Clinic App Report

Phase 2 of the Project - Advanced Programming Course 21060

Keivan Jamali 99104468 | Nassrin Sharifi 99104779
Erfan Bidmeshki 401103933

GitHub Repository

2/2/2024

Contents

1 Introduction

In this chapter, we try to defined our apps to the Jango and run it on server.

All the project is available on the djangoProject folder inside the *GitHub* repository. you can follow us there.

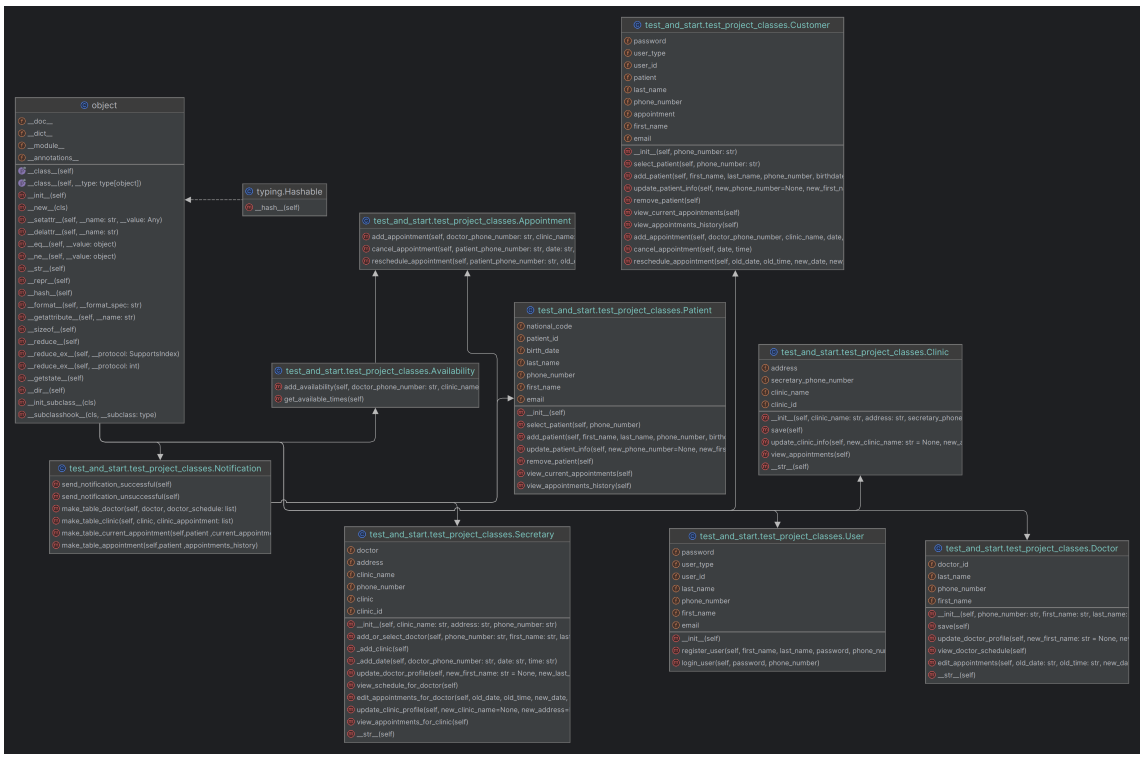

Figure 1: Class Diagram

2 model.py

1. The `User` model defines the user attributes like user ID, first name, last name, email, phone number, password, and user type. This model will hold information for all registered users in the system including patients, doctors, secretaries, etc.

2. The `Clinic` model contains attributes related to a clinic like clinic ID, clinic name, address, and secretary's phone number. This will store data for all clinics.

3. The `Doctor` model keeps information specific to doctors like doctor ID, phone number, first name, and last name.

4. The `Patient` model defines patient attributes such as patient ID, phone number, first name, last name, birth date, national code, and email.

5. The `Calendar` model holds appointment data with fields like appointment ID, doctor, clinic, patient, date, time, and cancellation status.

