COMP 636: Web App Assessment – Summer School 2024

Milestone submission due: 5pm Friday 2**4 January 2025 via Learn**

Final submission due: 5pm Monday **10 February 2025 via Learn**

Worth: 50% of COMP636 grade

Submit via Akoraka | Learn, with files set up and available on GitHub and PythonAnywhere.

Introduction

Aotearoa Tours Ltd (ATL) is a New Zealand based tourism company that runs international tours for customers from all over the world.

ATL would like to keep a record of Customers, Tours and Tour Groups (groups of Customers that have been on a particular Tour). Some Tours have age restrictions on them (i.e., adults only).

Information is stored about each Customer, and they are assigned a unique ID number.

Tours have a name and an itinerary which is a list of Destinations that a Tour visits.

A Tour Group is a group of Customers that goes on a particular Tour starting on a specified date.

You have been asked to help develop this system for ATL by undertaking the tasks described below.

Note: The requirements presented here are not exhaustive, you are expected to apply critical thought to them, and best practices taught in the course, as a key part of the software development process. Ask clarifying questions in the in-person or online support sessions.

Download the web application files from the Assessment block on Learn. 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. You may help others by *verbally* explaining concepts and making suggestions in general terms, but without directly showing or sharing your own code. You must develop the logical structure and the detail of your code on your own. No use of generative AI is permitted for any part of this assessment.

Code or content that is copied, shares a similar logic to others or is produced by generative AI *will receive zero marks for all parties involved*.

The university guidelines and policy on academic integrity can be found here.

Milestone Submission (5 marks, due 24 January)

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 changes to the code and templates provided.

By this date you only need to sync and share the provided files on GitHub, provide us teacher access to your PythonAnywhere, and host the provided code on PythonAnywhere so that the web app and provided routes run correctly.

Milestone Requirements

1) Create a **private** GitHub repository called **atl**.
    a. Your web application (**app.py**, etc.) must be in the main folder of your repository (not in a subfolder)
2) Host your web app on PythonAnywhere.
    a. Use files cloned/pulled from your GitHub repository – don't manually upload files (except connect.py).
    b. Your **atl** web app folder must be in your PythonAnywhere home directory (which should happen automatically when cloning the files from GitHub).
    c. Your MySQL database must be called **atl** and contain the required tables and data.
3) The provided web application pages must load correctly in PythonAnywhere:
    a. Home page: **/** route; **home.html** template page.
    b. Tours page: **/tours** route; **tours.html** template page.
    c. These pages require no modification of **app.py** or their HTML files to work.

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 successfully on PythonAnywhere.
- The **/** and **/tours** routes working (as provided in the files).
- Granted access to your PythonAnywhere account (set **lincolnmac** as teacher).
- Granted access to your GitHub repository (invite **lincolnmac** or computing@lincoln.ac.nz as collaborator).

Submit the following via the link on the Learn COMP636 page:

- Your PythonAnywhere URL (e.g., yourname1234567.PythonAnywhere.com/)
- Your GitHub username and repository name (e.g., yourname1234567/atl)
- Just copy the link text and the repository name. Don't add any other text.

Set-up Requirement	**Marks Available**
GitHub Repository set up and shared	1 mark
PythonAnywhere web app hosting correctly configured, including home URL and database setup, and teacher access granted	3 marks
/ and **/tours** routes and pages (as provided) running on PythonAnywhere	1 mark
TOTAL	**5 marks**

IMPORTANT: Do not pull further changes to your PythonAnywhere files until *after* you have received your Milestone marks. You may continue to work in the local copy on your computer in the meantime and you should also commit and *push* to your GitHub repository, but do not *pull* changes to PythonAnywhere or make any other changes there until the Milestone is marked.

