Gurgaon, India
September, 2021

"Implementing Django Application on AWS"

Project report

Submitted in partial fulfilment of the requirement for
The award of the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Kola Avinash
A50105218021

Under the guidance of

Ms. Pooja Batra Nagpal
Professor

Ms. Sarika Choudhary
professor

Department of Computer Science and Engineering
Amity School of Engineering and Technology
Amity University Haryana

Department of Computer Science and Engineering

Amity School of Engineering and Technology

Declaration

I, ***Kola Avinash, A50105218021***, student of Bachelor of Technology in Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Haryana, hereby declare that I am fully responsible for the information and results provided in this project report titled ***"Implementing Django Application on AWS"*** submitted Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Haryana, Gurgaon for the partial fulfilment of the requirement for the award of the degree of ***Bachelor of Technology in Computer Science and Engineering***. I have taken care in all respects to honour the intellectual property rights and have acknowledged the contributions of others for using them. I further declare that in case of any violation of intellectual property rights or copyrights, I as a candidate will be fully responsible for the same. My supervisor, Head of department and the Institute should not be held for full or partial violation of copyrights if found at any stage of my degree.

Signature(s)

Kola Avinash
A50105218021

Department of Computer Science and Engineering

Amity School of Engineering and Technology

Certificate

This is to certify that the work in the project report entitled ***"Implementing Django Application on AWS"*** by ***Kola Avinash*** bearing ***A50105218021*** is a bonfire record of project work carried out by him under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering in the Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Haryana, Gurgaon. Neither this project nor any part of it has been submitted for any degree or academic award elsewhere.

Date Signature of Supervisor(s)

Supervisor:

(Ms. Pooja Batra Nagpal)

Professor
Computer Science & Engineering
ASET, Amity University, Haryana

Co-Supervisor:

(Ms. Sarika Chaudhary)

Professor
Computer Science & Engineering
ASET, Amity University, Haryana

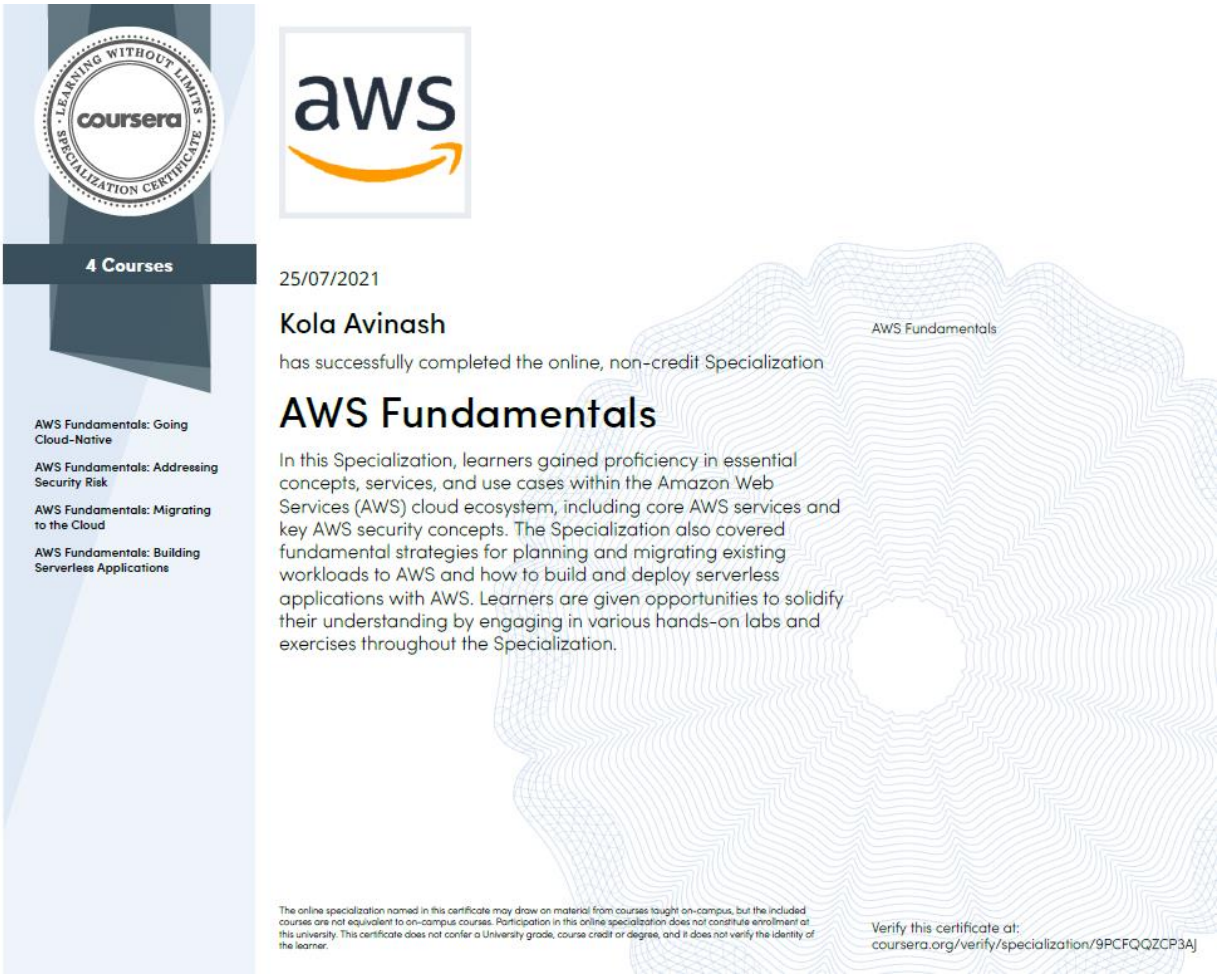
Head
Department of Computer Science & Engineering
Amity School of Engineering and Technology
Amity University Haryana, Gurgaon