6. The `Availability` model contains a clinic's available slots with attributes like availability ID, doctor, clinic, date, time, and reservation status.

7. The `DoctorClinic` model makes a one-to-one relation between a doctor and their affiliated clinic.

8. The `CustomerPatient` model links a customer user account to their associated patient profile with a one-to-one relationship.

Together these model classes define the schema for storing user, clinic, doctor, patient, appointment, and availability data to power the online appointment booking system. Each model represents a different entity with its own set of attributes.

3 view.py

The `view.py` file contains some main functions. Let's describe each of them.

1. `register_or_login(request)` function:

   - This function handles the registration and login logic based on the HTTP method of the request.
   - It expects a POST request and retrieves the values of `action`, `phone_number`, and `password` from the request's POST data.
   - It creates a new `User` instance.

- If the action is "register", it retrieves additional information such as `first_name`, `last_name`, `email`, and `user_type` from the request's POST data.
- The `user` instance is then registered using the `register_user()` method of the `User` class, passing in the provided information.
- If the registration is successful, it checks the `user_type` and renders the appropriate template.
- If an exception occurs during the registration process, it catches the exception, generates an error message, and renders the "register_or_login.html" template.
- If the action is "login", it attempts to log in the user using the `login_user()` method of the `User` class, passing in the provided password and phone number.
- If the login is successful, it checks the `user_type` and renders the appropriate template.
- If an exception occurs during the login process, it catches the exception, generates an error message, and renders the "register_or_login.html" template.
- If the action is neither "register" nor "login", it generates an "Invalid action" error message and renders the "register_or_login.html" template.
- If the request method is not POST, it simply renders the "register_or_login.html" template.

2. `secretary_register(request)` function:

- This function handles the registration of a secretary.
- It expects a POST request and retrieves the values of `clinic_name`, `address`, and `phone_number` from the request's POST data.
- It creates a new `Secretary` instance, passing in the retrieved values.
- If the secretary's clinic is `None`, it generates an "Invalid clinic name or address" error message and renders the "secretary_register.html" template.
- If the secretary's clinic is not `None`, it renders the "secretary_page.html" template along with the `phone_number` value.
- If an exception occurs during the registration process, it generates an "Invalid clinic name or address" error message and renders the "secretary_register.html" template.
- If the request method is not POST, it simply renders the "secretary_register.html" template.

3. `secretary_page(request)` function: This function handles the requests made to the secretary page. It performs different actions based on the request method and the submitted form data. The possible actions within this function are enumerated as follows:

   (a) Return the secretary's phone number, clinic name, and clinic address.

   (b) Add or select a doctor with the provided phone number, first name, and last name.

   (c) Update the clinic profile with a new clinic name and/or address.

   (d) View appointments for the clinic.

   (e) Redirect to the register or login page.

4. `doctor_page(request)` function: This function handles the requests made to the doctor page. It also performs different actions based on the request method and the submitted form data. The possible actions within this function are enumerated as follows:

   (a) Render the doctor's schedule.

   (b) Add or select a doctor with the provided phone number, first name, and last name.

   (c) Update the doctor's profile with a new phone number, first name, or last name.

   (d) Add a new appointment for the doctor with a specified date and time.

   (e) Edit an existing appointment for the doctor by changing the date and time.

   (f) Redirect to the secretary page.

   (g) Redirect to the register orlogin page.

5. `patient_page(request)` function: This function handles the requests made to the patient page. Similar to the previous functions, it performs different actions based on the request method and the submitted form data. The possible actions within this function are enumerated as follows:

   (a) Render the patient's appointments.

   (b) Redirect to the register or login page.

4 rest of templates

4.1 base.html

This is the base template for the Clinic System website. It serves as the foundation for all other pages and provides a consistent layout and styling across the site.

1. **HTML Structure**

   The file begins with the standard HTML doctype declaration and opening and closing ‘<html>‘ tags. It specifies the language as English and includes the necessary meta tags for character encoding and viewport settings.

