COMP 636: Web App Assessment

Milestone submission due: 5pm Monday **9 October 2023**

Final submission due: 5pm Monday **30 October 2023**

Worth: 50% of COMP636 grade

Submit via Akoraka | Learn, with files set up and available on Github and Python Anywhere.

Introduction

BRMM car club runs a monthly competitive Motorkhana event. These are **different** from the 'Have-a-go' family fun day that you saw in the first assessment, although there are similarities. Your task is to develop a small Web Application to help the manage drivers, cars, courses and runs for a single competitive Motorkhana event. You will also write a report.

A Motorkhana has six courses, and each driver has two runs (attempts) at each course, with the best of those two runs counting towards their overall result for the day. Each **run total** is the driver's time in seconds plus any cone/wrong direction penalties (5 seconds per cone hit, 10 seconds for a WD).

A driver's **course time** is the *best* of their two run totals for that course. If a driver completes only one of their two runs, then their course time is the run total of the completed run. If a driver misses both runs, their course time will display as 'dnf' (did not finish).

For example, some **run totals** and calculated **course times** from a previous Motorkhana were:

Driver	Course	Run num	Seconds	Cones	WD	**Run Total**
101	A	1	33	1	1	**48**
101	A	2	50.5	0	0	**50.5**
105	A	1			0	
105	A	2			0	
101	F	1			0	
101	F	2	42.25	0	0	**42.25**

Driver	Course	**Course Time**	
101	A	**48**	Best of his two course A calculated run totals.
105	A	**dnf**	Did not finish course A (no times for A1 and A2).
101	F	**42.25**	No time for his F1 run, so best for course F is his F2 run total.

The **overall result** for each driver is calculated as the sum of their six course times. Drivers with fewer than six course times are awarded an overall result of 'NQ' (Not Qualified). That is, a driver with 'dnf' for any course is not qualified to win overall.

In competitive Motorkhana, Juniors are aged 12-**<u>25</u>** and have both their date of birth and their current season age (in whole years) recorded at the time of entry. Junior drivers who are 16 or younger must also have a designated caregiver (the driver ID of another driver who isn't a junior).

Each driver is associated with exactly one car (see the Data Model, below.) However, more than one person can drive the same car. For example, Jack Atwood and his mother, Maggie Atwood, are both competing in car number 18.

Download the Web Application Project Files from the Assessment block on the Learn page. These will get you started, including for the Milestone. You will add more routes and templates as you develop your app.

Important

This is an **individual** assessment. You may not collaborate or confer with others (e.g., telling others exactly how to do a task, or sharing code, or using ghost writers or AI, etc), but the discussion of **general** concepts (e.g., how Jinja templates work **in general**) is allowed. The University policies on Academic Integrity are here.

Functional Requirements

There will be two interfaces:

- A public/driver interface to allow drivers to view information and results.
- A club Administrator interface for admin staff to use to update, edit and add data. This interface requires administrators to access the system via an Admin button or link from the home page (do not add password functionality – security would be added to a full system to restrict access, but it is not required for this assessment). The Admin link will provide a gateway to the Admin features. The Admin features must not be visible in any public navigation (apart from the Admin link or button on the home page).

Driver Interface on the default / route

- **List of courses:** Make the **courselist** page display the course's image, rather than the name of the image file. Modify or tidy the template as appropriate.
- **Driver's run details:** A driver's name is selected from a drop-down list of drivers to display a page showing the driver's run details and run totals, including the course names (but not the course ID letter). Include the driver ID and names, and car model and drive class, as headings.
- **List of drivers:** Modify the **/listdrivers** route so that each driver's car details are also displayed. Do not display the car_num. Show them in surname then first name order, and use Bootstrap to display the junior drivers in yellow. Make the driver name a clickable link that also displays the driver's run details page (see above).
- **Overall results:** Show the overall results in a table, from best to worst overall result, and with any NQ results at the bottom (at the bottom of the list or as a note below the table). The table will include the driver ID and names (including '(J)' for juniors), and car model. Display all 6 course times for each driver, as well as their overall result. The winner should display "cup" next to their result, and the next 4 display "prize" (just the text is fine, or optionally suitable alternative symbols).
- **Bar graph:** Display a horizontal bar graph of the top 5 drivers overall. Modify the supplied script in **top5graph.html**, but using driver names and overall results as passed variables instead of hard-coded constants. The horizontal bars should use one of the 140 standard HTML colour names (not pink or lime or anything too pale!), and make the max width sensible but considerably wider than 320 pixels.

