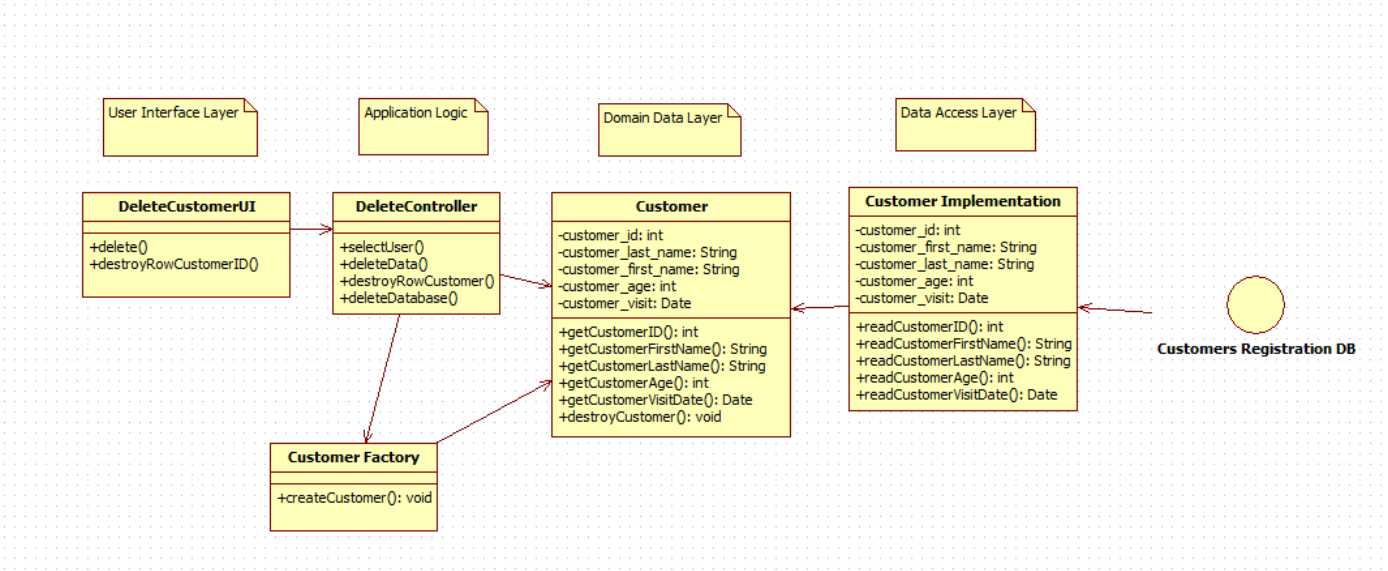
Individual Report – 260CT TASK 1

Student Name: Venelin Danielov Dimitrov	SID: 6297262
Task 1: To complete design specification, implementation, testing and integration – evidence with a report supported by research with references in the CU Harvard referencing style in a maximum of 1,000 words	

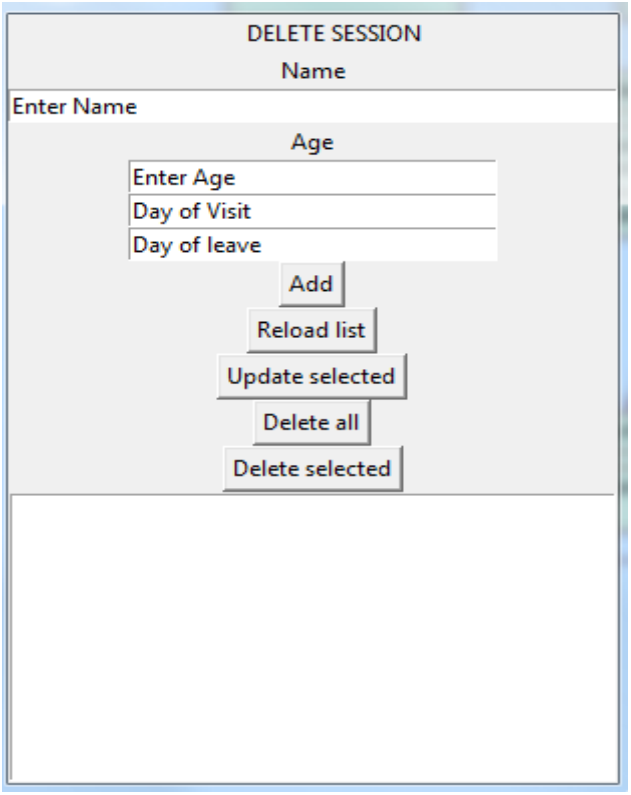


The group task was split into 7 parts with each person having their own function to develop fully with a diagram and documentation. I had the part to create a function to “Delete sessions” and “Delete Customers” from the Database. In the graph above is visible how I represented my part of the project and how it was planned to work. As a group we decided to use python and one of its pip libraries “tkinter” which had GUI (graphical user interface) integrated within its functions. And as everyone from our group had used it for our projects in the first year we all had some knowledge about its implementation. The project is done offline in a local database so that’s why i decided to make my graph as a four layer architecture.