2. **Title**

   The ‘<title>‘ tag is used to set the title of the webpage. In this case, it uses a template language syntax (‘% block title %...% endblock %‘) to allow individual pages to override the default title with their own custom titles.

3. **Styling**

   The template includes a link to the Bootstrap CSS framework hosted externally. This provides pre-defined styles for elements and helps with responsive design. Additionally, there is a ‘<style>‘ tag where custom CSS styles can be added.

4. **Navigation Bar**

   The template includes a responsive navigation bar using the Bootstrap framework. It consists of a dark-colored navbar with a brand logo and a collapsible menu. The menu items are defined using an unordered list ‘<ul>‘ with individual items as ‘<li>‘ elements. Each menu item is a hyperlink ‘<a>‘ with a corresponding label.

4.2 register_or_login.html

This is a template file that extends the ’base.html’ template. It is used for the registration or login page of the Clinic System website. It includes form fields for users to either register or login to the system.

1. **Content Structure**

   The file begins by extending the ’base.html’ template, which provides the overall structure and styling for the page.

2. **Content Block**

   The content of this file is enclosed within a ‘% block content %...% endblock %‘ statement. This allows the specific content of this page to be inserted into the corresponding section of the base template.

3. **Form Structure**

   Inside the content block, there is a container ‘ ‘ that holds the form for registration or login. The form is centered on the page using Bootstrap’s ‘justify-content-center‘ class.

4. **Messages**

   Conditional statements are used to display error messages and success messages if they exist. The error message is displayed in an alert box with a red background, while the success message is displayed in an alert box with a green background.

5. **Form Fields**

   The form includes the following fields:

   - **Action:** A dropdown ‘<select>‘ field with options for ’Login’ and ’Register’. The selected value determines the action to be performed.
   - **Phone Number:** A text ‘<input>‘ field for the user’s phone number. It is a required field.
   - **Password:** A password ‘<input>‘ field for the user’s password. It is a required field.
   - **Additional Fields for Registration:** These fields are initially hidden and only displayed when the ’Register’ option is selected in the ’Action’ field. These fields include first name, last name, email, and user type.

6. **Form Submission**

   The form is submitted using the ’post’ method to the URL specified by the ‘% url ’register_or_login’ %‘ template tag. It includes the necessary CSRF token for security.

4.3 secretary_register.html

This is a template file that extends the ’base.html’ template. It is used for the secretary registration page of the Clinic System website. It includes form fields for the secretary to register their clinic.

1. **Content Structure**

   The file begins by extending the ’base.html’ template, which provides the overall structure and styling for the page.

2. **Content Block**

   The content of this file is enclosed within a ‘% block content %...% endblock %‘ statement. This allows the specific content of this page to be inserted into the corresponding section of the base template.

3. **Form Structure**

   Inside the content block, there is a container ‘ ‘ that holds the form for secretary registration. The form is centered on the page using Bootstrap’s ‘justify-content-center‘ class.

4. **Form Fields**

   The form includes the following fields:

   - **Clinic Name:** A text ‘<input>‘ field for the secretary to enter the clinic name. It is a required field.
   - **Address:** A text ‘<input>‘ field for the secretary to enter the clinic address. It is a required field.
   - **Additional Fields:** The template mentions the inclusion of any additional fields needed for secretary registration, but the specific fields are not provided in the code snippet.

5. **Form Submission**

   The form is submitted using the ’post’ method to the URL specified by the ‘ url ’secretary_register’ ‘ template tag. It includes the necessary CSRF token for security.

6. **Hidden Field**

   The form includes a hidden ‘<input>‘ field for the phone number. Its value is set using the ‘ phone_number ‘ template variable, which is likely provided when rendering this template.

4.4 secretary_page.html

This template file extends the ‘register_or_login.html‘ template and is used for the secretary page of the Clinic System website. It provides functionality for secretaries to perform various actions, such as viewing appointments, updating clinic profile, and adding or selecting doctors.

1. **Content Structure**

   The file begins by extending the ‘register_or_login.html‘ template, which sets up the overall structure and styling for the page.