ABSTRACT

The main scope of this project is to understand how to best utilize different cloud computing services to implement a Django application. Amazon Web Services offers a broad set of global cloud-based products including compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security, and enterprise applications: on-demand, available in seconds, with pay-as-you-go pricing. From data warehousing to deployment tools, directories to content delivery, over 200 AWS services are available. New services can be provisioned quickly, without the upfront capital expense. This allows enterprises, start-ups, small and medium-sized businesses, and customers in the public sector to access the building blocks they need to respond quickly to changing business requirement.

The Django application that is developed for the hosting in AWS is customer registrations and tracking their orders. It is a data driven application used by the customers and admins to create, delete and update the orders of the customers. The implementation uses a tool called Django Framework which is an excellent open source web application frame work for complex data-driven website development.

CERTIFICATE ISSUED



LIST OF FIGURES

Contents

Chapter 1

INTRODUCTION

"So what is Cloud Computing? Is it utility computing? Is it an application service provider's offering? Is it virtual machines in the sky?" are some of the general questions that arise when we use the term "cloud computing". All of these are correct depending on who you ask. Generally, Cloud Computing can be defined as "Anything that involves delivering hosted services over the Internet".

Cloud computing provides a simple way to access servers, storage, databases and a broad set of application services over the Internet. A cloud services platform such as Amazon Web Services owns and maintains the network-connected hardware required for these application services, while you provision and use what you need via a web application. Using these applications of AWS hosting a django application is the primary goal of this project.

In 2006, Amazon Web Services (AWS) began offering IT infrastructure services to businesses as web services, now commonly known as cloud computing. One of the key benefits of cloud computing is the opportunity to replace upfront capital infrastructure expenses with low variable costs that scale with your business. With the cloud, businesses no longer need to plan for and procure servers and other IT infrastructure weeks or months in advance. Instead, they can instantly spin up hundreds or thousands of servers in minutes and deliver results faster. Today, AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world.

The Django application that is developed for the hosting in AWS is customer registrations and tracking their orders. It is a data driven application used by the customers and admins to create, delete and update the orders of the customers. The implementation uses a tool called Django Framework which is an excellent open source web application frame work for complex data-driven website development.

1.1 Objectives

The main objective of the Project is to understand how to best utilize different cloud computing services to implement a Django application. Creating a django application using python's top most web framework django. Usage of these aws services for making a cost effective application on aws. And also experimenting these services for the effective application development of the software and management of application on day to day basis. Without any physical infrastructure making everything in cloud makes it platform independent.

