## Step 1: Project Setup

- Install Django using pip: pip install Django.
- Create a new Django project: django-admin startproject recipe_manager.
- Create a new Django application within the project: python manage.py startapp recipes.

## Step 2: Model Setup

- Define the models for the recipe management application, such as Recipe, Ingredient, Category, and Chef.

Recipe should have:

- title (CharField): The title of the recipe.
- description (TextField): A description or instructions for the recipe.
- chef (ForeignKey to Chef): The chef who created the recipe.
- ingredients (ManyToManyField to Ingredient): The ingredients required for the recipe.
- categories (ManyToManyField to Category): The categories to which the recipe belongs.
- created_at (DateTimeField): The timestamp when the recipe was created.

Ingredient should have:

- name (CharField): The name of the ingredient.
- quantity (CharField): The quantity or amount required.

Category should have:

- name (CharField): The name of the category.

Chef should have:

- name (CharField): The name of the chef.
- bio (TextField): A brief biography or description of the chef.

- Run database migrations: python manage.py makemigrations and python manage.py migrate.

## Step 3: Create HTML Templates

- Inside the recipes application, create a templates directory.
- Create an HTML template for the home page (home.html) that displays a welcome message and a list of recipes.
- Create an HTML template for the recipe detail page (recipe_detail.html) that displays information about a specific recipe.

- Create an HTML template for the category page (category.html) that shows a list of recipes belonging to a specific category.
- Create an HTML template for the chef page (chef.html) that displays information about a specific chef and their associated recipes.

## Step 4: URL Configuration

- Open the urls.py file in the project's directory and add include to the urls.py from recipes app
- Define URL patterns for the home page, recipe detail page, category page, and chef page, mapping them to the corresponding views in the recipes application.

## Step 5: View Functions

- Open the views.py file in the recipes application.
- Write view functions for the home page, recipe detail page, category page, and chef page.
- In each view function, retrieve necessary data from the models and pass it to the corresponding HTML templates. Think about what each view should return to the context dictionary.
- Render the HTML templates using the render() function and return the resulting HTML to the user.

## Step 6: Static Files (optional as CSS is not the purpose of the project)

- Create a static directory inside the project's directory.
- Add a CSS file (styles.css) to the static directory for custom styling.
- Link the CSS file in the HTML templates to style the pages.

## Step 7: Run the Application