Administrator Interface

- **Junior driver list:** Display the junior drivers including any caregiver names, ordered by age from oldest to youngest, then by surname.
- **Driver search:** Search for drivers by first name or surname, allowing for partial text matches.
- **Edit runs:** Enter or edit times, cones and WD for a run. The admin will be able to select a driver and then edit any runs for that driver, or select a course to edit any runs for that course.

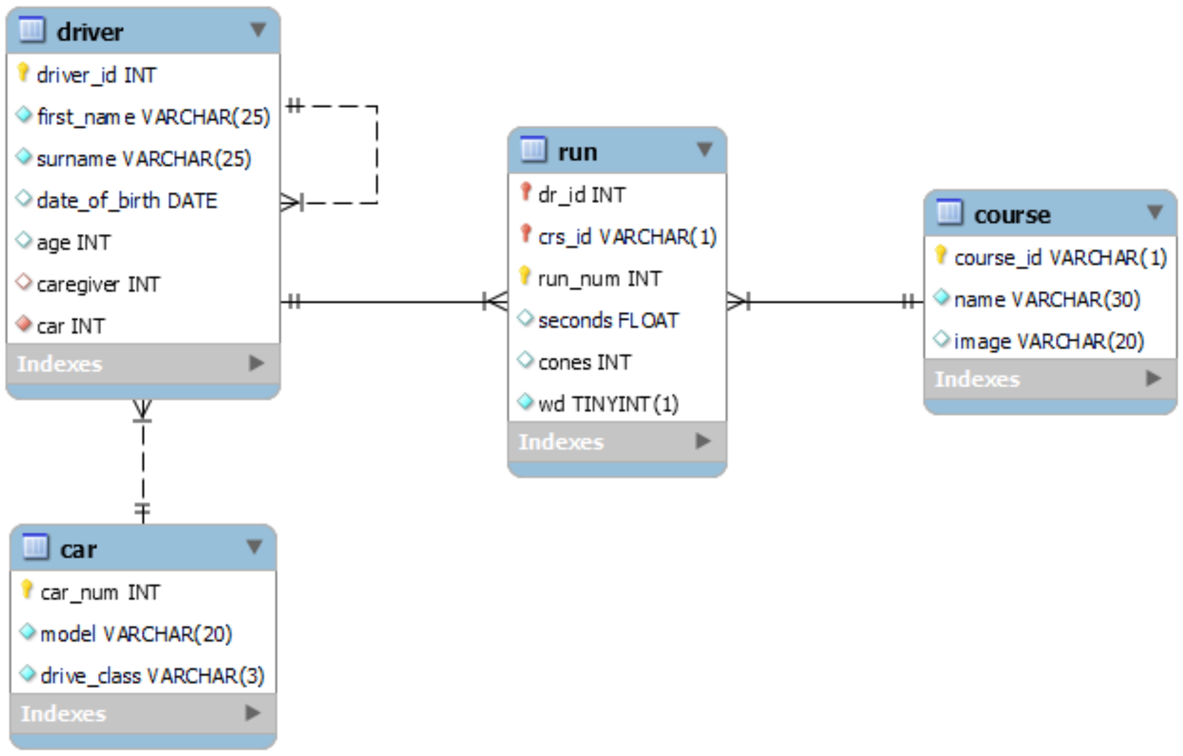
    - The admin should not have to re-enter any existing values that are not being entered/ edited.
- **Add driver:** Add a new driver and assign them to an existing car. (Do not add any new cars.)
    - This must also add 12 blank runs (two for each course) with null times and cones. WD will default to zero. (Times, cones and WD will be entered using the Edit Runs feature above.)
    - Must be able to specify whether new driver is a junior. If so, must also enter a date of birth, and calculate and record the age, and ensure that the driver has a caregiver, if needed (provide a list of eligible caregivers to select from). If not a junior, none of these options (date of birth, caregiver selection) should be visible.

Report

Your report must be created using GitHub Markdown format and saved in the **README.md** file of your GitHub repository. It does not need to be a formal report – a tidy document using the following headings will be sufficient. Write a brief project report that includes:

