ASSIGNMENT-3 (CSCI 5709)

Rajesh Kumar Gupta Lakshminarayan Gupta
(B00791207)

INTRODUCTION

Brief description of application

“We Donate” is a centralized repository of pre-checked and approved organ donors. The application provides the ability for hospitals to identify such approved donors and connect with them in order to make organ donations possible. This application also provides a platform for people who are willing/interested to donate their organs post their demise. The application’s main purpose is to bridge the gap between organ donors and hospitals as the number of organ donations in Canada are as such very less [1]. The provision of booking appointments by the application so as to check if the donor is medically fit helps in creating a repository of donors which can be used in times of need. The application not only helps both the hospitals and donors connect but it also cuts down on the delay in the donation process as the donors are already approved to make the donation whilst they are alive (in cases like kidney donation).

Features of the application

The intended features of our application are:

- User account management
- Donor management
- Donation history
- File upload system
- Appointment management system
- Hospital management
- Search by organ
- Form-to-PDF conversion
- E-mail system
- Reminder

Out of these features, I plan to develop the User account management and form-to-pdf conversion feature. **(Answer for A3.1)**

Why are we using Django framework as the back-end framework?

Python Django was selected because it provides an impressive admin console from which the database tables can be monitored and dummy records can be created from an attractive interface provided by the framework. This will come in handy while testing and it also follows a MVC design pattern, which is suitable for web development. Python Django also comes with an inbuilt SQLite database for development phase which can later be replaced with a database suitable for production with ease.

APPLICATION DETAILS

Target User Insights (Answer for A3.2.1 – First part)

Target user base

The target users for this web application are donors and hospitals. Users only above the age of 18 can register themselves as donors and their organs will be up for donation only post their demise (in most of the cases). This application is mainly targeted towards the hospitals who perform organ transplantations and the common public who is willing to donate their organs for a noble cause. Anybody in the hospital management can access the web application, provided the hospital has given them the appropriate credentials. This application for time being is restricted to hospitals and donors in Canada.

Why would users prefer our application?

Organ donation is a tedious process as a lot of medical checkups need to be run to ensure proper safety of both the donors and the recipients. Our application would provide an opportunity for both the donors and hospitals to check beforehand the medical condition of the donor for smooth organ donations. Our application would serve the purpose of the hospitals by removing the need to put up last-minute advertisements for organ donations and act as a centralized repository holding a list of pre-approved donors from various parts of Canada. Users who are willing to donate need not look far and beyond their computers for registering themselves as donors. The donors can also book appointments to get themselves checked according to their convenience.

Requirements/Pre-requisites

Users do not need specialized training to use our application as basic knowledge in internet browsing and computers will suffice. Users can use our web application on any of the Android or iOS devices including tabs and mobile phones, and also desktop computers.

User-Centered Design Approach (Answer for 3.2.1 – Second part)

Our application has been carefully designed and tailored according to the needs of the user. For instance, **our application lacks clutter** and makes proper consideration for design elements. The elements in the web pages have been organized and designed in such a manner that the user finds it easier to navigate through the content. The webpages of the application contain "**sky blue**" as the primary color palette as it resonates with **health and peace defining the overall purpose of the project, organ donation [2]**.

The web application is a necessity-based one and so we have taken that into account and kept the design **simple** as the users might get distracted and find it hard to focus on the purpose of the web application. This web application consists of a lot of forms and we have made sure that the forms are the **focal point of the webpage** in which they are present. We have also implemented the "**3-click rule"** so that the users find whatever they need under or on 3 clicks.

Since we do not want the users to waste their time searching/navigating, we have provided a **navigation bar** for effective and easier navigation across all the webpages. All the **buttons are standard in color** across webpages and **forms are of standard size** as we want to illustrate **content hierarchy.** We have also taken into account the **various types of users and various devices** that they might use and for this reason, we have made our webpages **responsive** across devices.

APPLICATION WORKFLOW

In a nutshell, this is the basic workflow of our application:

- The donors and hospitals create an account with the application and log themselves in.
- The donors make a donation request and book an appointment with one of the hospitals for a complete check-up to declare themselves as fit.
- The donation request that the donor made shows up on his homepage and the donor waits for the hospital to approve his appointment.