I have decided to represent the graph with the use case controller for the GRASP method including two factories to follow the control pattern – Customer and Delete Session. This way one of the functions represents one use case and controls the functionality. The class Customer is responsible for the functionality of the whole table which puts it as a façade controller. Its functions are to create customer’s application first name, last name, id, age and day of the visit and also if there is some change with the application process to delete selected row or delete the whole session in the process window. Delete Session deletes all the data collected in the database or just deletes the selected user’s data (cancel). All the functions are done instantly in the database file and if anything needs to be changed they can be updated per entry line or by whole.

As a gang of four pattern I have used the

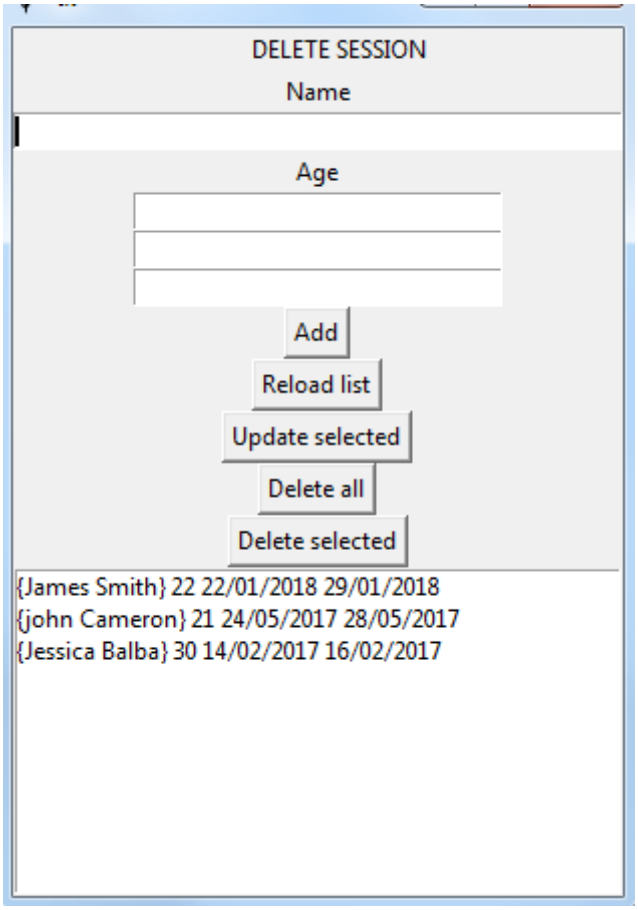
Examples:



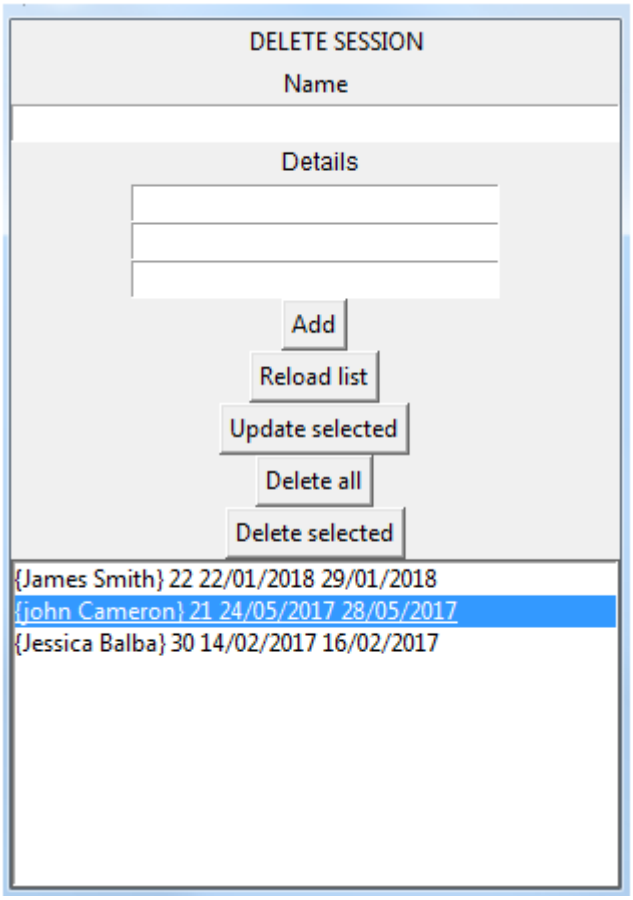
This is the main window of the application.

