.000000	Piano11:00-20...	English	NO	3
6393	Saxophone	2018-05-30	11:00:00.000000	Saxophone11:...	English	NO	4
6394	Drums	2018-05-30	11:00:00.000000	Drums11:00-2...	English	NO	5
6395	Guitar	2018-05-30	11:00:00.000000	Guitar11:00-20...	English	NO	6
6396	Violin	2018-05-30	11:30:00.000000	Violin11:30-20...	English	NO	1
6397	Flute	2018-05-30	11:30:00.000000	Flute11:30-20...	English	NO	2

## views.py:

The arrays used to generate lessons.

```
#Specify the teacher id in array
teacherslist = [1, 2, 3, 4, 5, 6]

#Specify names of each teacher in an array
teacherNames = ['Mika Williams', 'Andy Garrett', 'Milly Buxton', 'David Bernal', 'Luke Holmes', 'Caleb Dixon']

#Specify the available languages
languageslist = ['English', 'German', 'Spanish', 'Italian', 'French']

#Specify the available instruments
instrumentlist = ['Violin', 'Flute', 'Piano', 'Saxophone', 'Drums', 'Guitar']

#Specify the school's open times
timelist = ['10:00', '10:30', '11:00', '11:30',
            '12:00', '12:30', '13:00', '13:30',
            '14:00', '14:30', '15:00', '15:30',
            '16:00', '16:30', '17:00', '18:00']

#Rename instruments imported from music_app.models as it will conflict with function definition names and variables
theinstruments = instrument
```

The `'bookingsPage'` function below is executed when `'Bookings'` page is requested by the user.

```
@csrf_exempt #exempts csrf token needed to display page
def bookingsPage(request):
    if (request.method == 'GET'):
        #Makes sure teachers are on the database, if not create the teachers
        for b in range(0, 6):
            addTeacher = teacherNames[b]
            roomNO = teacherslist[b]
            #add selected teacher and the room number and add/update to teachers database table
            created = teacher.objects.get_or_create(name=addTeacher, room=roomNO)

        #create array to store the dates
        dates = []

        #Gets current date
        today = date.today()

        #Gets the date after 7 days of variable 'today'
        end_date = today + timedelta(7)
        bookedString = 'NO'

        #Delete all lessons in schedule table that haven't been booked
        #Gets rid of old dates
        schedule.objects.filter(Booked="NO").delete()

        #puts dates in 'date' array
        for y in range(0, 7):
            dates.append(today + timedelta(y))

        #add lessons
        for i in range(0, 7): #next 7 days, 1 to 7
            dateChosen = dates[i]

            #creates lessons and stores in schedule database table
            for x in range(0, 16): #lesson times, 10am to 6pm, 1 to 16 lessons
                timeChosen = timelist[x]
                for x in range(0, 6):
                    instrumentChosen = instrumentlist[x]
                    teacherChosen = teacherslist[x]
                    languageChosen = 'English'

                    #Add all retrieve values and store in the schedule database table
                    created = schedule.objects.get_or_create(Instrument=instrumentChosen,
                                                                Date=dateChosen,
                                                                Time=timeChosen,
                                                                Lesson_id=instrumentChosen + timeChosen + '-' + str(dateChosen),
                                                                Booked=bookedString,
                                                                teacher_id=teacherChosen,
                                                                Language=languageChosen)

            #Get all lessons that haven't been booked and display on the page
            data_list = schedule.objects.filter(Booked="NO")
            context = {'data_list':data_list }
            return render(request, 'music_app/bookings.html', context)
    else:
```

Similarly, for the instruments page, I implemented a function to automatically fill up the 'music\_app\_instruments' database table. This function runs when a user clicks on the 'instruments' page. If there is data inside the table, then the code will not insert anything new. Otherwise, it will use the values inside the 'instrumentslists' array (see the first picture in artefact 4) to insert data in the table with a starting amount of 100.