- The hospital, on the other side, approves/denies the appointment based on the donor's donation request.
- Post a successful appointment, if the donor is medically fit, then the hospital changes the donor's status to "approved" thereby making him an approved donor.
- Whenever a hospital requires a donor, it searches from a pool of approved donors and sends him an email requesting his help.

Interaction Design (Answer for 3.2.2 – First part)

The use-case scenarios described below for the features I've selected will reflect the user persona of this web application – the organ donors and the hospitals. The use-case scenarios will also expand on how each process takes place in the front-end as well as the back-end.

Features that I've selected: User Profile Management and Form-to-PDF Conversion.

Use case 1: A donor is changing his/her phone number and wants to update his/her contact number on the website. (User Profile Management feature)

1. The user visits the website's landing page and selects "donor" as the user type.
2. The user enters the credentials on the front-end.
3. The user clicks on the Login button and,
4. The credentials are sent to the back-end by triggering a **HTTP POST message** for validation**.**
5. The HTTP request is unwrapped and the credentials are then validated with the **SQLite** database used in Django framework. If the user is validated, then the homepage of the donor is returned and rendered on the front-end or else, the same login page is returned with an error message informing the user about the invalid credentials.
6. The user then navigates to the "Account Settings" from the homepage via the navigation bar provided on the top.
7. The user clicks on the "Edit" button provided in the webpage to make the form editable.
8. The user then finally changes his/her contact number and clicks on the "Update" button.
9. This triggers a **HTTP PATCH message** which contains the phone number value to be updated.
10. On the back-end, the HTTP message is unwrapped and by using the **USER_ID** of the account, the corresponding contact number is updated in the SQLite database.

Use case 2: It's been a while since the donor used his/her account and the donor forgets the password but he/she now wants to login. (User Profile Management feature)

1. The user visits the website's landing page and selects "donor" as the user type.
2. The user clicks on "Forgot password" button.
3. The user makes a REST API call to forgot password view on the back-end and the back-end renders the "Forgot Password" page.
4. The user inputs his/her e-mail id and clicks on "Send".
5. A **HTTP POST** message is triggered and the e-mail id is validated to check if an user with such an e-mail id actually exists. If the user exists, then an e-mail is sent to the obtained e-mail id and the e-mail contains a temporary password. If the user does not exist, then the user is rendered an error message which states the same.
6. The user visits his e-mail and retrieves the temporary password to login inside the web application.

Use case 3: The hospital wants to search for a kidney donor and view his medical documents along with his registration and contact details. (Form-to-PDF feature – this is the only use case)

1. The user visits the website’s landing page and selects “hospital” as the user type.
2. The user enters the credentials on the front-end.
3. The user clicks on the Login button and,
4. The credentials are sent to the back-end by triggering a **HTTP POST message** for validation**.**
5. The HTTP request is unwrapped and the credentials are then validated with the **SQLite** database used in Django framework. If the user is validated, then the homepage of the donor is returned and rendered on the front-end or else, the same login page is returned with an error message informing the user about the invalid credentials.
6. The user then navigates to the “Search for donor” from the homepage via the navigation bar provided on the top.
7. The user inputs kidney as the organ type and clicks on “search” button.
8. A **HTTP GET message** is triggered and a view retrieves all the approved kidney donors from the database and renders it as a HTML table.
9. The user clicks on the “View Details” button of the donor the user is interested in.
10. Another **HTTP GET message** is triggered and the back-end unwraps the message to understand whose details are required.
11. The back-end converts the details given by the donor in the registration form and converts it to PDF and merges it with his medical reports and renders it as a PDF file for the user.
12. The user downloads the PDF file returned to view the details of the donor.

Process and Service Workflow

The process and service workflows for both the features have been explained in terms of task flows which were drawn using DRAW.io [3].