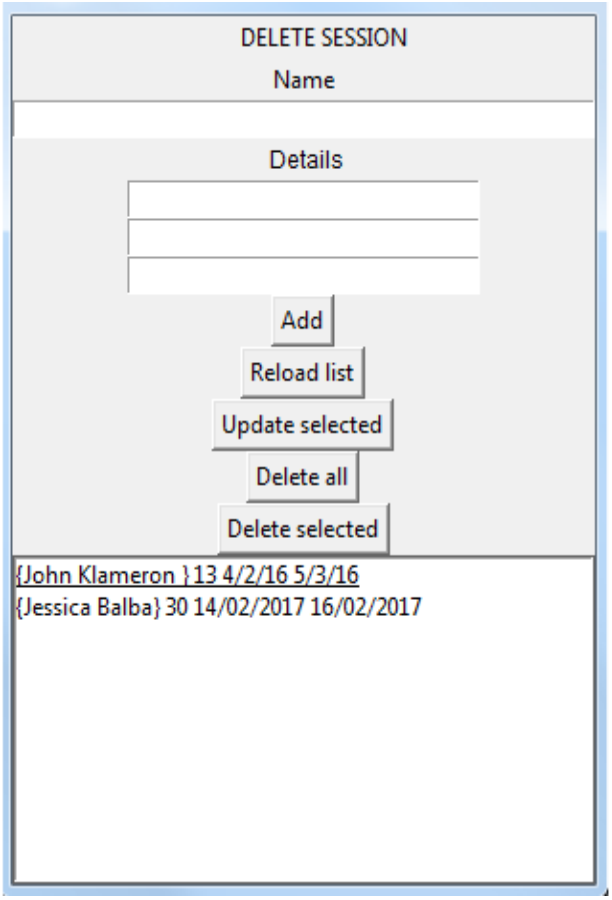
To add Make booking:
1. Operator uses the LOOKUP_CUSTOMER to find the customer from the database and then select the “Add” operation.
2. System displays all the names corresponding to that input.
3. Operator Inputs the necessary information for the booking (Day of Visit and Day of Leave, names or Age if needed)
4. The data is then saved to the system and can be transferred from the database to other operations.

To add a Customer:
1. Starting the program this comes to a window with a data that can be input into the database.
2. System displays name of the entry graphs per row and what information is needed from each.
3. Inputting all the information saves it automatically to the database file on the system.

