AJAY KUMAR GARG ENGINEERING COLLEGE

27 DELHI-HAPUR BYPASS ROAD
GHAZIABAD-201009

A
INDUSTRIAL TRAINING REPORT
ON
FULL STACK WEB DEVELOPMENT
(UPLOAD TO WIN APP)

AT
(MOTHERSON SUMI INFOTECH & DESIGNS LIMITED, NOIDA)

Submitted By-
Vishwarupa Basak
(1402713122)
4th Year
Information Technology

Under The Guidance Of

(Mr. Rajat Vijay)

AJAY KUMAR GARG ENGINEERING COLLEGE

27 DELHI-HAPUR BYPASS ROAD
GHAZIABAD-201009

TRAINING CERTIFICATE

This is to certify that **VISHWARUPA BASAK** student of **AJAY KUMAR GARG ENGINEERING COLLEGE** B.Tech Final year **INFORMATION TECHNOLOGY** branch has undergone Industrial Training in **FULL STACK WEB DEVELOPMENT** from **15th June 2017** to **31st July 2017**.

(Wg. Cdr.) R. P. Saw (Retd.)	**Gp. Capt. (Dr.) P.K. Chopra (Retd.) VSM**
Prof. & HoD - IT	Prof. & HoD - T&P

ACKNOWLEDGEMENT

The effort that I have put in my report would not have been possible without the support and help of many individuals. I would like to extend my sincere thanks to all of them.

I extend my gratitude to **Prof. R.P. Saw, HoD IT** for providing with excellent infrastructure and awesome environment that laid potentially strong foundation for my professional life.

I would like to especially thank **Ms. Vidushi**, **Asst. Professor** and **Ms. Shikha Jain, Asst. Professor** for being a source of support, advice and guidance in documentation and standardization of training report.

I would like to thank **T&P Department** for supporting us and providing valuable guidance in selection of best options for industrial training.

I would also like to thank **Mr. Rajat Vijay ,AcadView Instructor**, **ACADVIEW PVT. LTD.** for the positive attitude he showed towards my work and his valuable help and guidance which supported me during my project.

Vishwarupa Basak
1402713122

TABLE OF CONTENTS

CHAPTER 1
COMPANY PROFILE

Acadview is an EdTech startup based out of Delhi NCR. Founding team and advisors include top engineers and managers from Silicon Valley. Himanshu is the founder and CEO at Acadview. He has previously worked with Google for about 10 years in Silicon Valley.

Acadview is solving the problem of unemployability of higher education graduates in the country. We do this by a virtual internship where students are taught how to develop software. The student resume made through acadview has verified skills, verified project portfolio and educational details. As of now, Acadview has trained over 1500+ students in web development and related skills. Founding Team: Himanshu Batra, Varun Jain

Acadview

CHAPTER 2
INTRODUCTION

2.1FULL STACK WEB DEVELOPMENT

A Full-Stack Web Developer is someone who is able to work on both the front-end and back-end portions of an application. Front-end generally refers to the portion of an application the user will see or interact with, and the back-end is the part of the application that handles the logic, database interactions, user authentication, server configuration, etc. Being a Full-Stack Developer doesn’t mean that you have necessarily mastered everything required to work with the front-end or back-end, but it means that you are able to work on both sides and understand what is going on when building an application.Android applications are written in java programming language (An Oops based programming language). Android is available as open source for developers to develop applications which can be further used for selling in android market. There are around 200000 applications developed for android with over 3 billion+ downloads. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model.For software development, Android provides Android SDK(Software development kit).

2.1.1Back-end Development:

•This is the core layer of the any software stack where all the business logic resides. This is ***the hub***. So in-order to build this hub, you would require a *database*, to persist the data, a *programming language/framework* to write logic with and a *server* which entertains users requests.

Database: We may use *MySQL* to persist data but there are many other ones like *MongoDB* (a *NoSQL* database) etc which you can use to persist the data.

Programming Language/Framework: You can use any programming language to write application logic but *Java / PHP / Node / Python* are in wide use across the industry. I use *Laravel* (which is a *PHP* based framework to write all of my applications' backends). *It is my personal favorite*. I've used *Django*(a *Python* based framework) in this major project.