User Profile Management:

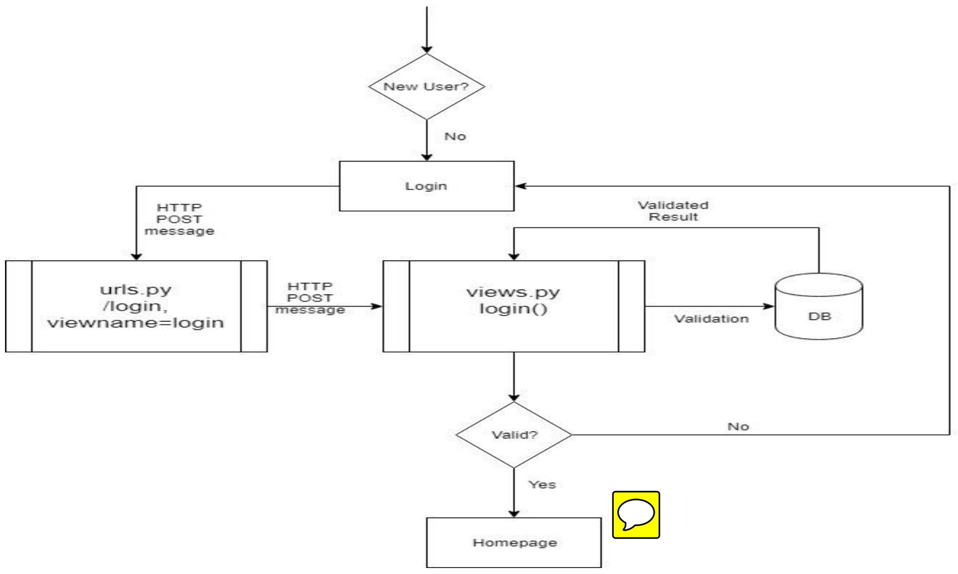

Figure 1: Login Workflow

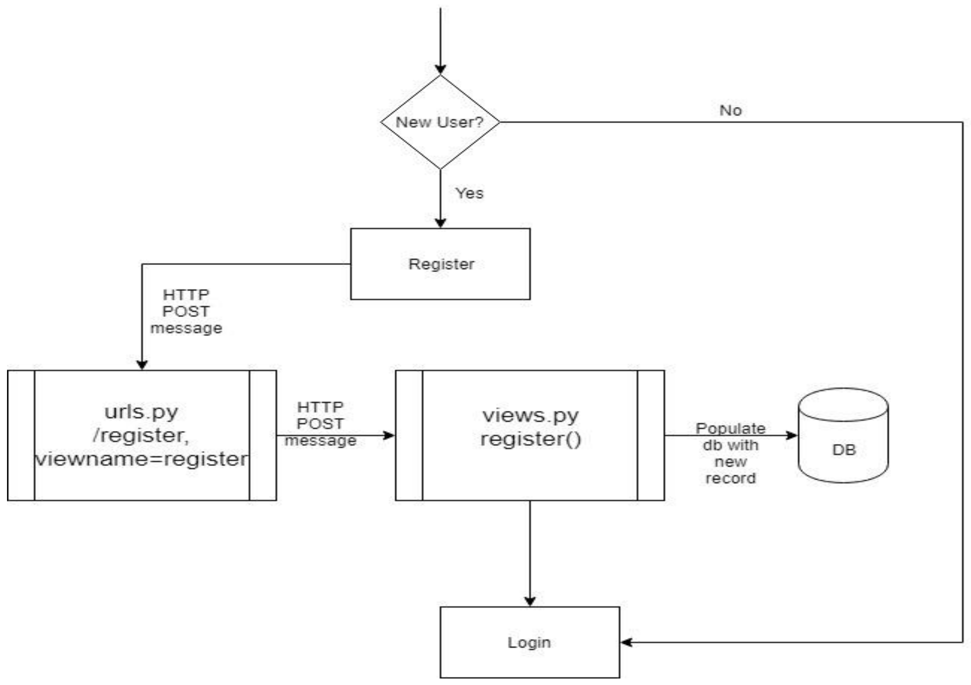