2. **Content Block**

   The content of this file is enclosed within a ‘% block content %...% endblock %‘ statement. This allows the specific content of this page to be inserted into the corresponding section of the base template.

3. **Appointments Table**

   If there are any appointments available, the template renders a table to display appointment details. The table contains columns for date, time, doctor name, doctor phone number, patient name, and patient phone number. The appointment details are dynamically generated using a loop over the ‘appointments‘ data.

4. **Form Structure**

   The template includes a form for secretary actions. The form uses the ’post’ method and submits to the URL specified by the ‘% url ’secretary_page’ %‘ template tag. It also includes the necessary CSRF token for security.

5. **Action Select Field**

   The form includes a dropdown ‘<select>‘ field named ‘action‘, which allows the secretary to choose an action to perform. The available options are "View Profile," "Add or Select Doctor," "Update Clinic Profile," "View Appointments for Clinic," and "Logout."

6. **Additional Fields**

   Depending on the selected action, additional fields are displayed dynamically using JavaScript. If the "Add or Select Doctor" option is chosen, the form displays fields for the doctor’s first name, last name, and phone number. If the "Update Clinic Profile" option is chosen, the form displays fields for the new clinic name and address. These additional fields are initially hidden and shown based on the selected action using JavaScript.

7. **Hidden Fields**

   The form includes two hidden ‘<input>‘ fields. One field is named ‘phone_number‘ and contains the value of the ‘phone_number‘ template variable, likely provided when rendering this template. The other field is named ‘submitted‘ and is set to "true" to indicate that the form has been submitted.

8. **JavaScript**

   The template includes a JavaScript script that listens for changes in the action select field. Based on the selected action, it shows or hides the corresponding additional fields for doctors or clinics using the ‘display‘ CSS property.

4.5 doctor_page.html

This template file extends the ‘secretary_page.html‘ template and is used for the doctor page of the Clinic System website. It provides functionality for doctors to perform various actions, such as viewing their schedule, updating their profile, adding availability, and editing appointments.

1. **Content Structure**

   The file begins by extending the ‘secretary_page.html‘ template, inheriting its overall structure and styling.

2. **Content Block**

   The content of this file is enclosed within a ‘% block content %...% endblock %‘ statement. This allows the specific content of this page to be inserted into the corresponding section of the base template.

3. **Error and Success Messages**

   If there are any error or success messages to display, they are rendered within ‘ ‘ elements with the corresponding alert classes (‘alert-danger‘ and ‘alert-success‘, respectively).

4. **Schedule Table**

   If there is a schedule available, the template renders a table to display the doctor's schedule. The table contains columns for date, time, clinic, patient name, and patient phone number. The schedule details are dynamically generated using a loop over the ‘appointments‘ data.

5. **Form Structure**

   The template includes a form for doctor actions. The form uses the 'post' method and submits to the URL specified by the ‘% url 'doctor_page' %‘ template tag. It also includes the necessary CSRF token for security.

6. **Action Select Field**

   The form includes a dropdown ‘<select>‘ field named ‘action‘, which allows the doctor to choose an action to perform. The available options are "View schedule for doctor," "Update doctor profile," "Add availability," "Edit appointments for doctor," "Back," and "Logout."

7. **Additional Fields**

   Depending on the selected action, additional fields are displayed dynamically using JavaScript. If the "Update doctor profile" option is chosen, the form displays fields for the new first name, last name, and phone number. If the "Add availability" option is chosen, the form displays fields for the new date

and time. If the "Edit appointments for doctor" option is chosen, the form displays fields for the old date, old time, new date, and new time. These additional fields are initially hidden and shown based on the selected action using JavaScript.

8. **Hidden Fields**

   The form includes several hidden ‘<input>‘ fields. These fields store various values related to the doctor, such as phone number, first name, last name, and a flag indicating that the form has been submitted.

9. **JavaScript**

   The template includes a JavaScript script that listens for changes in the action select field. Based on the selected action, it shows or hides the corresponding additional fields for updating the doctor profile, adding availability, or editing appointments using the ‘display‘ CSS property.