.

**1. README.md**:

* **Purpose**: Serves as the entry point for understanding the project. It provides a high-level overview of the system, its features, how to install and set it up, usage guidelines, and developer information. A good README is essential for usability and maintainability.

**2. db.sqlite3**:

* **Purpose**: This is the SQLite database file. In a Django project, it stores all the application data, including user accounts, clinic information, appointments, and everything else defined in your models. *Important:* This file is usually only used during development. For production environments, you'd switch to a more robust database (like PostgreSQL) and NOT commit the db.sqlite3 file to your repository.

**3. manage.py**:

* **Purpose**: A command-line utility for performing administrative tasks in Django. It's the primary way to interact with your Django project from the command line.
  + Running the development server (python manage.py runserver)
  + Creating database migrations (python manage.py makemigrations)
  + Applying migrations (python manage.py migrate)
  + Creating a superuser (python manage.py createsuperuser)
  + Running custom management commands (python manage.py <command\_name>)

**4. requirements.txt**:

* **Purpose**: A text file specifying the Python packages (and their versions) that your project depends on. It's used to ensure that all developers working on the project have the same dependencies installed. You can install all dependencies with pip install -r requirements.txt.

**5. /appointments/models.py**:

* **Purpose**: This file defines the data models for your application using Django's ORM (Object-Relational Mapper). Models represent the structure of your database tables as Python classes.
  + CustomUser: Extends the default Django user model to include custom fields like phone and role.
  + Clinic: Represents a dental clinic.
  + Room: Represents a room within a clinic.
  + Doctor: Represents a doctor.
  + Service: Represents a dental service.
  + Price: Represents the price of a service for a specific shift.
  + Roster: Represents a doctor's schedule (assignment to a clinic and shift).
  + Appointment: Represents a booked appointment.
  + Report: Represents a generated report (e.g., revenue report).

**6. /appointments/admin.py and /appointments/admin\_sites.py**:

* **Purpose**: These files configure the Django admin interface for your application. They allow you to manage your data through a web-based interface.
  + /appointments/admin.py: Registers models with the default Django admin site, allowing you to manage data through the standard Django admin interface (accessed via /admin/).
  + /appointments/admin\_sites.py: Defines custom AdminSite classes and model registrations for role-specific admin portals. This allows you to customize the admin interface and restrict access based on user roles. There are roles like CustomerAdminSite, DoctorAdminSite and other customized roles.

**7. /appointments/serializers/serializers.py**:

* **Purpose**: Defines serializers for converting model instances to JSON format and vice versa. Serializers are used to handle data serialization and deserialization in your REST API. They are important for exposing your data to external applications or frontend frameworks.

**8. /appointments/views/\***:

* **Purpose**: These files contain the view logic for your Django application. Views are Python functions or classes that handle incoming HTTP requests, process data, and return responses. There are files to implement the main GUI.
  + appointment\_views.py: Manages appointment view
  + auth\_views.py: Is the backend code for managing authenticator
  + clinic\_views.py: Handles the clinic and rooms management view
  + doctor\_views.py: Handles the doctor, roster and access management view
  + report\_views.py: Creates reports depending on the view, with correct data formatting to be showed.

**9. /appointments/management/commands/\***:

* **Purpose**: Contains custom Django management commands. These commands are used to automate administrative tasks or perform one-time operations.
  + create\_user\_roles.py: Creates the initial set of users with different roles.
  + assign\_permissions.py: Assigns the correct permissions to each user role.
  + setup\_clinic\_rotation.py: Sets up initial clinic locations, rooms, and doctor rotations.
  + setup\_system.py: Sets up users roles
  + setup\_test\_data.py: Sets up the test data required.

**10. /appointments/utils/\***:

* **Purpose**: Contains utility functions, constants, and other reusable code used throughout the appointments application.
  + constants.py: Defines constant values such as shift times, appointment statuses, and room types.
  + price\_calculator.py: Contains utility functions for calculating service prices.
  + validators.py: Contains validation functions for enforcing business rules related to appointment scheduling (e.g., no appointments on Fridays, surgery room restrictions).

**11. /dentaclinic/\*:**

* **Purpose:** These files represent the Django project-level configuration and settings:
  + settings.py: the Django project. It defines database settings, installed applications, middleware, static file locations, and other crucial settings.
  + urls.py: The main URL router, mapping URLs to specific views and functionalities within the application.
  + wsgi.py: Provides the entry point for WSGI-compatible web servers to serve your Django project.
  + asgi.py: Provides the entry point for ASGI-compatible web servers to serve your Django project.