Figure 2: Registration workflow

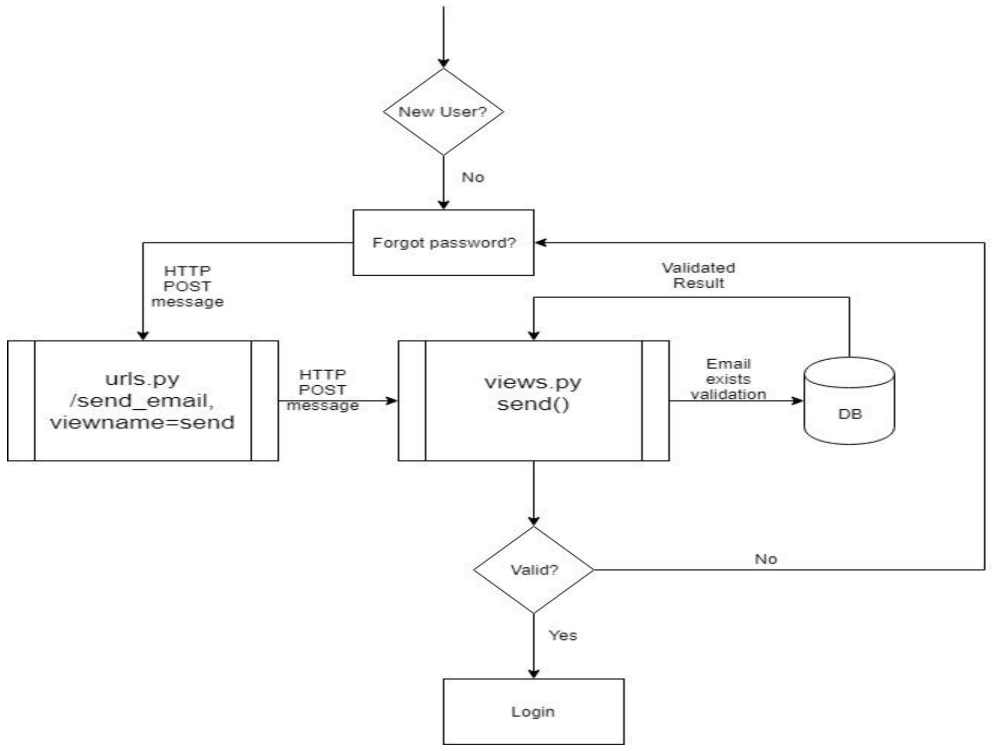

Figure 3: Forgot Password Workflow