CHAPTER 2

BACKGROUND OF PROJECT

Developers and architects looking to build new applications in the cloud can simply design the components, processes and workflow for their solution, employ the APIs of the cloud of their choice, and leverage the latest cloud-based best practices1 for design, development, testing and deployment. In choosing to deploy their solutions in a cloud-based infrastructure like Amazon Web Services (AWS), they can take immediate advantage of instant scalability and elasticity, isolated processes, reduced operational effort, on-demand provisioning and automation.

Normally a Django application comprises of the basic database, storing static files, starting and running the server locally. And all these components are totally dependent on the local server. Like database is stored locally and everything that is changed in the database can be stored in local database. And all the static files like a pictures and css styles every non-changeable item that have been included in the project are known as static files and the static files cannot be stored in outside the server so it is stored locally in the system itself in this situation everything becomes a locally dependent so in order to make this independent Usage of aws services makes it visible to web and also make it more convenient to manage and host.

One of the key differentiators of AWS' infrastructure services is its flexibility. It gives businesses the freedom of choice to choose the programming models, languages, operating systems and databases they are already using or familiar with. As a result, many organizations are moving existing applications to the cloud today.

It is true that some applications ("IT assets") currently deployed in company data centers or co-located facilities might not make technical or business sense to move to the cloud or at least not yet. Those assets can continue to stay in place. However, it is strongly believed that there are several assets within an organization that can be moved to the cloud today with minimal effort.

CHAPTER 3

TECHNOLOGIES USED

3.1 AWS

3.1.1 Cloud Computing

Cloud Computing service is one of the most on-demand service that provides various types of services like database, storage applications, and other It resources through a cloud service platform in web based application form so in this format It also provide pay as you go pricing functionality. If you are running applications that share millions of photos to millions of people in mobile devices or other desktop applications where with static files can be hosted in cloud computing services which provides the basic services that help you with flexible and low cost IT resources

Advantages of Cloud Computing

- **Trade capital expense for variable expense** – Instead of having to invest heavily in data centers and servers before you know how you're going to use them, the pay as you go service that have been introduced by this Cloud Computing services which by which only the customer pays as much as they consume and they only consume resources as per the requirement**.**
- **Benefit from massive economies of scale** – By using cloud computing, you can achieve a lower variable cost than you can get on your own. as number of customers that use cloud are drastically increasing, the cloud providers like AWS Provides these cloud services for lower pay as you go prices**.**
- **Stop guessing capacity** – One of the best feature is eliminating the guessing strategy in deployment of the application here one needs to expect the services that are required in a physical infrastructure and many time it is wasted or not sufficient

for eliminating this cloud services provide elasticity in these services where you can use as much as service you need not more or less as there is a large sources of service available in the cloud

- **Increase speed and agility** – Cloud computing also makes the time efficient development possible as the services that are provide are just minutes away rather than taking weeks to just provide. All this time is used effectively by the companies that take the cloud platform.
- **Stop spending money running and maintaining data centers** – Maintaining and running data centers is one of the major time taking and resource taking in today's IT industry**.** to eliminate this loud service provides the services so rather spending time and money on these they can use it effectively on projects, business etc..
- **Go global in minutes** – Easily deploy your application in multiple regions around the world with just a few clicks. That is it provides low latency and high efficient applications to the customers or users at low cost and with high performance.

3.1.2 AWS Services Used

Django Project that is created for this demonstration comprises of the database static files, user information. All these functionalities to implement in the cloud I got to use the following services.

3.1.2.1 AWS Management Console

An intuitive user interface where one can access and manage amazon web services through a web based graphical interface. Quickly get to know the services and the basic knowledge about them.

It also shows us the basic learning resources and the files that makes you practice on aws as a new learner for the new beginners.

The AWS Management Console gives you secure login using your AWS or IAM account credentials. For added security, your login session automatically expires after 12 hours. To resume your session, simply click the "Click login to continue" button and log in again.