Final Submission (95 marks, due 10 February)

Web application requirements

1) **Page header:** On every page, include a page header which includes: the name of the app and a navigation bar with links to the pages.
2) **Image and text:** Add an appropriate picture with some (brief) introductory text to the home page.
3) **Customer list:** Complete the **tourlist.html** template to display a list of customers who are in a particular tour group.
    a. The tour name and start date must be shown clearly at the top of the list
    b. A list of customers on the tour must be shown in alphabetical order by family name
    c. Customers with the same family name should be ordered by date of birth (youngest first)
    d. A customer must be clickable to display a customer booking overview (see 8 below)
4) **Add booking:** Complete the **/booking/add** route to insert a customer booking for a future tour into the database. Age restrictions must be enforced. (You do not need to edit any bookings for this assessment.)
5) **Customer search:** Create templates and routes to allow customers to be searched for and for the results to be displayed.
    a. Search results must show customer family name and first name along with contact details.
    b. Results must be clickable to display a customer booking overview (see 8 below)
6) **Add customer:** Create templates and routes to allow customers to be added to the system, and ensure appropriate validation is in place.
7) **Edit customer:** Create or modify templates and routes to allow customer information to be edited.
8) **Customer booking overview:** Create templates and routes to display a customer booking overview page that shows all the bookings that a customer has made.
    a. The page must show the name of the customer and the total number of tour destinations that customer has booked overall (any destination visited multiple times should be counted multiple times). Include all tour groups (past, present and future) in the tour destination total.
    b. For each booking display the tour name, departure date and number of tour destinations.
    c. The overview must show past, current and future bookings.
9) **Project Report:** Create a report as described in the Report section below in the README.md file in your GitHub repository.

General requirements

1) The application should follow the best practices mentioned in class, even where not specifically requested.
2) Data that is entered should be validated and errors from inappropriate data handled without crashing the web application.
3) Information should be presented in a manner appropriate for a professional web application, such as appropriate headings, labels, and date and number formatting.
4) General presentation and user interface should be tidy and professional making use of Bootstrap CSS for formatting, but it does not need to be ‘fancy’. Clean and functional is sufficient.

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:

- **Design decisions**:

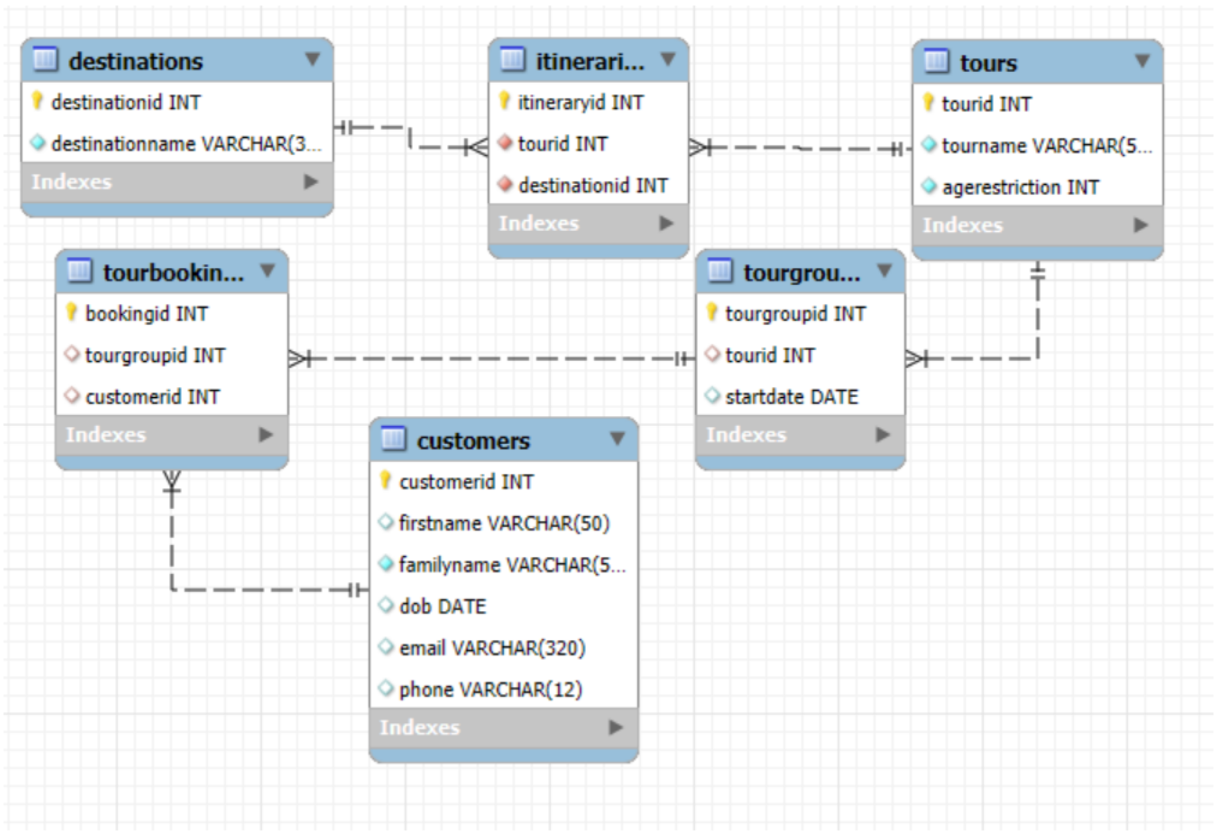
  Discuss the **design decisions** you made when designing and developing your app: why you created one set of pages, routes and functions to connect and interact in this way, and not in some other way. Discuss the structural and logical options you considered.