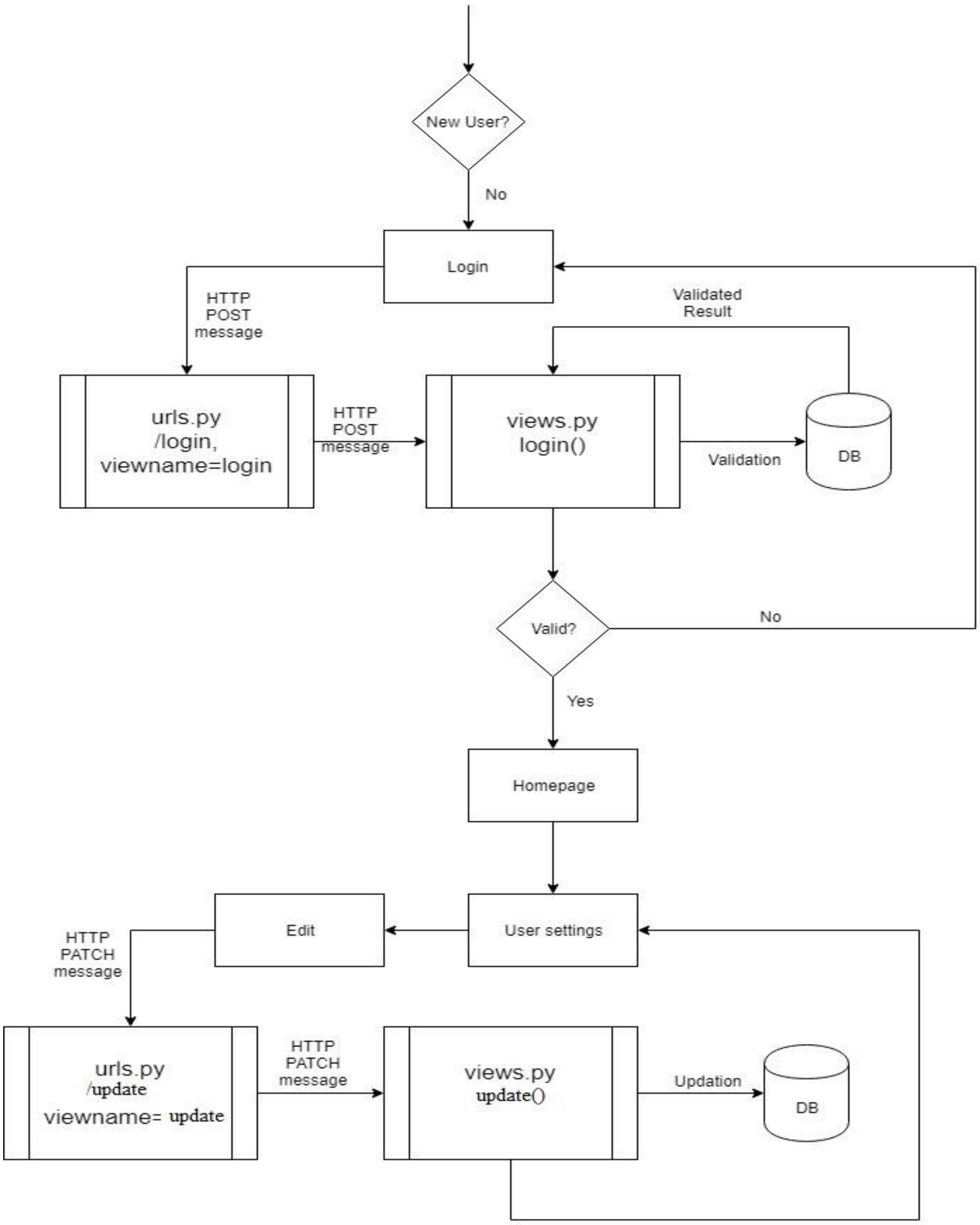

Figure 4: Edit and Update User Profile

Form-to-PDF Conversion:

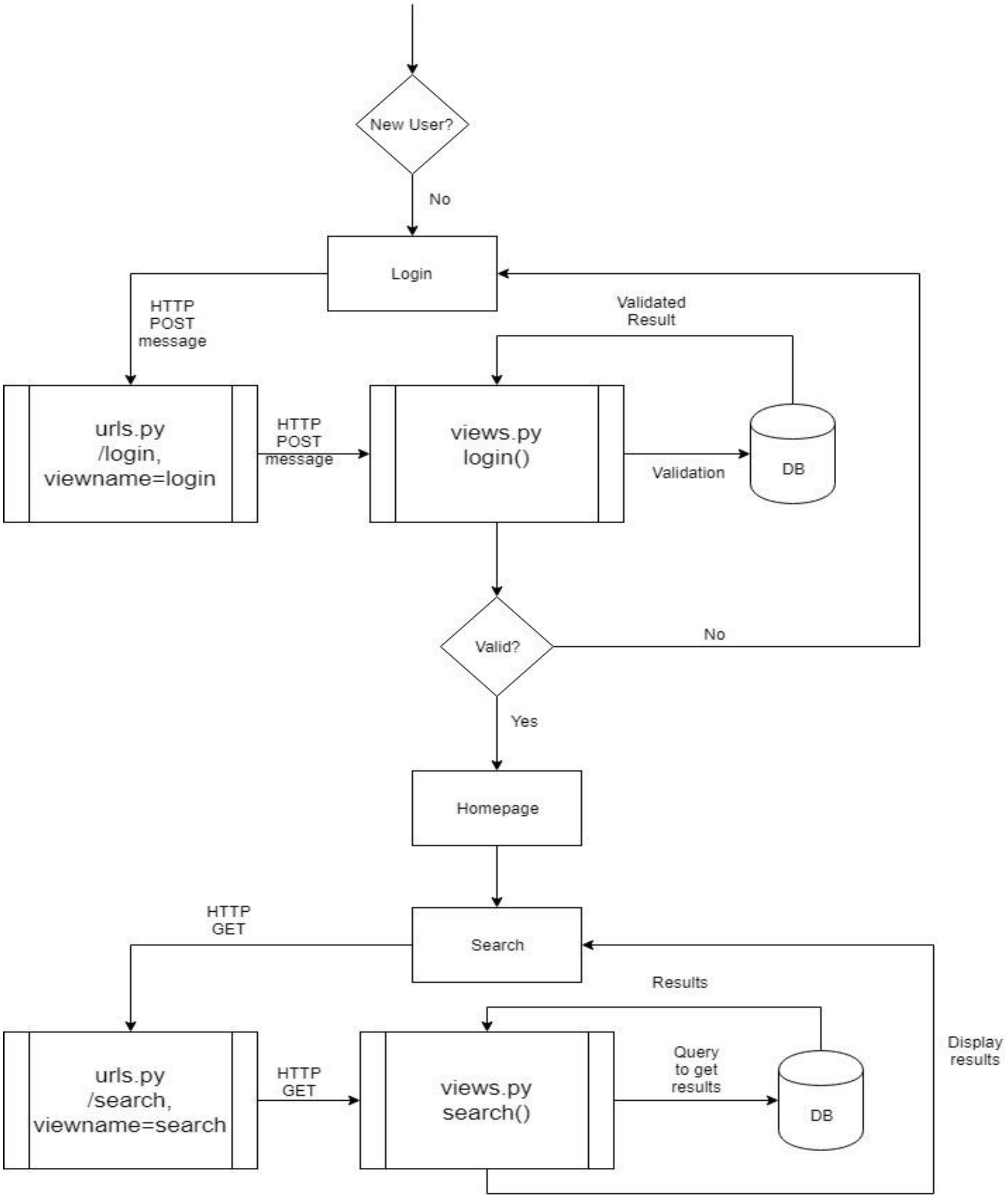

Figure 5: Search Results Workflow

FILE AND FOLDER STRUCTURE

Django mainly consists of a root project folder which in turn comprises the project configurations folder in the same name as the project, the templates and static (front-end files) and the various web app folders defined according to their functionalities.

So in our root project folder “organ-donation”, three main folders are going to be present and they are “templates”,” donor-app”, and “hospitals-app”.

Further folder structure is described below:

- organ-donation/
    - templates/
        - html/
            - all the .html files in the project
    - static/
        - css/
            - all the .css files in the project
        - js/
            - all the .js files in the project
        - img/
            - all the .img files in the project
        - fonts/
            - all the special fonts in the project
    - organ-donation/
        - urls.py [redirecting to urls.py in donor-app and hospital-app by importing them]
        - settings.py [project configuration as to where templates are, db connection]
        - wsgi.py
        - __init__.py
    - donor-app/
        - urls.py [urls related to donor]
        - views.py [views and controller logic related to donor]
        - models.py [tables related to donor]
        - admin.py [to register tables so they are visible on admin interface]
        - __init__.py
    - hospital-app/
        - urls.py [urls related to hospital]
        - views.py [views and controller logic related to hospital]
        - models.py [tables related to hospital]
        - admin.py [to register tables so they are visible on admin interface]
        - __init__.py
- db.sqlite3
- manage.py

REFERENCES

[1] Kidney.ca. (2019). Organ Donation - The Kidney Foundation of Canada | La Fondation canadienne du rein. [online] Available at: https://www.kidney.ca/organ-donation [Accessed 27 Feb. 2019].

[2] Color-Meanings.com. (2019). Blue Color Meaning - The Color Blue. [online] Available at: https://www.color-meanings.com/blue-color-meaning-the-color-blue/ [Accessed 27 Feb. 2019].

[3] Draw.io. (2019). *Flowchart Maker & Online Diagram Software*. [online] Available at: https://www.draw.io/ [Accessed 27 Feb. 2019].