**Chapter 6**

**Results and Discussions**

In this chapter the results of the study are presented and discussed with reference to the aim of the study, which was to make a transaction with the help of facial recognition and matching. The two sub-aims the first to take images from the user during the registration process and the second to match them during the time of transaction. These aspects were described in the previous chapter that presented the methodology used in the study and implementation for the same.

**6.1 Registration and Login**

This UI is made in ReactJS. React makes it painless to create interactive UIs. We have designed simple views for each state in our application, and React efficiently updates and renders just the right components when our data changes.

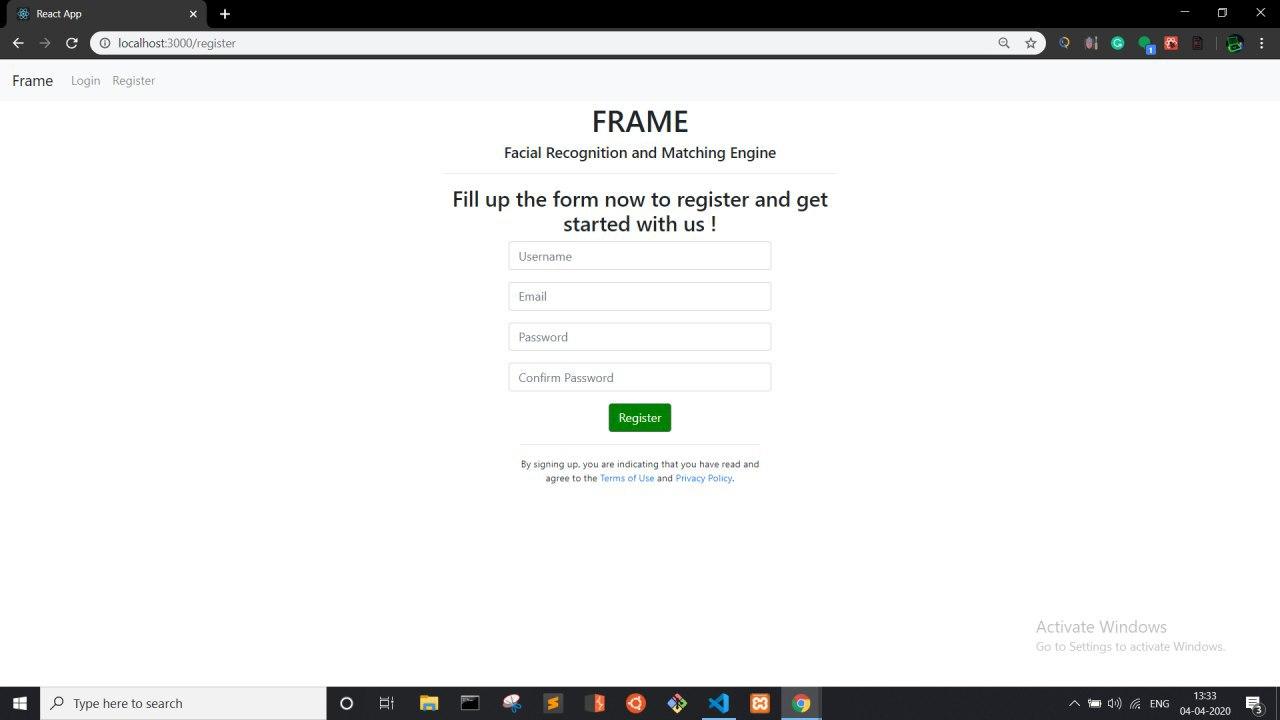


Figure 6.1 Registration Screen

Registration will be done on the above screen, the user is supposed to enter his username, email and password combination. After verification checks such as unique username and similar passwords the registration will be successful if all checks are passed.Upon successful registration, the user will be redirected to the login page.

