Project – Part 1

Today, we will be building a website for Torquay (‘Tor-kee’) Towers, a hotel in the English countryside.

The site will have two interfaces (besides the built-in Admin interface):

1. one for visitors,
2. and one for hotel staff.

Visitors’ Interface

- Display information about the hotel (location, images, description, what’s in the surrounding area, etc.)
- View vacancies (dates when rooms are available)
- Book a stay at the hotel
- Fill in a form requesting more information
- Leave a review of the hotel

Staff Interface

- View bookings and vacancies
- Add, modify, and cancel bookings
- Read messages from guests requesting more information
- Delete reviews (We only want guests with a touch of class…)

Tasks

1. Create a GitHub repository for the project (private)
2. For Groups
    1. Add permissions for all team members to commit/write to the repo.
    2. Each team member should clone this onto their local computer.

3. Build Django app skeleton
    1. Include requirements.txt, .gitignore, and the correct timezone in the project’s settings.

    2. Include relevant migrations in your commits.

4. Create ‘visitors’ app skeleton.
    1. This will manage the Visitors’ Interface, described above.
    2. Register the app and its urlconfig in the project.
    3. Include relevant migrations in your commits.

5. Visitors: create info page (In the visitors app create a view and template for the task below)
    1. Make an attractive page with paragraphs of interesting text and relevant images. Make sure to use the proper CSS styling, etc.
    2. Add prominent links to the booking page and reviews page (add dummy link (with a name but the destination should be #) – these will be changed later to the propper link).

6. Adding Navbar:
    1. Add a navbar to your base template so it can be seen on all the site pages.
    2. It will contain a link to the home page at the left, and login/logout info on the right (your user name and ‘Logout’ if logged in, otherwise a ‘Login’ button).

7. Visitors: Create login and logout pages
    1. You’ll need urls, view functions, and templates…
    2. Include relevant migrations in your commits.
    3. Also update the nav bar to include these changes.

8. Design models to represent hotel vacancies and bookings.
    1. I recommend you do this as a team strategy session, taking notes but not writing any code yet.

    2. What information do we need to store? Should we store vacancies, or bookings, or both?

    3. How many rooms does the hotel have? How many people can stay in each room?

    4. What is the price per room? Or per person?

    5. Should there be a minimum/maximum length per stay?

    6. Carefully plan the data types and structures. How many tables do you need? Which columns should each table have?

7. Also write algorithms to decide:
    1. if a visitor can book a stay (for X people in Y rooms, checking in on date C and checking out on day D).
    2. how to show vacancies to the user on the page (dates, days, rooms, people…?)

8. These algorithms will be coded later.

9. Visitors: Build the models from the previous task
    1. Include relevant migrations in your commits.
    2. Ensure all data is visible in the Admin site.

10. Visitors: Create a view function to show vacancies called booking.
    1. At this point we only need to get the days that are available for booking when getting the info from your database.
    2. Use the algorithm you developed earlier.

11. Visitors: Build the Booking page
    1. focus on getting the page to look right.
    2. Display the relevant options, depending on which vacancies the hotel has.

12. Visitors: Add to the Booking view
    1. Write the view code to handle input for a booking; this should be added to the view that takes all the vacancies.
    2. Validate data! Don't allow a visitor to book with invalid data (stay too long/short, rooms which are already booked, dates which are not free, etc.)
    3. 'Wire' this up to the booking page.
    4. Ensure booking data is visible in the Admin site.

13. Visitors: Implement the Info Request page.
    1. Use a form to get contact details and a free text message from the visitor.
    2. Store the data in the database.
    3. Ensure data is visible in the Admin site.