2.1.2Front-end Development: This is the UI part which users of your applications can see and interact with. Basically there are three basic building blocks to this i.e *HTML*, *CSS* and J*AVASCRIPT*. So, as a full-stack developer you are required to know how to design/fix/troubleshoot simple webpages and make them interactive using *JS*. To build web enabled front-ends, it would be really helpful if you know some front-end frameworks like *jQuery*, *Polymer Twitter Bootstrap*, *Foundation* or *Angular* to quickly spin up new web apps. There are many full stack developers who know at-least one major mobile platform like *Android* or *iOS*. If you want to be one of those, you additionally need to learn *Swift* for building *iOS*mobile clients or *XML*/*Java* for *Android* clients. I am one of them, an *Android*developer to be specific.
This is an absolute minimum for building a full fledged web/mobile app on your own and with this knowledge, you can call yourself a *beginner* full-stack developer. Though there a few things you need to learn additionally, in-order to be a proper, industry-level full-stack developer. Some of those things, according to me, are as follows:

2.1.3Other Requirements-

Caches: You must be aware of what*Redis* or *Memcached* are and how these can be used to solve latency problems. I don't care if you've used any of these in the past or not. You just have to be aware that these kind of things exist.

RESTful APIs: You should know the concepts of REST or SOAP and how to design REST/SOAP compatible APIs for your web or mobile clients.

Cookies/Sessions/Authentication: You must be aware of how authentication works in real world and what methods are prevalent in the industry, to protect data. You should also have a hands-on experience of building a service where users have to log in. I am not saying that you should code/build an authentication system on your own (it's best not to code you own, in-fact) but you should have a knowledge of setting it up in your tech stack, be it Java/J2EE or Django or Laravel or Javascript.

Secure HTTP/SSL: As the world is moving towards increased security, it would be very helpful for your web services to be fully secure and for that you would require the knowledge of SSL. You at-least need to know, what it is and how it works. If you know how to set it up, it is an additional advantage.

Cronjobs/Shell scripting: As a full-stack developer you sometimes have to write maintenance scripts for database backup or server cleanup. I write mine in BASH but there other ones like KSH, CSH etc.

Git/Version control: It is equally important to protect the code you write. So you need to know how to keep the code in control using version control systems and how to make use of it in-case of any mishap.

Ajax: You must know how to make AJAX calls to the back-end using *jQuery*or vanilla *JS* or using the in-built libraries of the framework of your choice.

Vagrant etc: There are these new things called *pre-configured development environments* which you can download and start using without installing anything extra plus these environments can be discarded at will. Personally, I don't use it but many full-stack developers are actively using these systems.

CHAPTER 3
PROJECT UNDERTAKEN

UPLOAD-TO-WIN APP

3.1 App Description:

Upload-to-win is basically an app in which the user can login, signup and upload/post their desired images using indivisual id's. They will then be awarded points if they post a certain picture.Image processing has been done using the Clarifai App which gives result based on images of any particular brand name.

3.2 Project Description:

The app has been mainly built using Django framework.The basic tempplates have been built using HTML, CSS, Bootstrap.

An AI App by Clarifai has been used to determine the probability of the presence of a particular Brand Image.

The app has been basically made like the popular app Instagram and by the name **Instaclone.**

CHAPTER 4
SNAPSHOTS OF PROJECT CONSTITUENTS

The following series of images describe various phases of the project's front end, which are important from user point of view.