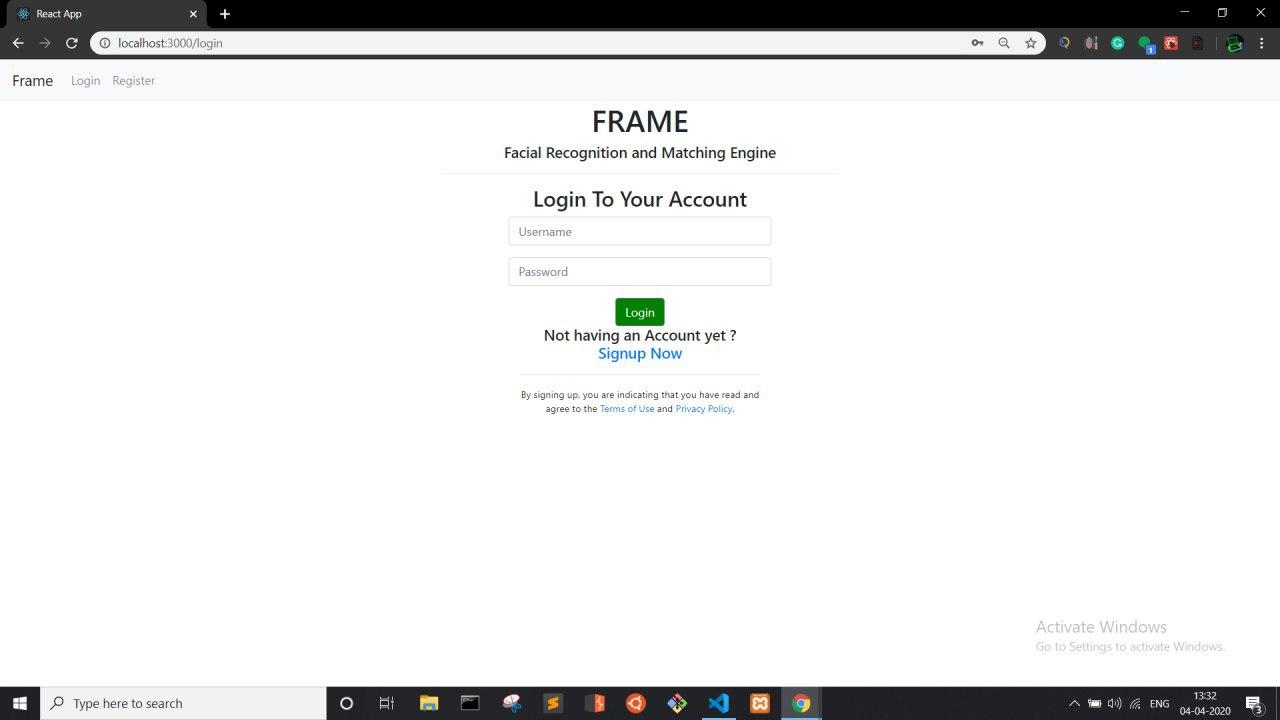


Figure 6.2 Login Screen

The user has to enter his correct combination of username and password which he entered during the time of registration. The information entered in the Registration gets stored in the user info database



Figure 6.3 Working Architecture

We will be using Django server for all the account handling services such as creating a new user, deleting user, changing password, recovering password , etc. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel.

Our Main Web-App will be responsible for User Registration. Here 60 user photos will be taken and stored into the User Image Database. The Django Server will be used for account handling, such as user registration, user login, password updating, data updating, etc. The entire project is based on microservice architecture. Microservices are a software development technique —a variant of the service-oriented architecture (SOA) structural style— that arranges an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight. Every module will be independent. The bottle server will help the modules communicate with each other. The client will hit the api endpoint of bottle server to access the matching engine inside the cab. The web app will also use bottle server api to store images in the database.

The below is the Django admin panel.