The 'music\_app\_instruments' database table before and after executing the 'instrumentsPage' function when there is no data.

id	instrument_name	quantity
NULL	NULL	NULL



id	instrument_name	quantity
1	Violin	100
2	Flute	100
3	Piano	100
4	Saxophone	100
5	Drums	100
6	Guitar	100
NULL	NULL	NULL

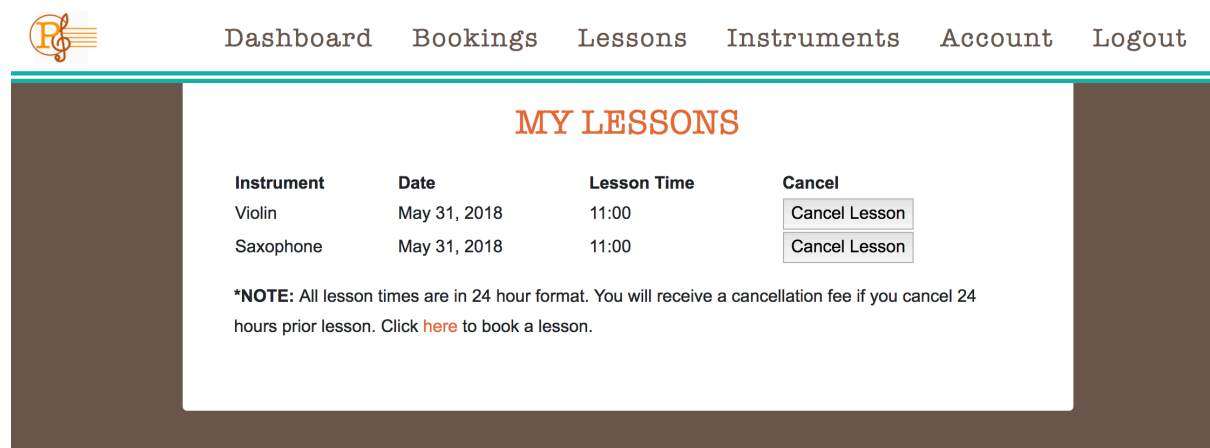
**views.py:** When the user clicks 'instruments' page, the 'instrumentsPage' function executes.

```
#Return the instrument html page
def instrumentPage(request):
    if (request.method == 'GET'):
        #Specifies the amount of instruments made to be available
        amount = 100
        for x in range(0, 6):
            #Check if instrument exists in database table 'instruments'
            if theinstruments.objects.filter(instrument_name=instrumentlist[x]).exists():
                continue
            #If instrument doesn't exist, add instrument to the database and add starting quantity
            else:
                created = theinstruments.objects.get_or_create(instrument_name=instrumentlist[x], quantity=amount)
        return render(request, 'music_app/instrument.html')
```

## Artefact 5 – (Cancelling/Rescheduling lessons)

The 'My Lessons' page allows students to view and manage their booked lessons. This page displays a list of all the lessons booked with the choice of cancelling it. When clicking the cancellation button with confirmation, the booked lesson will immediately be removed using the 'schedule\_id' to delete the row inside the 'music\_app\_bookings' table. The lesson is also updated in the 'music\_app\_schedule' table, changing the booked status from 'YES' to 'NO'. The cancelled lesson becomes available for other students. This system is important because students may not always be available for the lessons, something else important may come up on their own schedule. The cancellation system is a major part of the website because no one has to call in or see someone for lessons. I was able to extract the user's booked lessons from the database table and display it on the page inside a table format. Javascript was also used to confirm the user's cancellation request. Once the cancellation was confirmed, the lesson disappears from the page.

1) User's booked lessons. The current user has the student id of 1.



**MY LESSONS**

Instrument	Date	Lesson Time	Cancel
Violin	May 31, 2018	11:00	<button>Cancel Lesson</button>
Saxophone	May 31, 2018	11:00	<button>Cancel Lesson</button>

**\*NOTE:** All lesson times are in 24 hour format. You will receive a cancellation fee if you cancel 24 hours prior lesson. Click [here](#) to book a lesson.