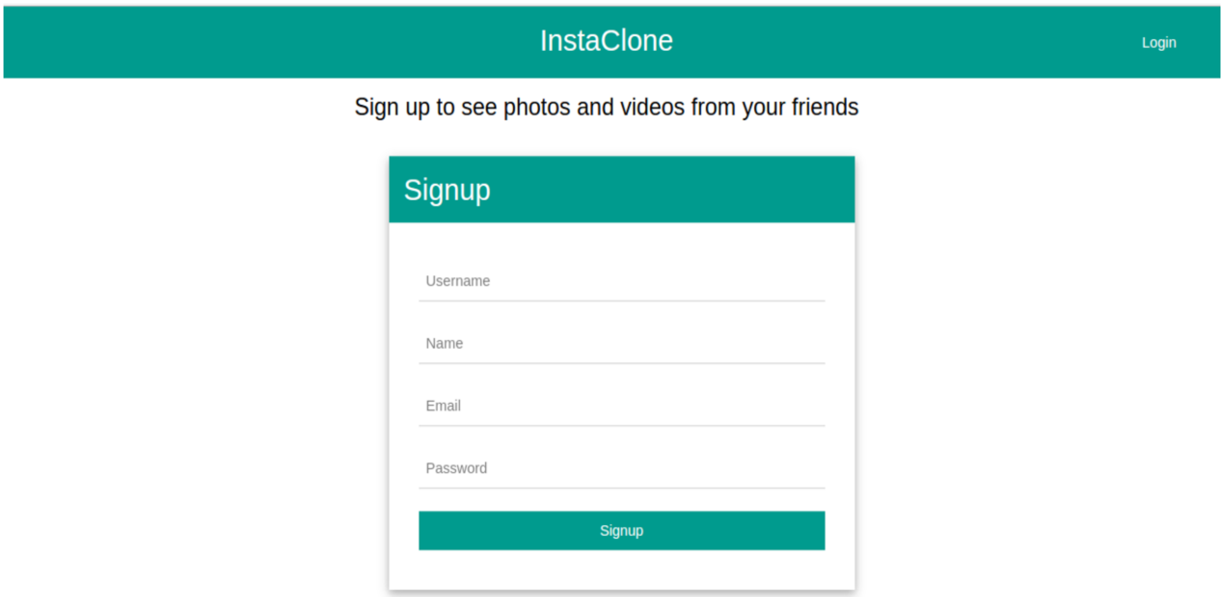


Fig. 4.1: SignUp screen

The SignUp page page for users to Sign Up.

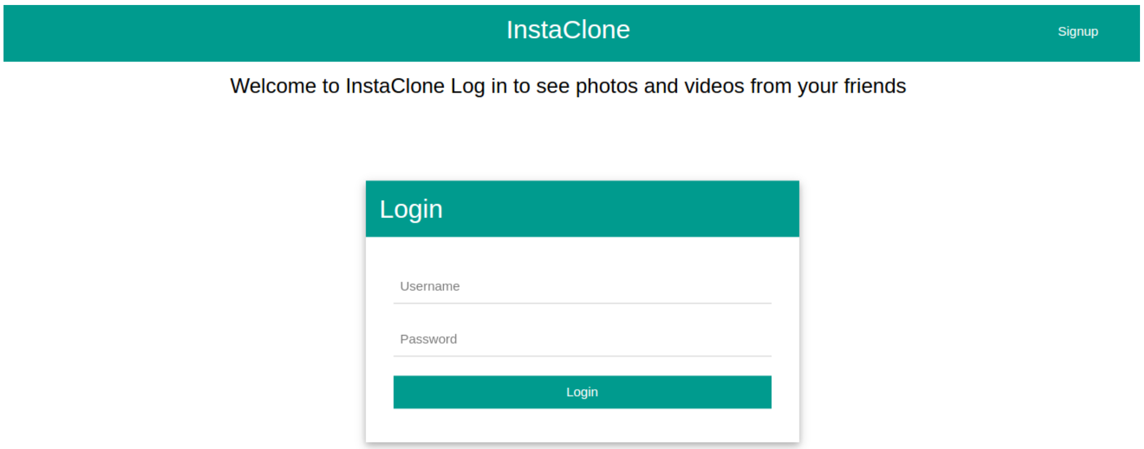
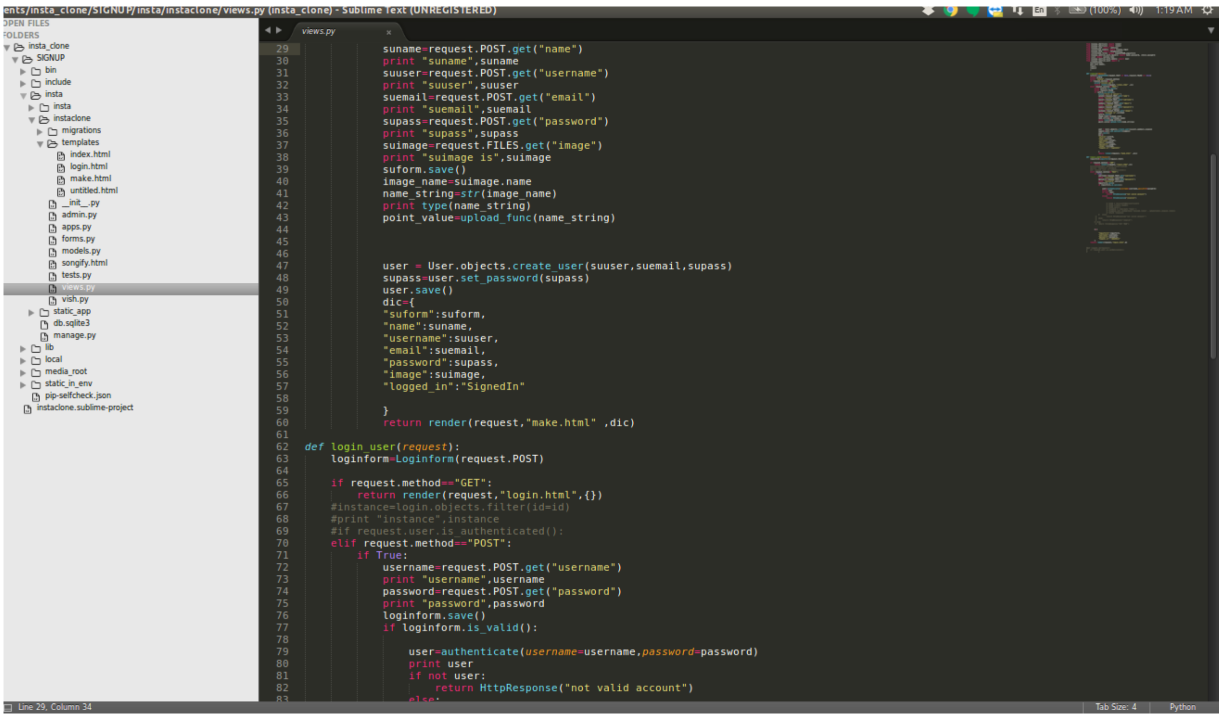