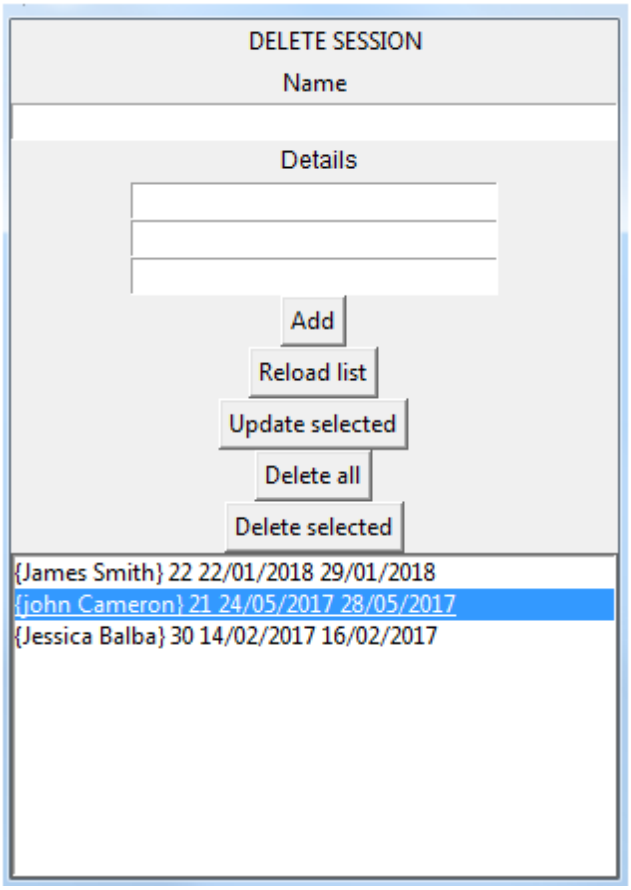


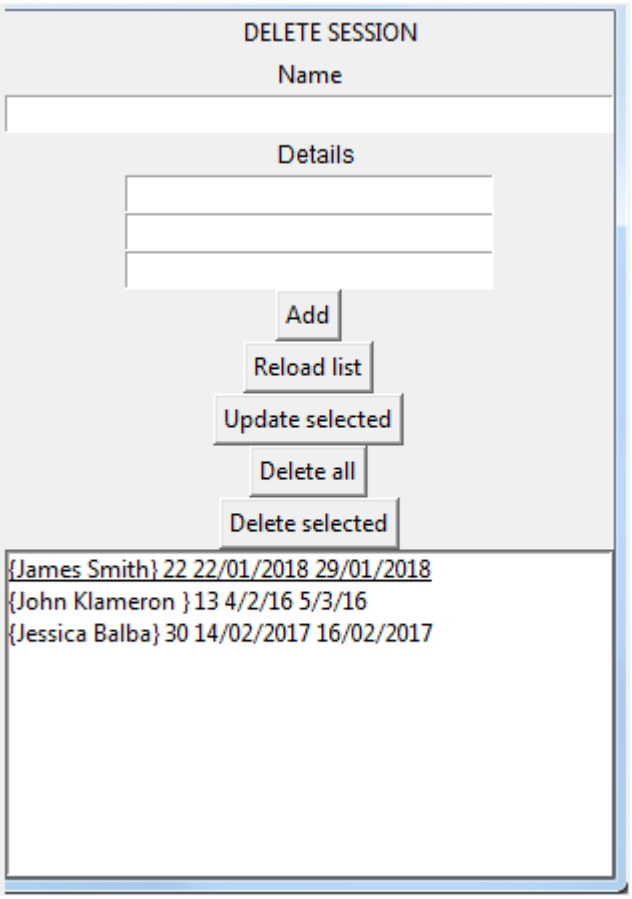
To delete a Customer:
1. Look up the information (age, name or day of visit) of the customer or select it from the rows seen in the database.
2. To delete:
a) If only selected rows need to be deleted – Select Row then “Delete Selected”
b) If the whole information needs to be deleted – “Delete All”





To update information of customer:
1. Look up the information of a customer.
2. Update selected rows with the new information
3. To save the changes – “Update Selected”





To reload the changes to the database (if others are using the application at the same time)
1. Enter the main window
2. Select “reload list”

Source Code:

```
import sqlite3 as lite
from tkinter import *
import tkinter.ttk as ttk

class CustomerImplementation:
    connection = None
    c = None
    def __init__(self, master):
        # construct of the main window
        self.text = Label(master, text="Delete Session")
        self.text.pack()
        self.text["text"] = "       DELETE SESSION AND BOOKING"

        name_label = Label(master, text = "Name")
        name_label.pack()

        self.nameField = Entry(master, text = "Name", width=50)
        self.nameField.insert(0, "Enter Name")
        self.nameField.pack()

        name_label = Label(master, text = "Age")
        name_label.pack()

        self.ageField = Entry(master, text = "age", width=30)
        self.ageField.insert(0, "Enter Age")
        self.ageField.pack()


        self.dayOfVisit = Entry(master, text = "Day of Visit", width=30)
        self.dayOfVisit.insert(0, "Day of Visit")
        self.dayOfVisit.pack()

        self.dayOfLeave = Entry(master, text = "Day of leave", width=30)
        self.dayOfLeave.insert(0, "Day of leave")
        self.dayOfLeave.pack()

        self.btn=Button(master, text='Add', command=self.add_customer_info)
        self.btn.pack()

        self.reloadbtn = Button(master, text='Reload list',
command=self.reload_list)
        self.reloadbtn.pack()

        self.showbtn = Button(master, text='Update selected',
command=self.update_selected)
        self.showbtn.pack()

        self.delbtn = Button(master, text='Delete all', command=self.del_all_notes)
        self.delSelectedbtn = Button(master, text='Delete selected',
command=self.del_selected)

        self.delbtn.pack()
        self.delSelectedbtn.pack()

        self.content=Listbox(master, width=50)
        self.content.pack()
```

```
        # open database
        self.connect_db(db_name = 'register.db')
        self.initial_listBox()

    def connect_db(self, db_name):
        self.conn = lite.connect(db_name)
        self.c = self.conn.cursor()
        # create table
        self.c.execute('''CREATE TABLE IF NOT EXISTS people(name TEXT primary key,
age TEXT, dayofVisit TEXT, dayOfLeave TEXT)''')
        self.conn.commit()

    def initial_listBox(self):
        # read people
        c = self.conn.cursor()
        people = c.execute("SELECT * FROM people")
        self.conn.commit()

        # add to list
        for person in people:
            self.content.insert(END, person)
        self.c.close()

    def reload_list(self):
        self.content.delete(0,END)
        self.initial_listBox()

    def clearNameField(self, event):
        self.nameField.delete(0,END)

    def clearAgeField(self, event):
        self.ageField.delete(0,END)

    def cleardayOfVisit(self, event):
        self.dayOfVisit.delete(0,END)

    def clearFbField(self, event):
        self.fbField.delete(0,END)

    def add_customer_info(self):
        if self.nameField.get() == "":
            self.text["text"] = "Please type something"
        else:
            name = self.nameField.get()
            age = self.ageField.get()
            dayOfVisit = self.dayOfVisit.get()
            dayOfLeave = self.dayOfLeave.get()
            self.nameField.delete(0, END)
            self.ageField.delete(0, END)
            self.dayOfVisit.delete(0, END)
            self.dayOfLeave.delete(0, END)

            c = self.conn.cursor()

            c.execute("INSERT INTO people VALUES (?, ?, ?, ?)", (name, age,
dayOfVisit, dayOfLeave))
            self.conn.commit()
            c.close()

            # add to list
            self.content.insert(END, (name, age, dayOfVisit, dayOfLeave))

    def update_selected(self):
```

```
        person = self.content.get(ACTIVE)
        name_search, age_search, dayOfVisit_search, dayOfLeave_search =
self.content.get(ACTIVE)
        if self.nameField.get() == "":
            self.text["text"] = "Please type something"
        else:
            name = self.nameField.get()
            age = self.ageField.get()
            self.nameField.delete(0, END)
            self.ageField.delete(0, END)

        # delete in database
        c = self.conn.cursor()
        c.execute("UPDATE people SET name = ? ,age = ?, WHERE name= ?", (name, age,
name_search))
        self.conn.commit()
        c.close()
        self.reload_list()

    def del_all_notes(self):
        # get selected person
        c = self.conn.cursor()

        people = self.content.get(0, END)
        if len(people):
            for name,age,dayOfVisit,dayOfLeave in people:
                # delete all from database
                c.execute("DELETE FROM people WHERE  name=? and age=? and
dayOfVisit=? and dayOfLeave=?", (name, age, dayOfVisit, dayOfLeave))
                self.conn.commit()
        c.close()

        # delete on list
        self.content.delete(0,END)

    def del_selected(self):
        # get selected person
        person = self.content.get(ACTIVE)
        name, age, dayOfVisit, dayOfLeave = self.content.get(ACTIVE)
        # delete in database
        c = self.conn.cursor()
        c.execute("DELETE FROM people WHERE name=? and age=? and dayOfVisit=? and
dayOfLeave=?", (name, age, dayOfVisit, dayOfLeave))
        self.conn.commit()
        c.close()


        # delete on list
        self.content.delete(ANCHOR)

root = Tk()
CustomerImplementation(root)
root.mainloop()
```