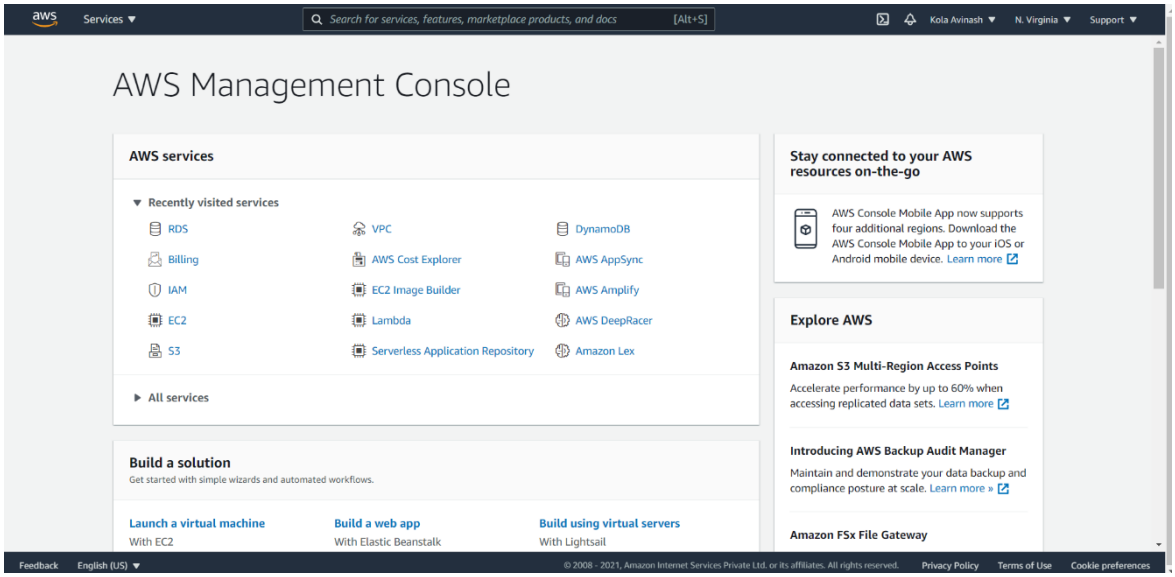


Fig 3.1 AWS Management Console

3.1.2.2 AWS VPC (virtual Private cloud)

(Amazon Virtual Private Cloud) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways.

You can use both IPv4 and IPv6 in your VPC for secure and easy access to resources and applications. You can easily customize the network configuration for your VPC. For example, you can create a publicfacing subnet for your web servers that has access to the Internet, and place your backend systems, such as databases or application servers, in a private-facing subnet with no Internet access. You can leverage multiple layers of security (including security groups and network access control lists) to help control access to EC2 instances in each subnet.

Additionally, you can create a hardware virtual private network (VPN) connection between your corporate data center and your VPC and leverage the AWS Cloud as an extension of your corporate data center.