- **Web application structure:**
    - Outline the **structure** of your solution (routes & functions, and templates). This should be brief and can be text-based or a diagram (as an image).
    - It must indicate how your **routes** and **templates** relate to each other, as well as and what **data** is being passed between them.
    - Do not just give a list of your routes. Do not include all of your code. Do not describe the interface and user experience or html layout.
- **Assumptions and design decisions**:
    - Detail any **assumptions** that you made (what isn't clear or stated in this brief that you had to assume or ask about).
    - Discuss the **design decisions** you made when designing and developing your app (what design options you weighed up, why you designed your app the way that you did, your decisions about the routes, templates, navigation, broad layout, etc, that you made).
    - Note your assumptions and your decisions as you work, so you do not forget them! For example, did you use multiple similar pages, or share some page templates perhaps with hidden items? Did you use GET or POST to request and send data, and how and why? You will have considered many design possibilities. These are only two examples.
- **Database questions**: Refer to the supplied ***motorkhana_local.sql*** file to answer the following questions:
    - What SQL statement creates the car table and defines its three fields/columns? (Copy and paste the relevant lines of SQL.)
    - Which line of SQL code sets up the relationship between the car and driver tables?
    - Which 3 lines of SQL code insert the Mini and GR Yaris details into the car table?
    - Suppose the club wanted to set a default value of 'RWD' for the **driver_class** field. What specific change would you need to make to the SQL to do this? (Do not implement this change in your app.)
    - Suppose logins were implemented. Why is it important for drivers and the club admin to access different routes? As part of your answer, give two specific examples of problems that could occur if all of the web app facilities were available to everyone.
- **Image sources:** It is not necessary to use any external images in your web app, apart from the 6 course diagrams provided, but if you do, ensure you reference the image source in your report.

Data Model



Model Notes:

Child table.*field* *	(refers to)	**Parent table**.*field*
driver.*caregiver*	>——	**driver**.*driver_id*
driver.*car*	>——	**car**.*car_num*
run.*dr_id*	>——	**driver**.*driver_id*
run.*crs_id*	>——	**course**.*course_id*

* the 'Foreign Key'

Cars are classed (*drive_class*) as one of '4WD', 'RWD' or 'FWD' (4-wheel drive, rear-wheel drive, front-wheel drive). Your app does not need to modify the data in this table in any way.

Drivers with a *date_of_birth* must also have an *age* calculated and recorded. Drivers aged 16 or younger must have a *caregiver*. All drivers must have a *car*.

The driver's *driver_id* is an 'auto-incremented' field. That means that when adding a new driver, do not supply a *driver_id*; the database will assign one for you.

Each run record is for a particular driver (*dr_id*) on a particular course (*crs_id*) and is for the driver's first attempt or second attempt at that course (*run_num,* value of 1 or 2). *cones* is usually less than a handful and never more than 25. *wd* is a Boolean type field, shown in the model above as type TINYINT(1), and defaults to the value 0 (false).

The number of courses at a BRMM competition Motorkhana is six (*course_id* is always A-F).

Project Requirements

You must:

- Use only the COMP636 technologies (Python & Flask, Bootstrap CSS, MySQL). Do not use SQLAlchemy or ReactJS (or other similar technologies) in your solution.

  Do not use any scripts, including JavaScript, except for the <script> at the bottom of base.html, and the supplied script in **top5graph.html**. Do not write your own CSS (use Bootstrap).
- Use the provided SQL files to create the database within your MySQL database & pythonAnywhere. This also creates initial data in the database.
  - You can re-run this SQL script at any time to reset your data back to the original version and remove any changes you made to it.
- Use the provided files to develop a Flask Webapp that:
  - Meets the functional requirements.
  - Is appropriately commented.
  - Connects to your database.
  - Provides appropriate routes for the different functions.
  - Provides templates and incorporates HTML forms to take input data.
  - Uses Bootstrap CSS to provide styling and formatting.
- Include your report as outlined above.
  - This report must be created using GitHub Markdown and saved in the **README.md** file of your GitHub repository.
- Create a **private** GitHub repository that contains:
  - All Python, HTML, images and any other required files for the web app.
  - A **requirements.txt** file showing the required pip packages.
  - Your project report as the **README.md** document.
  - Your repository must have a **.gitignore** file and therefore *not* have a copy of your virtual environment.
  - Add **lincolnmac (computing@lincoln.ac.nz)** as a collaborator to your GitHub repository.
- Host your system (including database) using pythonAnywhere
  - Add **lincolnmac** as your “teacher” via the site configuration.

Project Hints

Create your GitHub repository first, and create all your required code and files in your local folder. Regularly commit and push changes from your local computer to your GitHub repository.

PythonAnywhere is case sensitive so test your app early – **we will mark the pythonAnywhere version of your app**.

Spend some time sketching the structure of your application before you start developing. Think about which features could share the same (or nearly the same) templates. Remember that you can nest templates (templates within templates).

Take note of your design decisions, compromises, workarounds, etc. for your report as you develop your web app. Otherwise you may struggle to remember all of the issues you worked through afterwards.

Milestone Submission (10 marks, due 9 Oct)

This milestone is to ensure that your app is correctly configured, and any set-up issues are resolved early. The milestone does not require any additional functionality; by this date you only need to host the provided code, check that everything is set up and that the web app and provided routes run correctly, and grant us access.