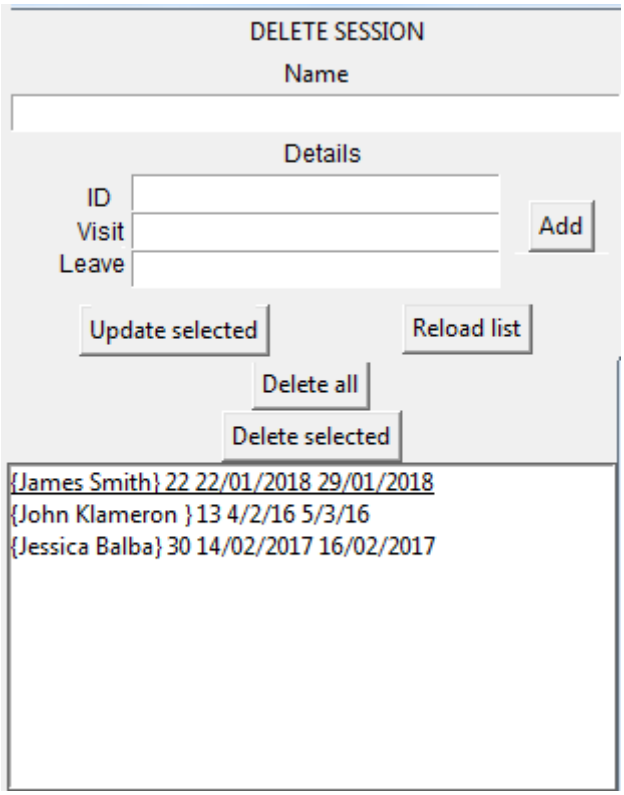
As a gang of four I have used the Customer Factory, which creates the customer and is working closely with the controller

to delete and create Customer.

PROTOTYPE TESTING:

For the first prototype which was at the very first month after the start of the project I had the single “delete selected” and “Delete All” from the database. As it completed my function. For it to be tested the client gave me the information that it needs fields to input the data so it can be deleted. Also that if changes happen to the database while in this window they won’t appear and a refresh button was needed to update the database for when the project is needed a new addition or change to existing members.

To change something before its deleted a button “Change Selected” was added as it was not seen in the other prototypes in the group. If something was in need of a small change just a single selection of entry and writing down the change and submitting will update that to the database and wont need to be deleted and then re-entry to the database. Also in case of the database to be way bigger with thousands of entries there should be a search box to find a specific user from ID or name. And a drop down menu to search for the date and the leave date of the customer in case of queries.



After second Prototype.
The client also had some specific problems with usability and the visual design of the window. As wanted from the first prototype all the data was input to the program and has satisfied the outcome of the project part of deletion of the customer but a few things could be fixed if more time was available and if it needed to be in a professional shape. Field data should be on the sides and not inside the entry boxes for the testing and searching in the database. All the buttons could be arranged. Drop Down menu for the search of the visit dates and feasibility. Main function was deleting customer so tried to not go too far from it and let my team mates develop their parts. All the parts wanted from the first prototype were met and I am quite happy for my individual part as I have completed my task in quite fast matter and had it implemented with my group mates. In the very first quarter after the start of developing of the group.