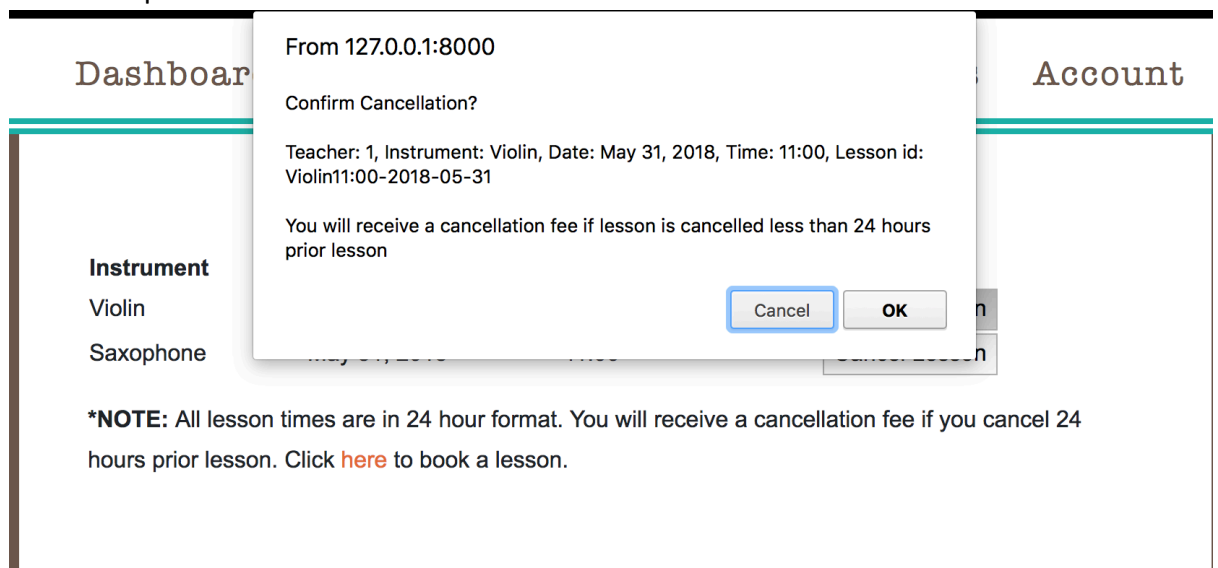
The 'music\_app\_bookings' table.

id	schedule_id	student_id
1	3018	2
3	9715	1
4	11734	1
NULL	NULL	NULL

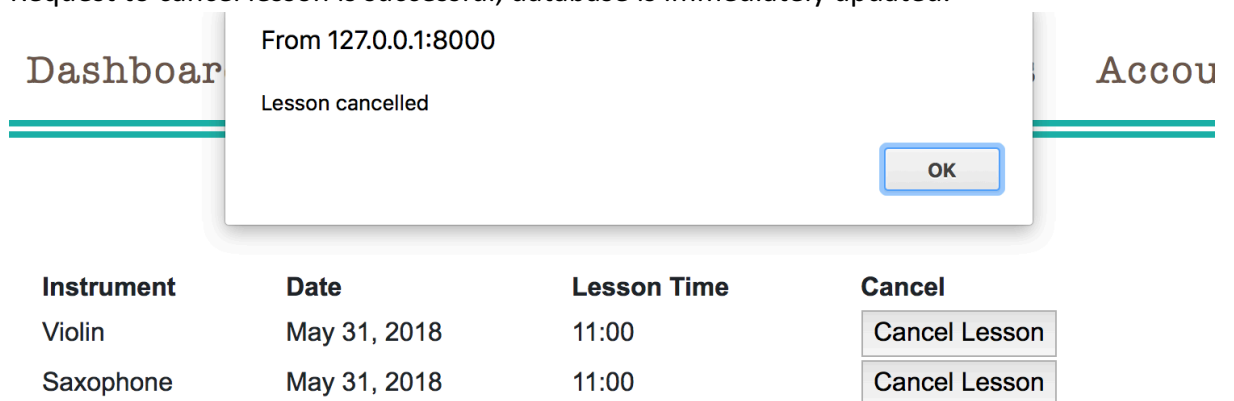
The 'music\_app\_schedule' table.