Submit via the link on the Learn COMP636 page:

- Your PythonAnywhere URL (e.g., joebloggs.pythonanywhere.com/)
- Your GitHub repository name (e.g., joebloggs99)

For this submission you **must** have:

- Your GitHub repository set up correctly.
- The provided files loaded in GitHub and in pythonAnywhere.
- Your database set up on PythonAnywhere.
- Your app hosted on PythonAnywhere.
- The **/listdrivers** and **/listcourses** routes working (as provided in the files).
- Granted access to your PythonAnywhere account (set **lincolnmac** as teacher).
- Granted access to your GitHub repository (**lincolnmac** or computing@lincoln.ac.nz as collaborator) .

IMPORTANT: Do not change your GitHub or Python Anywhere files until *after* you have received your Milestone marks. You may continue to work in the local copy on your computer in the meantime.

At this submission we will check your GitHub, PythonAnywhere and database setup.

Set-up Requirement	**Marks Available**
GitHub Repository set up and shared	3 marks
PythonAnywhere web app hosting correctly configured, including home URL and database setup, and teacher access granted	5 marks
/listdrivers and /listcourses routes and pages (as provided) running on PythonAnywhere	2 marks
TOTAL	**10 marks**

Final Submission (90 marks, due 30 Oct)

Submit your URLs *again* via the link on the Learn COMP636 page:

- Your PythonAnywhere URL (e.g., joebloggs.pythonanywhere.com/)
- Your GitHub repository name (eg, joebloggs99)

This confirms where your work is, and tells us that your final submission is ready for marking.

Final Submission Marking

Report and General Project Aspects (35 marks):

Project Element	Marks Available
Project Report – Part 1: • Outline the structure of your solution (routes & functions). • Detail any assumptions that you made, discussing your design decisions. • Report created using GitHub Markdown and saved in the README.md file of your GitHub repository Project Report – Part 2: • Report Database Questions sufficiently answered. Spelling, presentation, etc	17 marks • 7 marks assigned to appropriate and accurate structure outline. • 9 marks assigned to Assumptions and Design Decisions. • 1 mark for .md being in the right place and a reasonable length. (Just a heading, or a couple of sentences, is not a reasonable length.) 10 marks 3 marks Spelling, punctuation, grammar and presentation, and appropriate referencing of external images (if any).
Consistent 'Look and Feel' (interface, Bootstrap styling & templates, ease of use, etc)	5 marks
TOTAL	**35 marks**

Functional Project Aspects (55 marks):

Within each of the functional areas (see table below with indicative marks) we are looking for:

- Functionality working as specified in the requirements.
- Well commented and formatted HTML, SQL, and Python code throughout.
- A user interface that looks and functions to a professional standard, including Bootstrap colour and styling choices, and sensible navigation.
- Primary key fields are mainly for internal system use only and should only be made visible to system users when mentioned in the requirements.
- Data Validation on forms. Wise choice of form elements.
- Well-structured SQL queries.
- Appropriate naming, both of variables and labels

An *indication* of marks (may be adjusted when marking) :

Item	**Interface**	**Functionality**	**Approx. Marks**
Access	Public	/ route exists and operates	1
	Admin	Admin gateway exists and operates. Admin functionality not available via any public interface.	2
Navigation	*All*	Good navigation throughout.	4
List of courses	Public	Course list including images	2
Driver's run details	Public	Driver name selectable in dropdown. Run details correct. Driver ID, names and car details included.	5
List of drivers	Public	Driver list including car names. Clicking name displays run details.	4
Overall results report	Public	Overall results correct and complete, with juniors and winners identified.	6
Bar graph	Public	Horizontal bar graph displays using top 5 driver data.	3
Junior driver list	Admin	Junior list including caregivers.	4
Driver search	Admin	Driver search works and accepts partial text matches in search term.	6
Edit runs	Admin	Runs can be edited. Appropriate validation. Existing values do not have to be re-entered.	8
Add drivers	Admin	New driver can be added, existing car can be selected, appropriate validation, loops to create 12 runs with appropriate values. Appropriate way to allow Junior entry. Extra form controls and validation and verification for Juniors, with age correctly calculated.	10
TOTAL			**55**

Functionality not required:

Your web app should **not** add, edit, or delete **cars** or **courses** in any way.

Your app does not need to check *existing* ages and dates. You can assume the ages and dates we supply are correct. Once a driver is created, the age does not need to be updated.

You may modify **data** in your database for testing purposes and may add new data, but do not alter the courses, and you must **not** modify the schema (the model shown on page 3) .

Markers will modify or add alternative data to your database as part of the marking process.