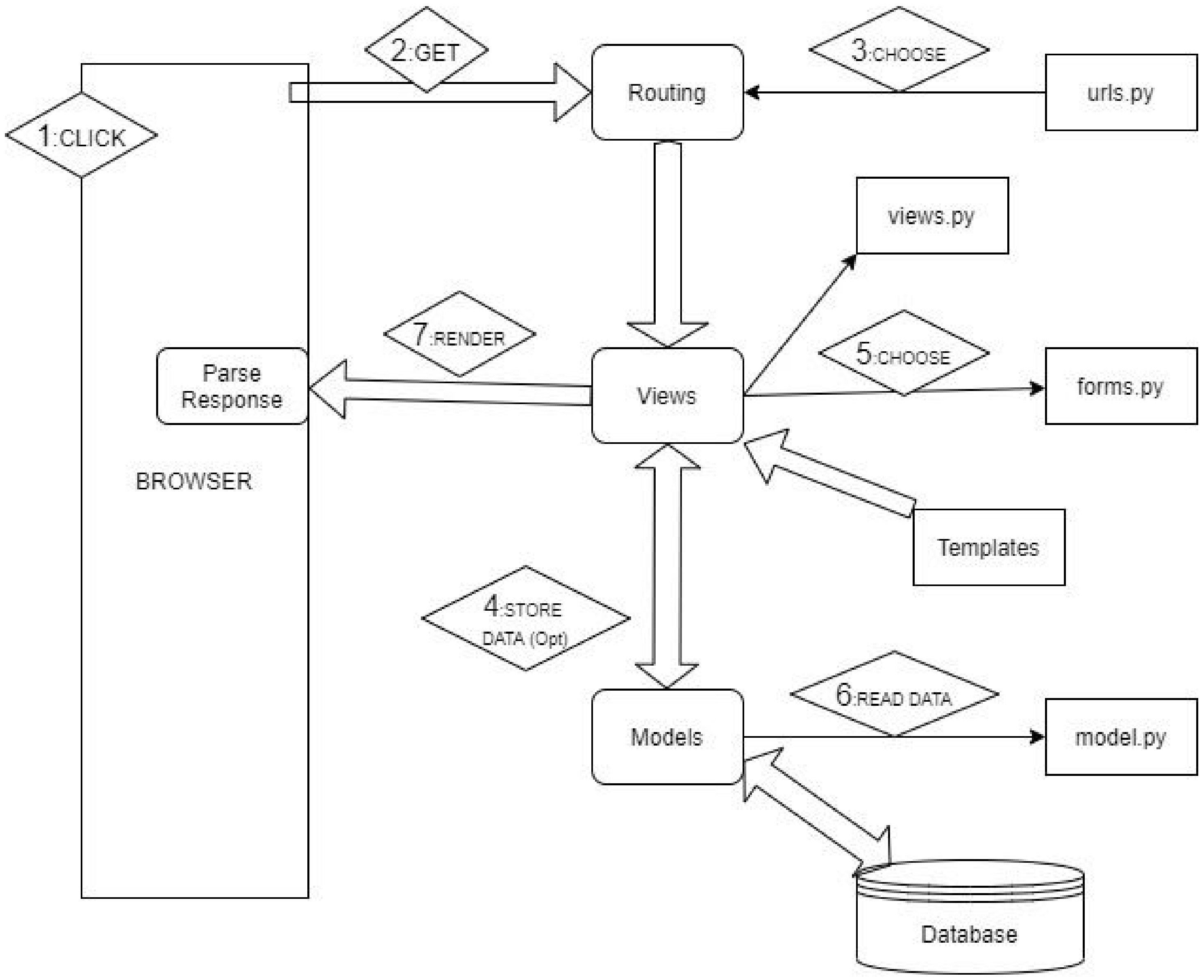
Low Level Design : Backend

Team: Backend

Flow of the Code:

This is a general overview of how a Django website runs.



Login:

1. Client sends request for url <website_url>/login
2. urls.py(location : DjangoProject/urls.py) will call in inbuilt Django Auth view for login
3. Login view will access login form from Django Auth library
4. Since there is no POST request initially, the view renders the login.html (location : DjangoProject/users/templates/users/login.html) page with the form sent to login.html as context.
5. Now the user can see the login page.
6. If the user submits the form the following steps are executed.
7. The POST request is sent to the same url (<website_url>/login)
8. login view will now access this POST request an the formats into a query.
9. To execute this query the user model from Django.contrib library is called.
10. Through *Models*, it accesses Django's database (db.sqlite3) to verify the entered credentials by the user.
11. Then, if verified, the user’s account is activated (logged in) and the user is redirected to the home page.

Signup:

1. Client sends request for url <website_url>/signup
2. urls.py(location : DjangoProject/urls.py) will call in custom signup view (location : DjangoProject/users/views.py) .
3. signup view will accesses custom signup form (Name : UserLoginForm, Location : DjangoProject/users/forms.py)
4. Since there is no POST request initially, the view renders the signup.html (location :

DjangoProject/users/templates/users/signup.html) page with the form sent to signup.html as context.

5. Now the user can see the signup page.
6. If the user submits the form the following steps are executed.
7. The POST request is sent to the same url (<website_url>/signup)
8. signup view will now access this POST request and check whether the form is valid or not (valid means correct according to rules set in UserLoginForm form).
9. if the form is not valid then we send a request with an error message and render the same signup page.
10. Now the user can see what error was thrown and then fill the form accordingly.
11. If the submitted form is valid then view saves the form (save means writing to details of new user to database)
12. Through *Models*, the UserLoginForm form to access Django's database (db.sqlite3) for the write query.
13. Now the view redirects to the login page.

Logout:

1. Client sends request for url <website_url>/logout
2. urls.py(location : DjangoProject/urls.py) will call in logout library view (from Django.contrib.auth)
3. this view will logout the user (destroy session) and then render the logout.html (location DjangoProject/users/templates/users/logout.html)

Home page/Search page:

1. Client sends request for url <website_url>
2. urls.py(location : DjangoProject/urls.py) will call in home view (location DjangoProject/mediaApp/views.py)

3. This view will handle all the searching and video playing. (this part is not in our domain | contributors for this part Google Drive , YouTube API, and REST API teams)

Contributors :-
Ashish Kupsad (180020003@iitdh.ac.in)
Biju Amrut (180020004@iitdh.ac.in)
Hemant Reddy (180010023@iitdh.ac.in)
Paritosh Gavali (180010024@iitdh.ac.in)