id	Instrument	Date	Time	Lesson_id	Language	Booked	teacher_id
3018	Guitar	2018-06-02	13:00:00.000000	Guitar13:00-2...	English	YES	6
9715	Violin	2018-05-31	11:00:00.000000	Violin11:00-20...	English	YES	1
11734	Saxophone	2018-05-31	11:00:00.000000	Saxophone11:...	English	YES	4
12391	Violin	2018-05-31	10:00:00.000000	Violin10:00-20...	English	NO	1
12392	Flute	2018-05-31	10:00:00.000000	Flute10:00-20...	English	NO	2
12393	Piano	2018-05-31	10:00:00.000000	Piano10:00-20...	English	NO	3
12394	Saxophone	2018-05-31	10:00:00.000000	Saxophone10:...	English	NO	4
12395	Drums	2018-05-31	10:00:00.000000	Drums10:00-2...	English	NO	5
12396	Guitar	2018-05-31	10:00:00.000000	Guitar10:00-2...	English	NO	6
12397	Violin	2018-05-31	10:30:00.000000	Violin10:30-20...	English	NO	1
12398	Flute	2018-05-31	10:30:00.000000	Flute10:30-20...	English	NO	2
12399	Piano	2018-05-31	10:30:00.000000	Piano10:30-20...	English	NO	3
12400	Saxophone	2018-05-31	10:30:00.000000	Saxophone10:...	English	NO	4
12401	Drums	2018-05-31	10:30:00.000000	Drums10:30-2...	English	NO	5
12402	Guitar	2018-05-31	10:30:00.000000	Guitar10:30-2...	English	NO	6
12403	Violin	2018-05-31	11:00:00.000000	Violin11:00-20...	English	NO	1
12404	Flute	2018-05-31	11:00:00.000000	Flute11:00-20...	English	NO	2
12405	Piano	2018-05-31	11:00:00.000000	Piano11:00-20...	English	NO	3
12406	Saxophone	2018-05-31	11:00:00.000000	Saxophone11:...	English	NO	4
12407	Drums	2018-05-31	11:00:00.000000	Drums11:00-2...	English	NO	5
12408	Guitar	2018-05-31	11:00:00.000000	Guitar11:00-20...	English	NO	6

- 2) Confirmation window pops up when the user clicks on the 'Cancel Lesson' button. Customer presses 'OK'.



- 3) Request to cancel lesson is successful, database is immediately updated.



Row removed from the 'music\_app\_bookings' table using the schedule's id.

id	schedule_id	student_id
1	3018	2
4	11734	1
NULL	NULL	NULL

Schedule id 9715 becomes available for other students. Updating the 'Booked' status from 'YES' to 'NO'.

id	Instrument	Date	Time	Lesson_id	Language	Booked	teacher_id
3018	Guitar	2018-06-02	13:00:00.000000	Guitar13:00-2...	English	YES	6
9715	Violin	2018-05-31	11:00:00.000000	Violin11:00-20...	English	NO	1
11734	Saxophone	2018-05-31	11:00:00.000000	Saxophone11:...	English	YES	4
12391	Violin	2018-05-31	10:00:00.000000	Violin10:00-20...	English	NO	1
12392	Flute	2018-05-31	10:00:00.000000	Flute10:00-20...	English	NO	2
12393	Piano	2018-05-31	10:00:00.000000	Piano10:00-20...	English	NO	3
12394	Saxophone	2018-05-31	10:00:00.000000	Saxophone10:...	English	NO	4
12395	Drums	2018-05-31	10:00:00.000000	Drums10:00-2...	English	NO	5
12396	Guitar	2018-05-31	10:00:00.000000	Guitar10:00-2...	English	NO	6
12397	Violin	2018-05-31	10:30:00.000000	Violin10:30-20...	English	NO	1
12398	Flute	2018-05-31	10:30:00.000000	Flute10:30-20...	English	NO	2
12399	Piano	2018-05-31	10:30:00.000000	Piano10:30-20...	English	NO	3
12400	Saxophone	2018-05-31	10:30:00.000000	Saxophone10:...	English	NO	4
12401	Drums	2018-05-31	10:30:00.000000	Drums10:30-2...	English	NO	5
12402	Guitar	2018-05-31	10:30:00.000000	Guitar10:30-2...	English	NO	6
12403	Violin	2018-05-31	11:00:00.000000	Violin11:00-20...	English	NO	1
12404	Flute	2018-05-31	11:00:00.000000	Flute11:00-20...	English	NO	2
12405	Piano	2018-05-31	11:00:00.000000	Piano11:00-20...	English	NO	3
12406	Saxophone	2018-05-31	11:00:00.000000	Saxophone11:...	English	NO	4
12407	Drums	2018-05-31	11:00:00.000000	Drums11:00-2...	English	NO	5
12408	Guitar	2018-05-31	11:00:00.000000	Guitar11:00-20...	English	NO	6