Task 2: To evaluate the management of the project – evidence with a report supported by research with references in the CU Harvard referencing style in a maximum of 1,000 words

Record of group meetings

Date	What happened at meeting
Week 1 – Tuesday 24th and Thursday 26th January 2017	In these sessions, we formed our group, read through the brief and discussed with each other what we had to as a group. Furthermore, we decided how we would achieve our goal and as a group we agreed the GUI had to be a simple one so we felt we should use the tkinter module in python as we could

	use it with sql and we all had experience coding with it.
Week 2 – Tuesday 31st January and Thursday 2nd February 2017	After discussing our proposed solution with the client, we were given the all clear to start the project and to do this we assigned all group members with a separate functionality that they had to create a prototype for e.g booking or adding a customer. Once everyone was assigned a functionality it then became a case of identifying the classes and attributes within the system which we did as a group so then we could all create our own separate diagrams. Finally, we made a Gantt chart for the project so we could track progress and plan our project accordingly.
Week 3 – Tuesday 7th and Thursday 9th February 2017	This week it was time to start the creation of our class diagrams that would be used for our system design. To create our class diagram, we used StarUML and each member of the team was assigned the role of creating a class diagram for their functionality based on the four-layer design architecture as well as including a gang of four design pattern.
Week 4 – Tuesday 14th and Thursday 16th February 2017	In this week, we continued with our class diagrams so we were sure of our design before creating the first prototypes of our functionality. Also during this week, we designed and created the database for our ski slope booking system. We needed a database set up before creating the prototypes so we could prove our prototypes worked.
Week 5 – Tuesday 21st and Thursday 23rd February 2017	During this week, we started the coding process to create our prototypes for our own functionalities. This week was a lot of individual work as the combination of programs would come later in the process. Also, this week involve a lot of debugging of our prototypes in order to show the client a working version.
Week 6 – Tuesday 27th February and Thursday 2nd March 2017	During the meetings this week we communicated and discussed our individual prototypes and shared information about things such as how to input the values into the database and how to get the GUI to work properly. In the second meeting of the week we showed our individual functionalities to our client to receive feedback so we could improve or GUIs for the final version.

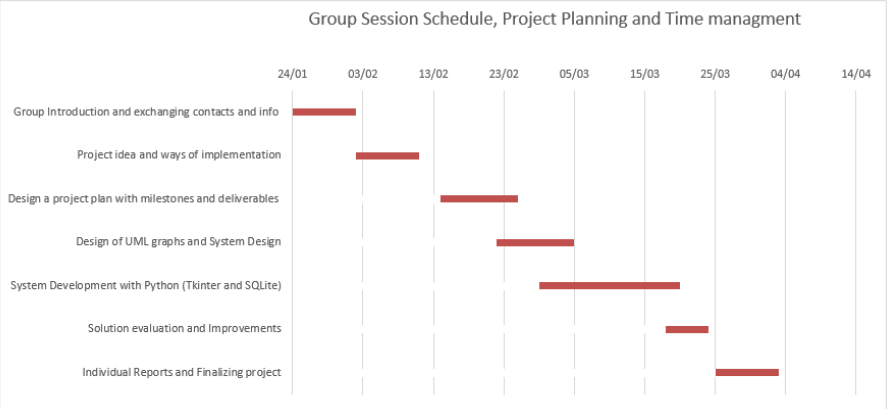
Week 7 – Tuesday 7th and Thursday 9th March 2017	This week was all about improving our individual functionalities based on the client's feedback. Once again we worked as individuals but also came together to assist each other in debugging etc.
Week 8 – Tuesday 14th and Thursday 16th March 2017	This week our group once again showed our programs to the client so we could be sure that it was what they wanted and if it meets their requirements. Overall the client was happy with our work which meant that we were ready to start combining to create a prototype of the full system.
Week 9 – Tuesday 21st and Thursday 23rd March 2017	In the first meeting of this week we combined our separate functionalities, to do this we paired up and created prototypes that featured two functionalities closely related for example adding a session and adding a booking. Upon creating these prototypes, we demonstrated their capabilities to the client.
Week 10 – Tuesday 28th and Thursday 30th March 2017	During our meetings this week we simply discussed our submission for this project and tied up any loose ends such as making sure everyone's class diagrams were in order and the code worked using automated unit testing to test our code.
Week 11 – Tuesday 3rd and Thursday 6th April 2017	This week we collected all our work together and wrote up our individual reports to submit on the 6th.

Each session was a two-hour meeting where the team discussed the project and had the opportunity to discuss the project with the client.
Tuesday – 9am to 11am
Thursday – 4pm to 6pm

The meetings consisted of code correlation, implementing code between people to look how it interacts and get informed about the changes that need to be done in the group tasks for the next stage of the program and implementing our codes.