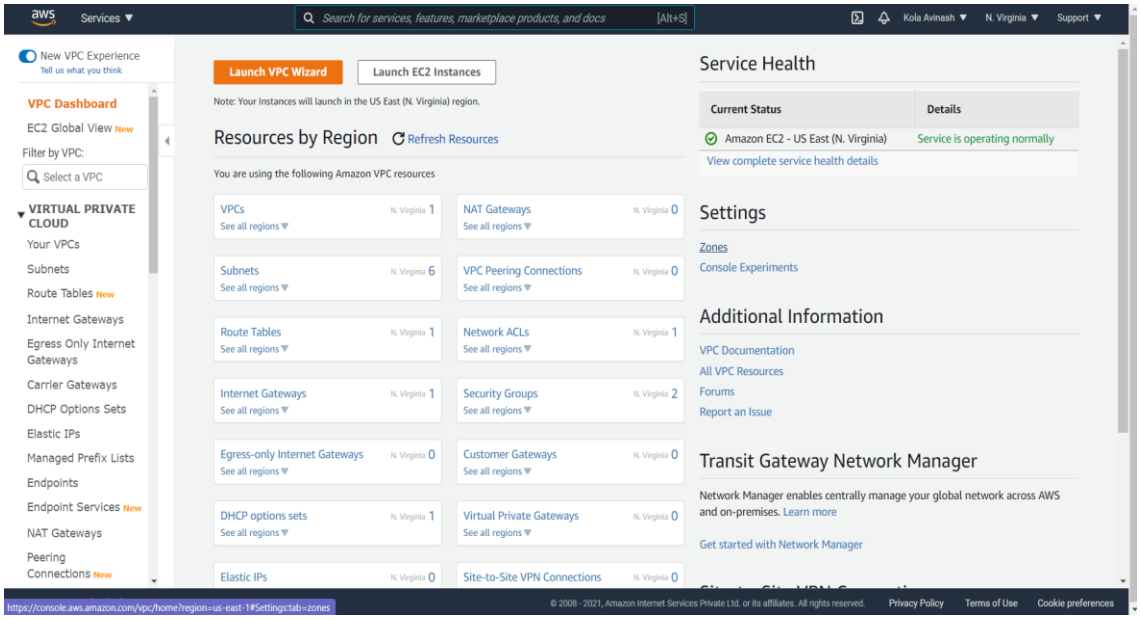


Fig 3.2 AWS VPC (Virtual Private Cloud)

3.1.2.3 Amazon S3

Amazon Simple Storage Service (S3): Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers. Amazon S3 is intentionally built with a minimal feature set.

- Write, read, and delete objects containing from 1 byte to 5 gigabytes of data each. The number of objects you can store is unlimited.

- Each object is stored in a bucket and retrieved via a unique, developer-assigned key.

- A bucket can be stored in one of several Regions. You can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. Amazon S3 is currently available in the US Standard, EU (Ireland), US West (Northern California) and Asia Pacific (Singapore) Regions. The US Standard Region automatically routes requests to facilities in Northern Virginia or the Pacific Northwest using network maps.

- Objects stored in a Region never leave the Region unless you transfer them out. For example, objects stored in the EU (Ireland) Region never leave the EU.

Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users.

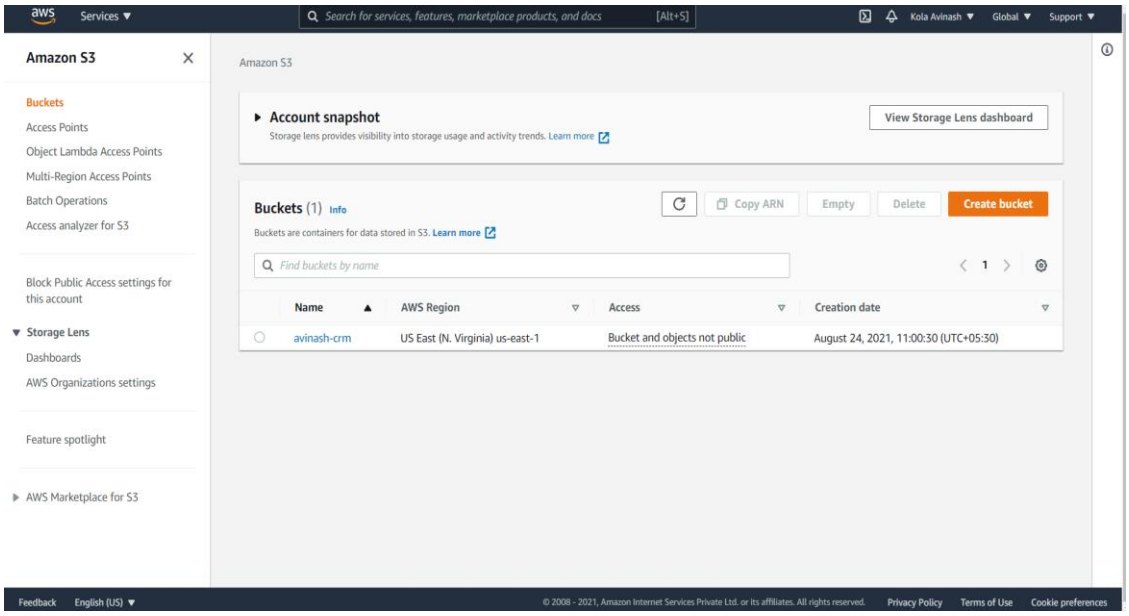


Fig 3.3 Amazon Simple Storage Service

3.1.2.4 AWS Relational Database System

Amazon Relational Database Service Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time consuming administration tasks such as hardware provisioning, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need.