InstaClone
Signup
Welcome to InstaClone Log in to see photos and videos from your friends
Login
Username
Password
Login

Fig. 4.2: Login

This is the basic Login Page.


ents/insta_clone/SIGNUP/insta/instaclone/views.py (insta_clone) - Sublime Text (UNREGISTERED)
views.py
Line 29, Column 34
Tab Size: 4
Python

```python
            suname=request.POST.get("name")
            print "suname",suname
            suuser=request.POST.get("username")
            print "suuser",suuser
            suemail=request.POST.get("email")
            print "suemail",suemail
            supass=request.POST.get("password")
            print "supass",supass
            suimage=request.FILES.get("image")
            print "suimage is",suimage
            suform.save()
            image_name=suimage.name
            name_string=str(image_name)
            print type(name_string)
            point_value=upload_func(name_string)



            user = User.objects.create_user(suuser,suemail,supass)
            supass=user.set_password(supass)
            user.save()
            dic={
            "suform":suform,
            "name":suname,
            "username":suuser,
            "email":suemail,
            "password":supass,
            "image":suimage,
            "logged_in":"SignedIn"

            }
            return render(request,"make.html" ,dic)

def login_user(request):
    loginform=Loginform(request.POST)

    if request.method=="GET":
        return render(request,"login.html",{})
    #instance=login.objects.filter(id=id)
    #print "instance",instance
    #if request.user.is_authenticated():
    elif request.method=="POST":
        if True:
            username=request.POST.get("username")
            print "username",username
            password=request.POST.get("password")
            print "password",password
            loginform.save()
            if loginform.is_valid():

                user=authenticate(username=username,password=password)
                print user
                if not user:
                    return HttpResponse("not valid account")
                else:
```

Fig. 4.3: A snap of code from the View of Django.