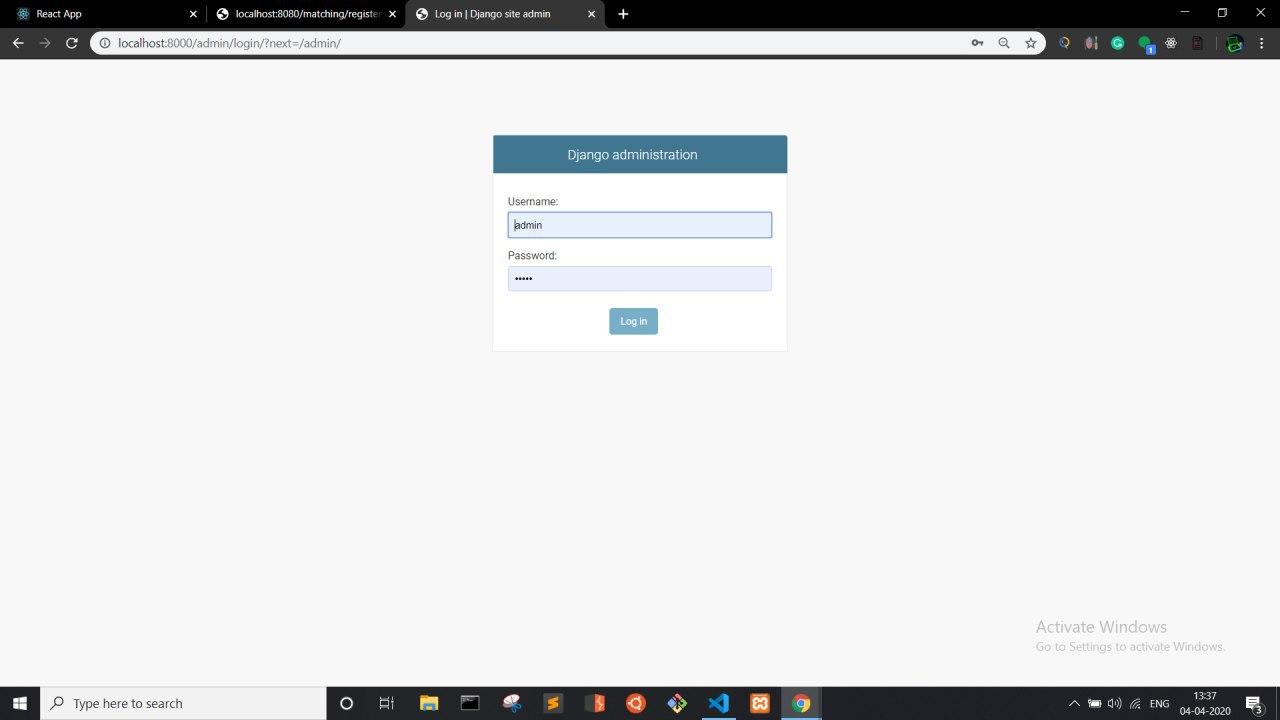


Figure 6.4 Django Administration Login

The above is the login screen for Django Administration. The admin can log in through this.