  For example, , does the edit open a different template for editing or does it use the same template with IF statements to enable the editing? What GET or POST choices to request and send data did you make and why? (These are just examples; you do not have to include them in your own discussion.) You will have considered many design possibilities; write personally in plain language about your development decisions.

  Make notes about your decisions *as you work*, so you do not forget them!
- **Image sources:** Reference external image sources in your report.
- **Database questions**: Refer to the supplied **atl-local.sql** file to answer the following questions. *Do not* implement these changes in your app, just write your answers in your report:

  1. What SQL statement creates the **tours** table and defines its fields/columns? (Copy and paste only the relevant lines of SQL.)
  2. Which lines of SQL script set up the relationship between the **tours** and **tourgroups** tables?
  3. The current ATL only works for individual travellers. Write SQL script to create a new table called **families**, which includes a unique family ID, the family name, an optional short description. The ID must be added automatically by the database. (Relationships to other tables not required.)
  4. Write an SQL statement that adds a row for a new family to your new **families** table. Make up some values for a fictional family.
  5. What changes would you need to make to other tables in the database to fully incorporate the new **families** table into the data model below? (**Describe** the changes. SQL script not required.)

Data Model



Model Notes:

Child table.*field* *	(refers to)	**Parent table**.*field*
intineraries.*tourid*	>——	**tours**.*tourid*
intineraries.*destinationid*	>——	**destinations**.*destinationid*
tourgroups.*tourid*	>——	**tours**.*tourid*
tourbookings.*tourgroupid*	>——	**tourgroups**.*tourgroupid*
tourbookings.*customerid*	>——	**customers**.*customerid*

* the ‘Foreign Key’

Project Constraints

You must:

- Use only the COMP636 technologies (Python & Flask, Bootstrap CSS, MySQL). Do not use SQLAlchemy or ReactJS (or other similar technologies or libraries) in your solution.
    - Do not use any scripts, including JavaScript and do not write your own CSS (use Bootstrap).
    - Exceptions: You may use:
        - The Bootstrap <script> at the bottom of **base.html**.
        - The supplied form validation javascript.
- Use the provided SQL files to create a database within your local MySQL and in PythonAnywhere. Each script also populates the database with initial data.
    - You can re-run the appropriate SQL script at any time to reset your data back to the original version and remove any changes you made to it (if it starts to get messy during testing).
    - You may modify *data* in your database for testing purposes and may add new data, but you must not modify the *schema* – **do not** add any new tables or columns.
    - When marking, we will not change the *structure* of the data, but we will modify or add alternative data to your database as part of the marking process. You can rely on table names and columns staying the same, but the data in the rows in the tables may change.
- Use the provided files to develop a Flask web app that:
    - **Must be in a top-level folder called 'atl' (locally and on PythonAnywhere).**
    - Meets the functional requirements.
    - Is appropriately commented.
    - Connects to your database.
    - Uses **%s** to insert values into SQL statements.
    - 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 in your GitHub **README.md** file as outlined earlier.
    - This report must be created using GitHub Markdown and saved in the **README.md** file of your GitHub repository.
- Create a **private** GitHub repository called **atl** that contains:
    - All Python, HTML, images and any other required files for the web app.
    - Your repository must have a **.gitignore** file and therefore *not* have a copy of your virtual environment.
    - A **requirements.txt** file showing the required pip packages.
    - Your project report as the **README.md** document.
    - Add **lincolnmac** (computing@lincoln.ac.nz) as a collaborator to your GitHub repository.
    - **Your repository must show a minimum of two commits per week after the milestone submission** (but do not pull to PythonAnywhere until after your Milestone has been marked).
- Host your system (including database) using PythonAnywhere.
    - Add **lincolnmac** as your "teacher" via **Account** > **Education**.
    - The webapp must be in a directory called '**atl**' in the home directory.

Project Hints

Create your GitHub repository first then create all your required code and files in your local folder, then push to GitHub.com and clone/pull changes through to PythonAnywhere.

IMPORTANT: We will mark the *PythonAnywhere version* of your app. Even if your web app works locally on your computer, it may not work on PythonAnywhere. PythonAnywhere is case sensitive in ways that your local web application is not and can have other small differences. So, frequently test your app in PythonAnywhere, *as you develop it (pull to PythonAnywhere, reload, test)*. This makes it easier to debug issues as they arise and to ask for help. **You won't get marks if your local version works but your PythonAnywhere version doesn't.** So, test in PythonAnywhere well before the due date!

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

Make notes of your design decisions, compromises, workarounds, etc. as you develop your web app. Otherwise afterwards you may struggle to remember all of the issues you worked through, when it comes time to discuss those design decisions in your report. Do not include masses of insignificant decisions in your report though.

The requirements in this project brief are not exhaustive, you are expected to apply critical thought and think about the user experience of the web application.

Final Submission (95 marks, due 10 February)

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

- Your PythonAnywhere URL (e.g., yourname1234567.PythonAnywhere.com/)
- Your GitHub username and repository name (e.g., yourname1234567/atl)
- Just copy the link text and the repository name. Don't add any other text.

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

Report and General Project Aspects (25 marks):

Project Element	Marks Available
Project Report – Part 1: • Discuss your design decisions (aim for discussion of 7-15 decisions). Also detail assumptions that you made, if any. • 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.	**8** marks • 7 marks for appropriate personal discussion, in your own words. • 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 **2** marks Spelling, punctuation, grammar, and presentation, and appropriate referencing of external images.
Consistent 'look and feel' (interface, Bootstrap styling & templates, ease of use, etc). No additional CSS or scripts. COMP636 technologies only.	**5** marks
TOTAL	**25 marks**

Functional Project Aspects (70 marks):

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

- Functionality working as specified in the requirements and demonstrating critical thought about the implementation and user experience.
- 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, sensible navigation and appropriate sorting of lists.
- ID numbers for table rows are mainly for internal system use only and should be made visible to system users when only 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	**Functionality**	**Approx. marks**
Access	Home page exists via '/' route, with appropriate layout and no broken links. Picture and text added to home page.	3
Navigation	Appropriate use of extending templates. Sensible, well laid-out navigation throughout.	3
Customer List	Appropriate display of customer information, click through to booking overview and sorting applied	6
Customer Search	Appropriate routes created, search by first and/or last name with appropriate display and linking of results	7
Add booking	Appropriate interface to select the customer, tour and tour group. Only future tour groups are available to book. Age restrictions are enforced.	16
Add customer	Add Customer – a customer can be added with good validation and error messages in place.	9
Edit customer	Customer can easily be selected for editing with appropriate validation and error messages in place.	11
Customer Booking overview	Customer name and summary information shown along with breakdown of the details.	15
TOTAL		**70**