Amazon RDS is available on several database instance types - optimized for memory, performance or I/O - and provides you with six familiar database engines to choose from, including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server. You can use the AWS Database Migration Service to easily migrate or replicate your existing databases to Amazon RDS.

For this Project the RDS service used is the db.t2.micro server which is included in the free tier and comprises of the basic 20 GB storage with 1 GB Ram and the database used is PostgreSQL

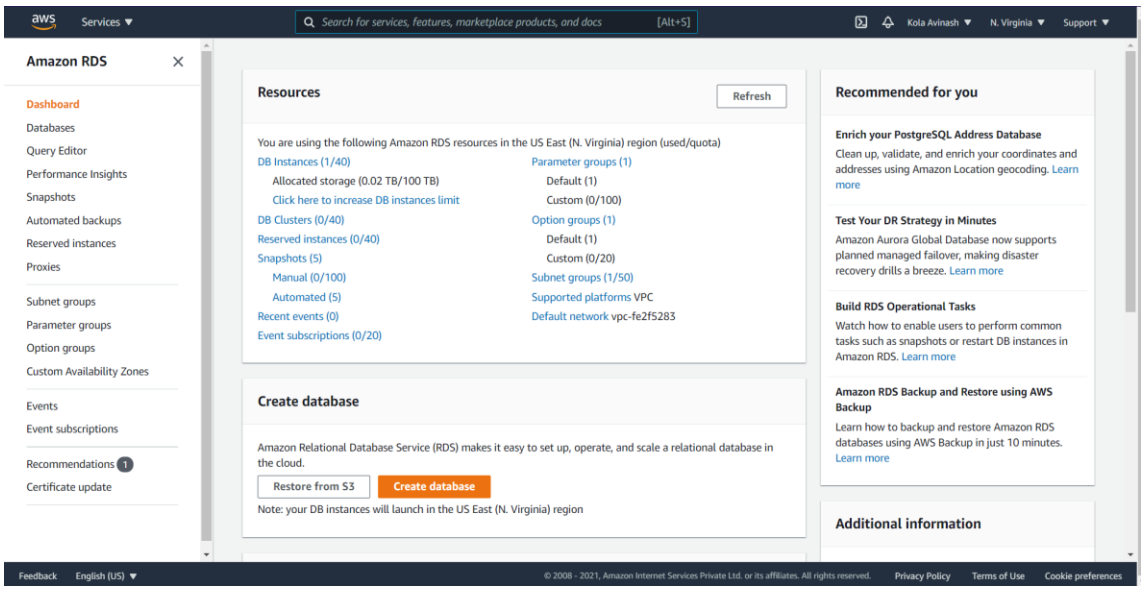


Fig 3.4 AWS Relational Database Service

3.1.2.5 AWS Elastic Cloud Computing

EC2 introduces a new paradigm for web hosting. By allowing developers to scale their number of machines up or down within minutes, it offers the capability to create distributed and scalable applications that run in the cloud. EC2 is flexible, reliable, secure, and most importantly cheap! By only paying for the resources that you actually use, you can bring your multi-server application to market much cheaper than ever before, and maintain an extremely high level of quality and availability.

EC2 is the computing part of the Amazon services. EC2 provides the CPU, memory, operating system and transient storage. EC2 is the equivalent of a barebones PC. You get to pick the amount of RAM you need (from a predefined list of configurations), the amount of transient storage you need (also from a list) and the number of CPUs you need (from a series of compute options). For the operating system, you can choose from various flavors of Linux, Solaris or Microsoft Windows Server. The basis if EC2 is the Amazon Machine Image (AMI). An AMI is a virtual machine with your chosen operating system and applications bundled together. You can create your own AMIs from scratch if you want to. To get started though, Amazon offers hundreds of public AMIs with many operating systems and pre-installed applications.

Elastic IP Addresses:

By default, all Amazon EC2 instances are assigned two IP addresses at launch: a private (RFC 1918) address and a public address that is mapped to the private IP address through Network Address Translation (NAT).