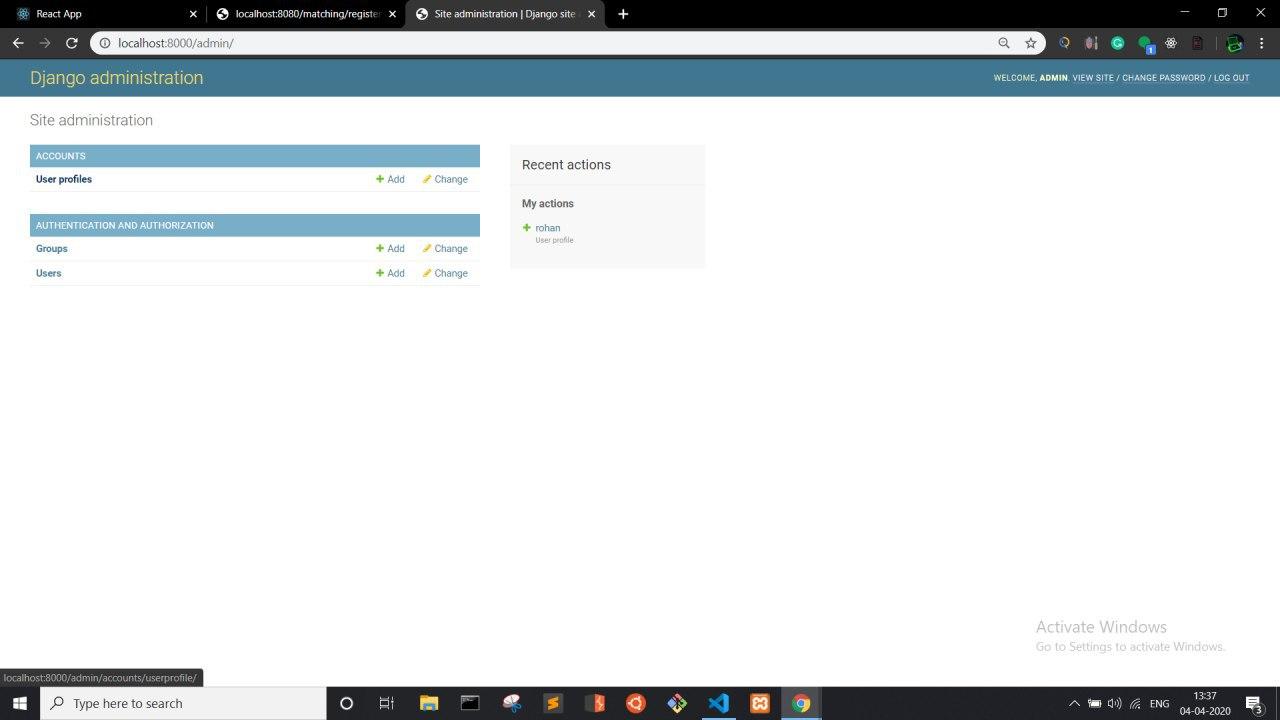


Figure 6.5 Django Admin Dashboard

With the Admin panel the admin has the rights to create, moderate or remove any user.

Upon login the user will be greeted with the below screen from which he can take images for the database. The system takes up to 60 images and stores it in the user image database.