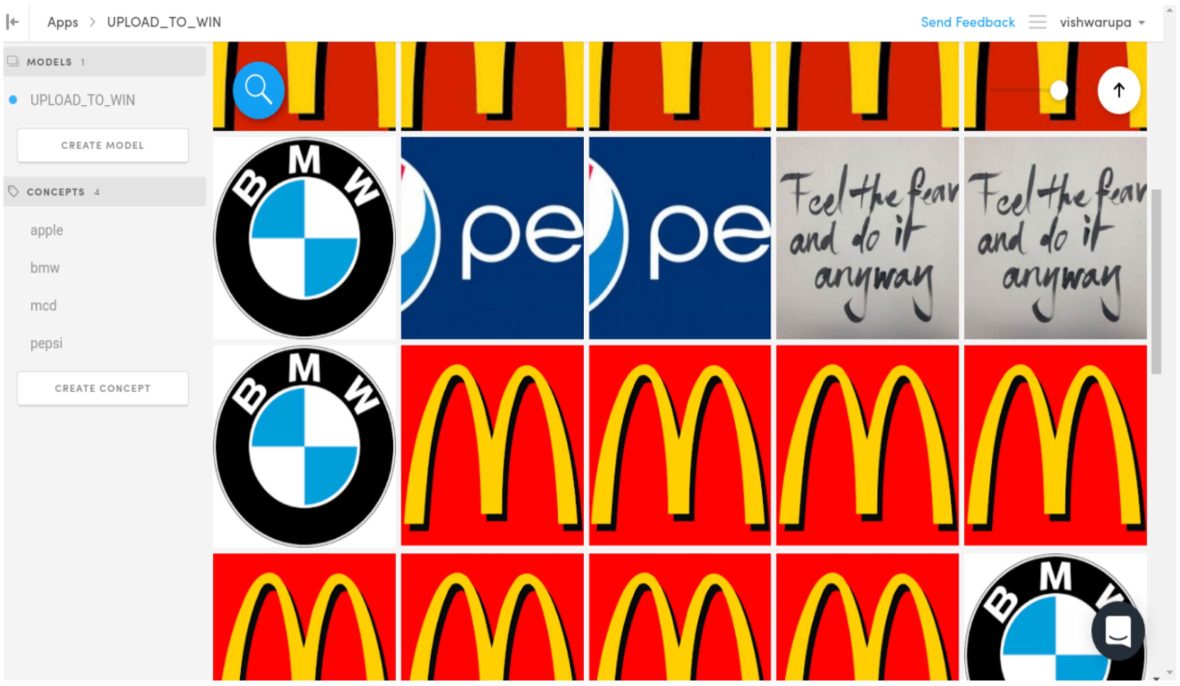


Fig:An image from the Clarifai site where it processes all the brand images.

CHAPTER 5
BASIC DESIGN OF FRONT END

The front-end part consist of number of templates-

- A basic html page-

  serving as the base file-consists of mainly all bootstrap/css/js file links.
- A Login page-

  for the user to login in into the app and start his session.
- A Sign Up page

  Sign up for new users, consisting of name, email and their profile picture upload.
- A Profile section-

  consisting of a welcome note,navigation bar containing settings and post feed option
- Post section-

  consisting of all main posts
- Settings Page-

  consist of only one option to change password.
- Password set up page-

  A field to fill previous password and then the new one.
- Points page and post success.-

  Tells the user about their current score.

CHAPTER 6
FEATURES OF BACK END

The backend was where I majorly worked. The backend mainly comprised of the Django framework which consisted of django-admin project and the application itself.
The main MVC part. My view.py consisted view rendering the respective templates for signup, login, points, settings, etc
My models consisted of LOGIN, SIGNUP, POSTS, PASSWORDS,POINTS,PROFILE PICTURE.
The other files were-

- urls.py-for url routing
- clarifai.py(named)-for hitting the clarifai api and getting back the response for images
- admin.py-the django admin
- forms.py(django forms)
- settings.py
- wsgi.py
- init.py
- etc

CHAPTER 7
MAJOR DESIGN ISSUES

There were several moments during the development of project that posed a threat to the greater integrity of the project. Though solved and gotten rid of, few of them were as noteworthy to mention as the major design issues of the project. Some of them are:

- The biggest issue was getting the appropriate response from the api.The api was processing the image and generating the probability of a certain image to be present.The only solution I could find for it to work was to assume the probability to be above 0.95.Though the results it generated so far are correct but if major test cases be introduced it might end up giving incorrect responses.

- Incorrect point tally

- Session creation(as attempted for the first time, took time to implement).

CHAPTER 8
DESIGN CONSTRAINTS

SOFTWARE CONSTRAINTS	HARDWARE CONSTRAINTS
o Clarifai API o Django framework o Internet Connectivity.	o Minimum RAM requirement of 512mb. o 1ghz processor o Screen size minimum of 320*480.

Table 10.1: Design Constraints of the project

CHAPTER 9
LIMITATIONS OF THE PROJECT

Though readily coded and structured as per basic requirements, the project still has limitations on some grounds. Few of them are as mentioned below:

- The project requires proper internet connectivity.

- Has not been hosted yet.

CHAPTER 10
CONCLUSION

This was my first attempt to build a major app.I did it during my internship. Itt was nothing short of challenge but it made me realise how much there is to learn and made me feel confident about my present skills and learning ability as well. Despite the efforts this project was not a 100% owing to certain limitations which I hope to nullify in future.

This internship as well as the project was a new milestone in my life towards a professional beginning. It helped me learn and grow, know my shortcomings and what I could do to improve. I learnt way more than any other medium applying my knowledge practically .

I hope to embark on such endeavors further in my life with a sheer urge to grow everyday.

CHAPTER 11
FUTURE SCOPE

Without doubt, the project undertaken can be extended to a great scale in future by adding some potential features. Some potential ideas are:

- To create various such social media platform only with intentions of conducting events and competitions amongst peers.
- Allowing users to interact together.

.

CHAPTER 12
REFERENCES

Various web resources proved to be vital for successful completion of the training project. Some of them are mentioned here.

1. https://github.com/guinslym/django-by-example-book

2. https://clarifai.com/developer/account/applications/

3. http://127.0.0.1:8000/admin
4. https://acadview.com/student/dashboard