Student Clubs Example

# Database Structure

Here is the Entity Relationship Diagram for our database linking *person* and  *activity* tables.

Graphical user interface, Word

Description automatically generated

We are going to build CLI menus for interacting with this database.

# Designing the CLI Menus

In order to interact with this data base, we are going to create some menus.

These menus will present some information and then ask the user to navigate using a list of options.

What information will the user need to see?

What options will allow the user to navigate to other screens?

What will be the sequence of events leading the user from one option to the other?

# Building a CLI application

Here is diagram of the Project file structure:

Text

Description automatically generated with medium confidence

The file structure shows how the application has been modularised into smaller sections. This follows the predominant [three-tier architecture design pattern](https://en.wikipedia.org/wiki/Multitier_architecture) for applications

Annotate this diagram with notes saying what each of the files do.

## Overview diagram of three-tier application

# Diagram Description automatically generated

Fork this Git: <https://github.com/AndrewDales/student_clubs/tree/Template> and open your version in PyCharm.

# Data Tier

We write queries to retrieve data from the database to present to the users and to write back data generated from the application.

Run *start\_up.py* in the python console and run:

session = Session()

names\_data = queries.qry\_names(session)

person\_2\_activities = queries.qry\_person\_activities(session, 2)

What do these queries do?

Now run the tests in *test\_db.py*. You wil see that some of the tests pass, but we need to finish two additional queries.

See if you can code up these queries to meet the test requirements.

# Presentation Tier

Look at the code in *menus.py*. This will create a generic CLI menu.

What are the components of the menu?

How are the option events (callbacks) triggered?

# Application Tier

The application\_layer is being programmed in *cli\_application.py*

The application layer uses queries to access any necessary information and organises the data into a structure that can be passed to the presentation menus.

This layer contains the logic which will allow the user to navigate from menu to menu.

The menu data is passed to the *create\_cli\_menu* to transform it into a Menu object, which can be displayed to the user.

Enter the code for the Main Menu – you will then be able to run the program, see the initial menu and run the (mock) callbacks.

def open\_main\_menu(self):  
 title = "Main Menu"

menu\_options = {"option\_list":  
 [("Select person", self.callbacks["open\_select\_person"]),  
 ("Select activity", self.callbacks["open\_select\_activity"]),  
 ("Quit", None)]  
 }  
 menu = {"title": title, "menu\_options": menu\_options}  
 menu = self.create\_cli\_menu("main\_menu", menu)  
 return menu

Note that application tier creates objects that are independent of the presentation layer. This will make it easier to maintain and build up the application as each layer can be changed separately. For example, changing the presentation tier to a GUI structure without radically changing the application tier logic.

Note also that that each method returns the menu object. This is not necessary for the application, but modularises the code and allows the developer to test each part of the program separately.

## Exercise

What should the person menu look like?

Write the code for the application tier logic that can be put into the menu structure.

Solutions

# Query all the activities in the database

def qry\_activities(session):

qry = session.query(m.Activity).order\_by(m.Activity.name)

return [(row.id, row.name) for row in qry.all()]

# Query the attendees of a given activity

def qry\_activities\_register(session, id\_num):

activity = session.query(m.Activity).get(id\_num)

return {"activity": activity.name, "id": id\_num,

"attendees": [f"{attendee.first\_name} {attendee.last\_name}" for attendee in activity.attendees]}

def open\_person\_menu(self, id\_num):  
 with self.session\_scope() as session:  
 person\_data = queries.qry\_person\_activities(session, id\_num)  
 title = person\_data["name"]  
  
 data = {"data\_title": "Activities", "data\_list": person\_data["activities"],  
 "person\_id": id\_num}  
 menu\_options = dict(option\_list=[("Add new activity", self.callbacks["open\_select\_activity"]),  
 ("Select new person", self.callbacks["open\_select\_person"]),  
 ("Return to main menu", self.callbacks["open\_main\_menu"])])  
 menu = {"title": title, "data": data, "menu\_options": menu\_options}  
 menu = self.create\_cli\_menu("person\_activity\_menu", menu)  
 return menu