## lessons.html:

Javascript implemented in the html file to pass the button's id to the lessonsPage function in views.py. This id is used to update the database.

```

</script>
<table style="width: 100%">
  <tr>
    <th>Instrument</th>
    <th>Date</th>
    <th>Lesson Time</th>
    <th>Cancel</th>
  </tr>
  {% for item in user_bookings %} /* gets each row data from the bookings table */
  <tr>
    <td>{{item.schedule.Instrument}}</td> /* gets current row's instrument column and displays it */
    <td>{{item.schedule.Date}}</td> /* gets current row's date column and displays it */
    <td>{{item.schedule.Time|time:"H:i"}}</td>
    <!-- Create button's id and give it the value of the the row data to be displayed for confirmation in javascript alert -->
    <td><button id="{{item.schedule.id}}" value="Teacher: {{item.schedule.teacher_id}}, Instrument: {{item.schedule.Instrument}}, Date: {{item.schedule.Date}}, Time: {{item.schedule.Time|time:"H:i"}}", Lesson id: {{item.schedule.Lesson_id}}"onClick="clicked(this.value, this.id)">Cancel Lesson</button></td>
  </tr>
  {% endfor %}
</table>
<p></p>
<span>
<p><strong>NOTE:</strong> All lesson times are in 24 hour format. You will receive a cancellation fee if you cancel 24 hours prior lesson. Click <a href="{% url 'bookings' %}">here</a> to book a lesson.</p>
</p>
</span>
<script type="text/javascript">
function clicked(value, id) { //Gets value and id of clicked button
  if (confirm('Confirm Cancellation? \n \n' + value + '\n\nYou will receive a cancellation fee if lesson is cancelled less than 24 hours prior lesson')) {
    $.ajax({type: 'POST', //sets request type as 'POST' when calls lessonsPage function in views.py
      url: '/music_app/lessons/', //Calls the lessonsPage function in views.py choosing the else if statement 'POST'
      data: {'id': id}, //Passing id to the lessonsPage function
      success: function(resp) {
        if(resp.status == 'ok') { //lessonsPage function respond to request, if all values given are ok, and has updated database
          alert('Lesson cancelled');
        }
        else {
          alert(resp.message); //display error message saying that the id doesn't match any database record
        }
      }
    });
    location.reload(); //Reload page after cancellation
  } else {
    return false; //User doesn't confirm cancellation, don't do anything
  }
}
</script>

```



**views.py:**

The *'lessonsPage'* function executed when lessons web page is called. When customer requests to cancel a lesson, the *'else'* statement is run because the request method is *'POST'* (see 3<sup>rd</sup> line of the *'clicked'* function of *lessons.html*).

```
#displays user's booked lessons
@csrf_exempt #exempts csrf token needed to display page
def lessonsPage(request):
    if (request.method == 'GET'):
        user_bookings = Bookings.objects.filter(student_id=request.user.id).select_related('schedule'); #Get user's bookings using the current user's id
        return render(request, 'music_app/lessons.html', {'user_bookings':user_bookings}) #render html page & retrieved data and display for user
    else:
        try:
            schedule_id = int(request.POST.get('id')) #get selected lesson for cancellation
            instance = Bookings.objects.filter(schedule_id=schedule_id) #finds lesson booked
            instance.delete() #removes booked lesson from the bookings table
            schedule.objects.filter(id=schedule_id).update(Booked='NO') #finds lesson in the schedule table and updates it from 'YES' to 'NO'
            return JsonResponse({'status': 'ok'})
        except schedule.DoesNotExist:
            return JsonResponse({'status':'error', 'message': 'Schedule does not exists'}) #display error is obtained id is wrong
        except ValueError:
            import traceback
            traceback.print_exc()
            return JsonResponse({'status':'error', 'message': 'Invalid shcedule id'}) #display error is obtained id is wrong
```