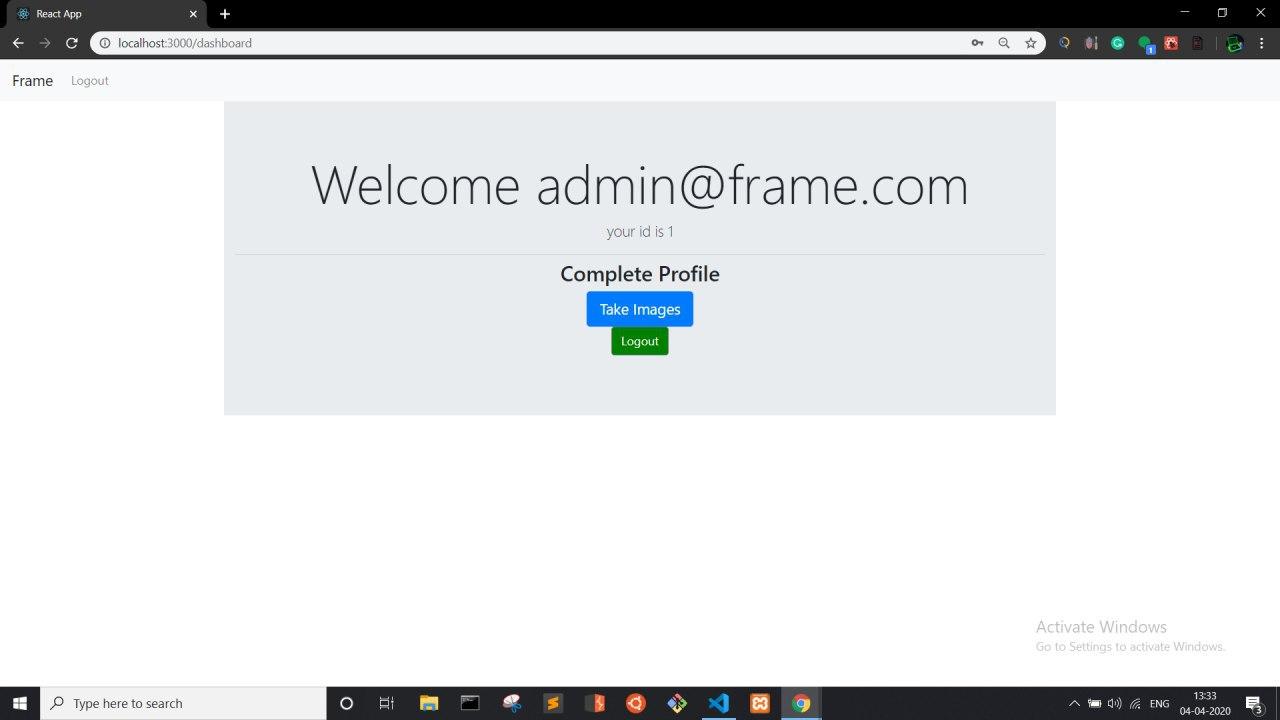


Figure 6.6 Logged in User Dashboard

User will be provided with 2 options, ***Take images*** and ***Logout.*** The take images option will open the front camera and take 60 images of the user.

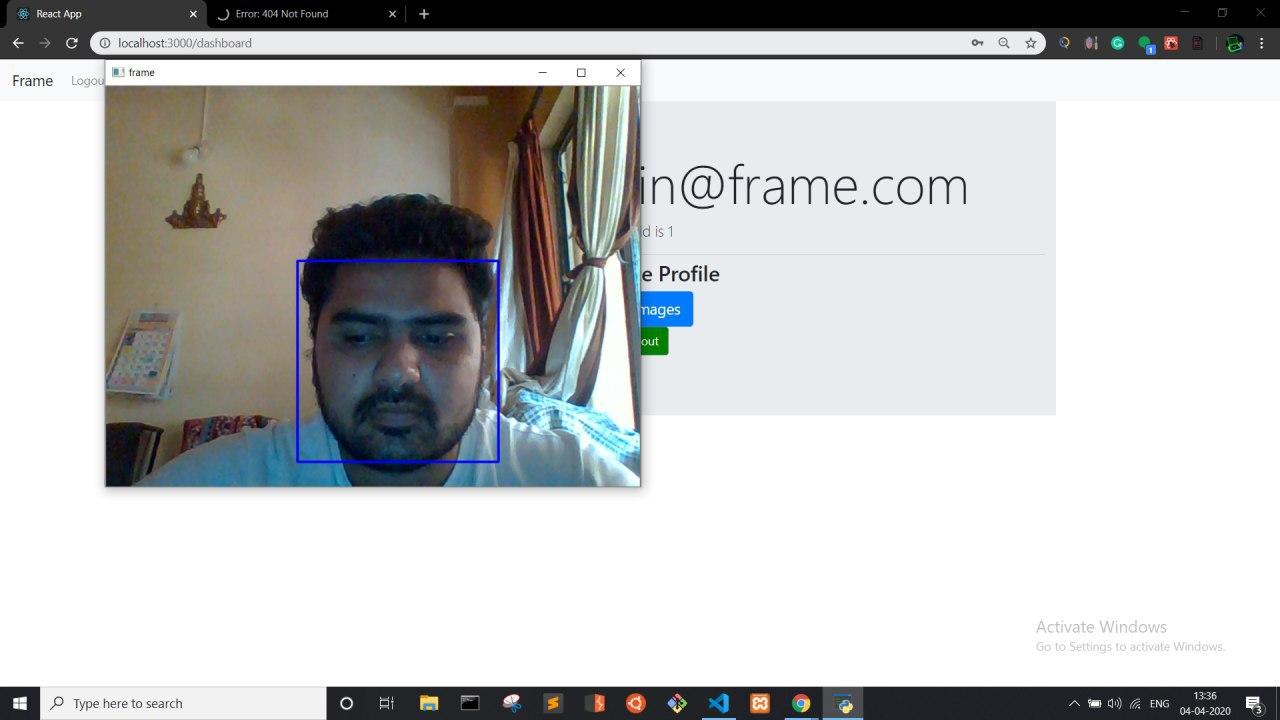


Figure 6.7 Frame Capturing

All the images taken above will be stored in the the user image database via the bottle server.