The feedback from the client for our project was that it was done in time and it consisted with all the needed elements for the task. The separate elements have been done in different methods and with separate database implementations which was a big trouble with the group prototype implementation as everything looked one to another. The data was collected and with some improvements to the functionality we had a fully functioning offline database editor for the ski slope. That was just the finalising product problems, there were some errors in separating tasks to one another in the beginning as some of us have been attending events and interviews our group consisted of 2-3 people and communicating through chat didn't turn up as useful.

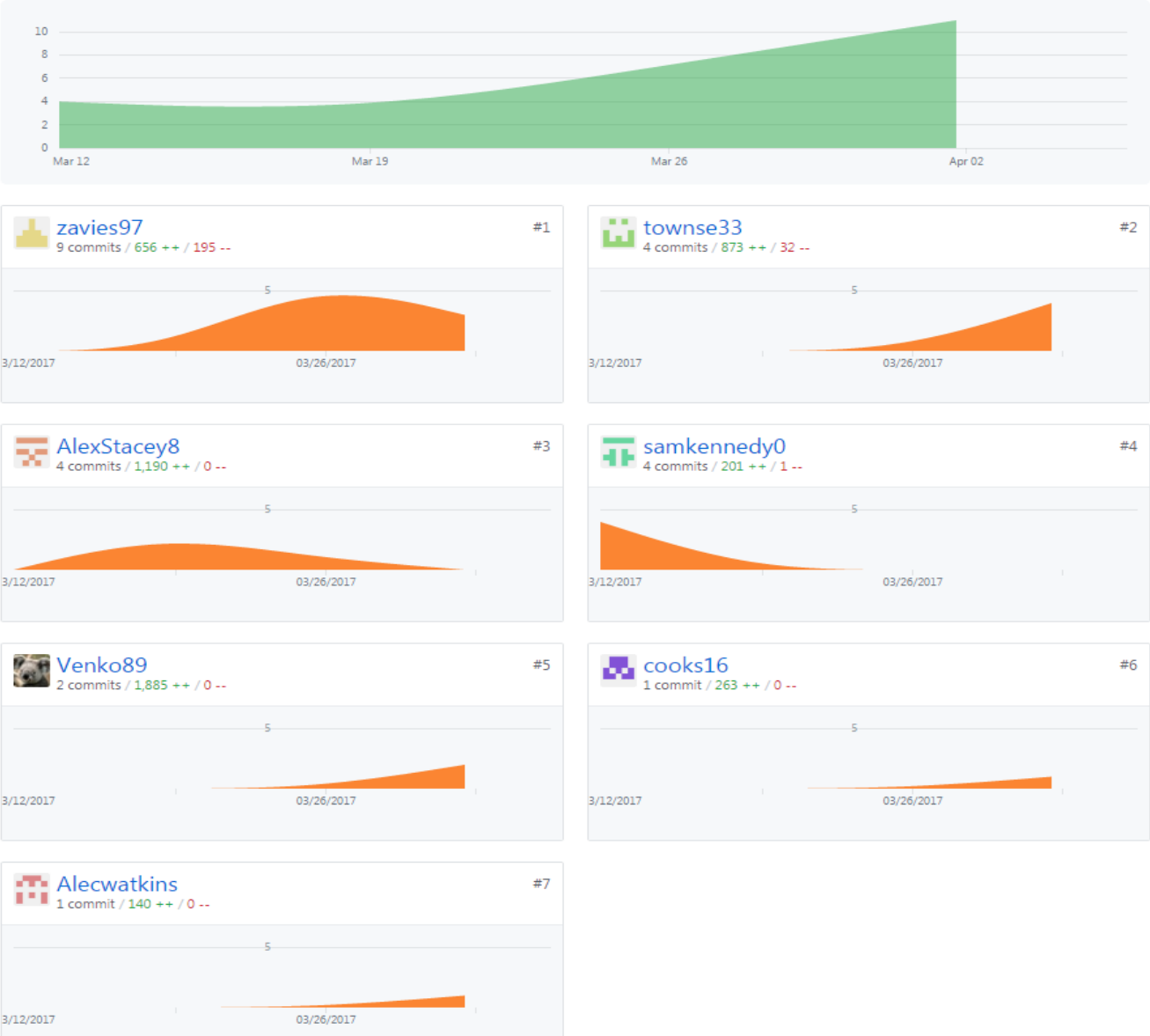
Task Name	Start	End	Duration (days)
Group Introduction and exchanging contacts and	24/01/2017	02/02/2017	9
Project idea and ways of implementation	02/02/2017	11/02/2017	9
Design a project plan with milestones and deliverables	14/02/2017	25/02/2017	11
Design of UML graphs and System Design	22/02/2017	05/03/2017	11
System Development	28/02/2017	20/03/2017	20
Solution evaluation	18/03/2017	24/03/2017	6
Individual Reports	25/03/2017	03/04/2017	9



