Restaurant Management system

create a restaurant management system with functionalities for handling menus, orders, reservations, and inventory.

1. Models (`restaurant/models.py`):

The models represent the structure of the database tables. In this case, you have models for `MenuItem`, `Table`, `Reservation`, `Inventory`, `Order`, and `OrderItem`.

`MenuItem` Model:

This model represents an item on the restaurant's menu.

- `name`: The name of the menu item.
- `description`: A text field describing the item.
- `price`: The price of the menu item as a decimal field.
- `is_available`: A boolean indicating if the item is available.

`Table` Model:

This model represents a restaurant table.

- `number`: Unique identifier for each table.
- `capacity`: The number of people the table can accommodate.

`Reservation` Model:

This model handles customer reservations.

- `customer_name`: The name of the customer.
- `customer_contact`: The contact information for the customer.
- `reservation_time`: The time of the reservation.
- `table`: Foreign key to the `Table` model, representing which table is reserved.

`Inventory` Model:

This model keeps track of inventory items.

- `item_name`: The name of the inventory item.
- `quantity`: The number of units in stock.
- `unit`: The unit of measurement for the item (e.g., "kg", "liter").

`Order` Model:

This model represents customer orders.

- `customer_name`: The name of the customer.
- `items`: Many-to-Many relationship with the `MenuItem` model through the `OrderItem` model.
- `order_time`: Timestamp of when the order was placed.
- `total_amount`: Total price of the order.
- `is_completed`: Boolean indicating whether the order is completed.

`OrderItem` Model:

This is an intermediate model between `Order` and `MenuItem`, representing each item in an order.

- `order`: Foreign key to `Order`.
- `menu_item`: Foreign key to `MenuItem`.
- `quantity`: Number of units of the menu item in the order.

---

2. Forms (`restaurant/forms.py`):

The forms define the structure for HTML forms to create or update the models.

`MenuItemForm`:

- Used to create or update a `MenuItem`.
- Includes fields: `name`, `description`, `price`, and `is_available`.

`ReservationForm`:

- Used to create or update a `Reservation`.
- Includes fields: `customer_name`, `customer_contact`, `table`, `reservation_time`, and `notes`.

`OrderForm`:

- Used to create or update an `Order`.
- Includes fields: `customer_name`, `items`, and `total_amount`.

`InventoryForm`:

- Used to create or update an `Inventory` item.
- Includes fields: `item_name`, `quantity`, and `unit`.

---

3. Views (`restaurant/views.py`):

The views define the logic for rendering templates and handling form submissions.

`add_menu_item`:

This view handles adding a new menu item to the database.

- Renders a form (`MenuItemForm`) for creating a new menu item.
- If the form is valid (submitted correctly), the menu item is saved.

`update_menu_item`:

This view handles updating an existing menu item.

- It retrieves the menu item using the primary key (`pk`) and updates it with the submitted data.

`make_reservation`:

This view handles making a reservation.

- It displays the reservation form (`ReservationForm`) and processes it to save the reservation in the database.

`place_order`:

This view processes placing a customer order.

- It displays the order form (`OrderForm`) and saves the order and its items.

`update_inventory`:

This view updates an inventory item, allowing the user to modify stock levels.

---

4. URLs (`restaurant/urls.py`):

This file maps URL patterns to the corresponding views.

- `/menu/add/`: Maps to the `add_menu_item` view to add a new menu item.
- `/menu/<int:pk>/update/`: Maps to `update_menu_item` view to update a specific menu item based on its ID.
- `/reservation/make/`: Maps to `make_reservation` view for making a reservation.
- `/order/place/`: Maps to `place_order` view to place an order.

- `/inventory/<int:pk>/update/`: Maps to `update_inventory` view to update an inventory item.

---

5. Migrations:

Migrations are auto-generated Python scripts to reflect changes made to the models in the database schema.

- `0001_initial.py`: The initial migration for creating the database tables for `MenuItem`, `Table`, `Reservation`, `Inventory`, `Order`, and `OrderItem`.
- `0002_auto.py`: Subsequent migrations, such as renaming fields in the `Table` model.

---

6. Templates:

These are HTML files that are rendered by views.

- **`add_menu_item.html`**: A form for adding new menu items.
- **`make_reservation.html`**: A form for making reservations.
- **`menu_list.html`**: A list of menu items, displaying their name, price, and description.
- **`place_order.html`**: A form for placing an order.
- **`update_inventory.html`**: A form for updating inventory items.
- **`update_menu_item.html`**: A form for updating existing menu items.

---

7. Admin Panel (**`restaurant/admin.py`**):

Registers models so that they are accessible via the Django admin interface.

- Registers `MenuItem`, `Table`, `Reservation`, `Inventory`, `Order`, and `OrderItem` models with the admin panel, allowing restaurant staff to manage them through the Django admin dashboard.

---

8. Tests (**`restaurant/tests.py`**):

Unit tests are written to verify that the application behaves as expected.

`MenuItemTest`:

Tests the creation of `MenuItem` instances and checks if their attributes are correctly stored.

`ReservationTest`:

Tests the creation of a reservation, ensuring that it associates correctly with the `Table` model and stores customer details.

`MenuItemViewTest`:

Tests the view responsible for adding menu items, ensuring that the correct template is rendered and the form submissions are handled properly.

`MenuItemFormTest`:

Tests the validation of `MenuItemForm` for both valid and invalid form data.