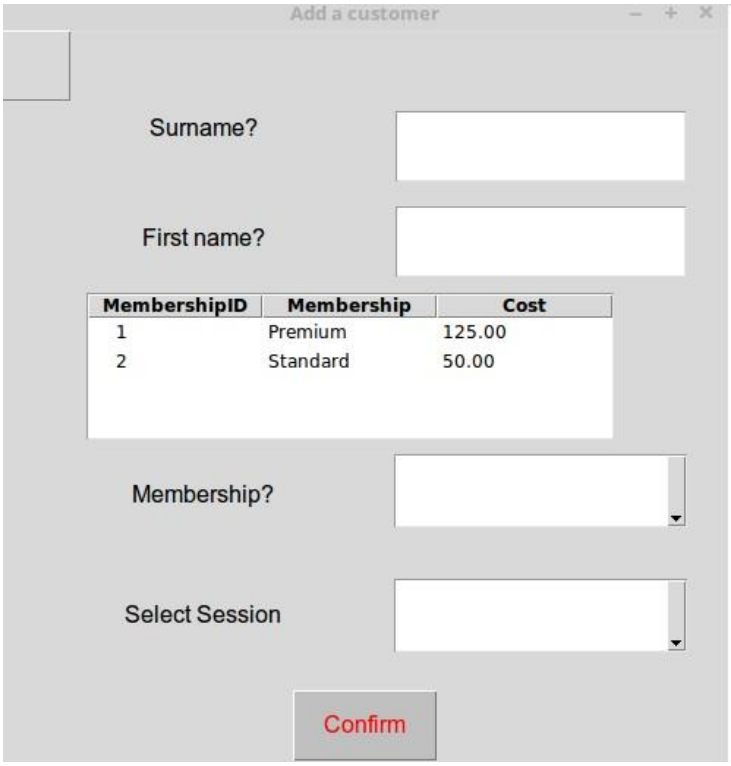
The creation process went pretty smoothly we stopped for implementing the project with a well-known library of python – tkinter which was covered in one of our previous modules. The creation of a GUI elements was the simplest and most presentable way for the whole group and everyone was happy with developing it under this circumstances. For the database manipulation we stopped on another library by python acting as a sql editor and creator. First of all we started with submitting gui for booking and adding a customer which other people from my group have developed. When everyone had the functionality assigned to them, we had to add everything under different classes and attributes to best represent our user case done under StarUML or Rational Rose. For the further coming expecting dates and deadlines we developed a Gantt chart to follow our steps until completion.

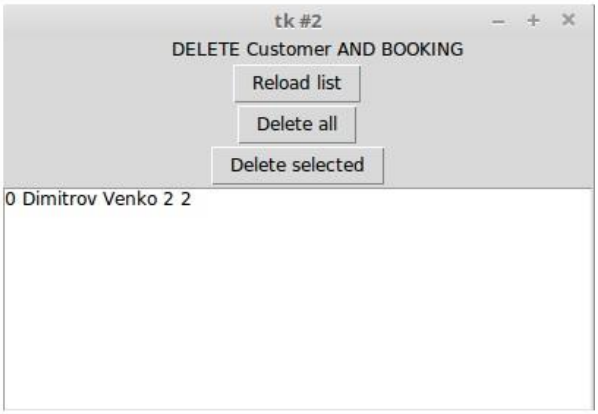
After the initial creation of our tasks we had a minor problem making the system adapt to the graphs created for it to function properly. For our chosen task we had the four layer design architecture as well as a gang of four (GoF) design pattern. In this time moment we had to combine our functionalities so we start pairing them by separate pairs two by two which would interact better one with another until we had the complete system. Upon creation of the prototypes we demonstrated the representation to the client. Which we received good notes to work on for the next weeks. We tried to stay on track and not fall on the schedule we had created for the next

week we polished the smaller things and the requirements by the client.



For the group management we have chosen to use github at which we had initial problems. We have passed most of our consistent lab sessions in the agile scrum way of sprints. After every session group mates showed their advancement to their part and asked the others for ideas, improvements and problems showing up with development. The scrum was done almost daily with screenshots and communication outside classes. Group testing were done in labs and in personal free time. As everything was done as a real on site client was present for our sessions and the testing compabilities.





For the testing part in the second prototype I have chosen to use the Add of Membership and account creation as it was one of the first things to be developed in the project group and it was directly correlated to my table and actions. As seen in the screen shot, after inputting the data from the “Add a customer” window, selecting membership and session type they all get sent to the database and furthermore are free to alter, update or be viewed by others. It has all of the data visible in the graph below in which was represented ID, type of Memberships, names and Session type.

Overall I am very satisfied with the way our group has handled the project, time schedules, work done in the time period with prototypes, graphs and the final functionality of the product. Choosing sprints and agile scum methodology to motivate and simulate our professional process really helped develop all of our capabilities and snap our productivity to the overall changing data.