If you use dynamic DNS to map an existing DNS name to a new instance's public IP address, it might take up to 24 hours for the IP address to propagate through the Internet. As a result, new instances might not receive traffic while terminated instances continue to receive requests.

To solve this problem, Amazon EC2 provides elastic IP addresses. Elastic IP addresses are static IP addresses designed for dynamic cloud computing. Elastic IP addresses are associated with your account, not specific instances. Any elastic IP addresses that you associate with your account remain associated with your account until you explicitly release them. Unlike traditional static IP addresses, however, elastic IP addresses allow you to mask instance or Availability Zone failures by rapidly remapping your public IP addresses to any instance in your account.

You can only associate one elastic IP address with one instance at a time. When you associate an elastic IP address with an instance, its current public IP address is released to the Amazon EC2 public IP address pool. If you disassociate an elastic IP address from the instance, the instance is automatically assigned a new public IP address within a few minutes.

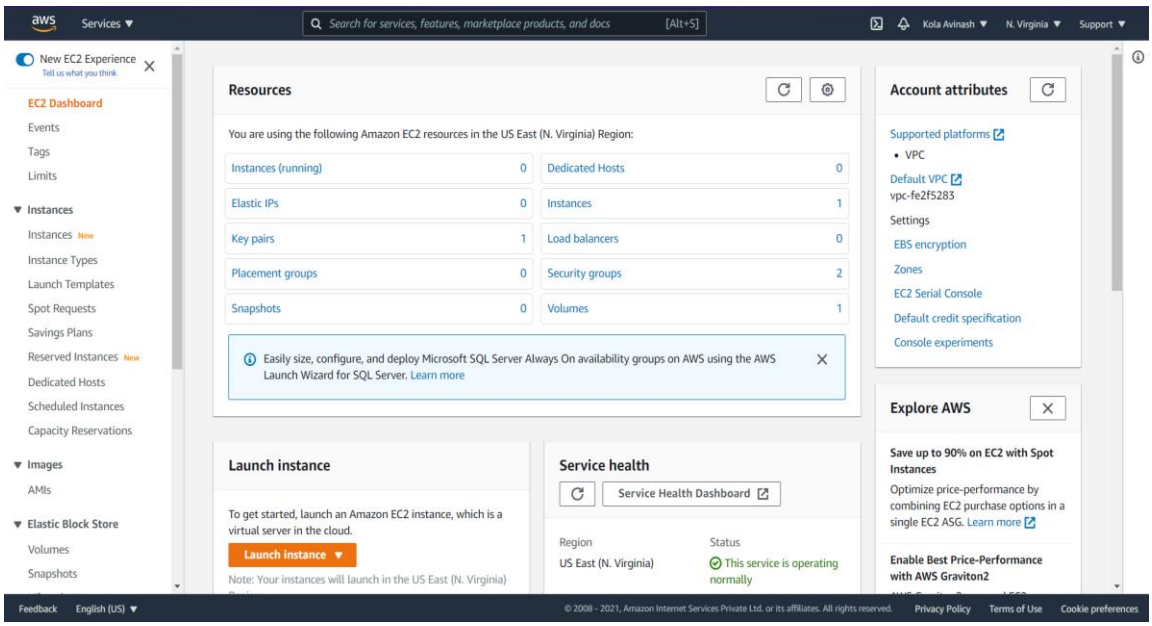


Fig 3.5 AWS Elastic Cloud Computing

3.1.2.6 AWS Identity and Access Management

AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources. IAM allows you to do the following: 69 Overview of Amazon Web Services AWS Whitepaper AWS Key Management Service

- Manage IAM users and their access: You can create users in IAM, assign them individual security credentials (access keys, passwords, and multi-factor authentication devices), or request temporary security credentials to provide users access to AWS services and resources. You can manage permissions in order to control which operations a user can perform.
- Manage IAM roles and their permissions: You can create roles in IAM and manage permissions to control which operations can be performed by the entity, or AWS service, that assumes the role. You can also define which entity is allowed to assume the role.
- Manage federated users and their permissions: You can enable identity federation to allow existing identities (users, groups, and roles) in your enterprise to access the AWS Management Console, call AWS APIs, and access resources, without the need to create an IAM user for each identity

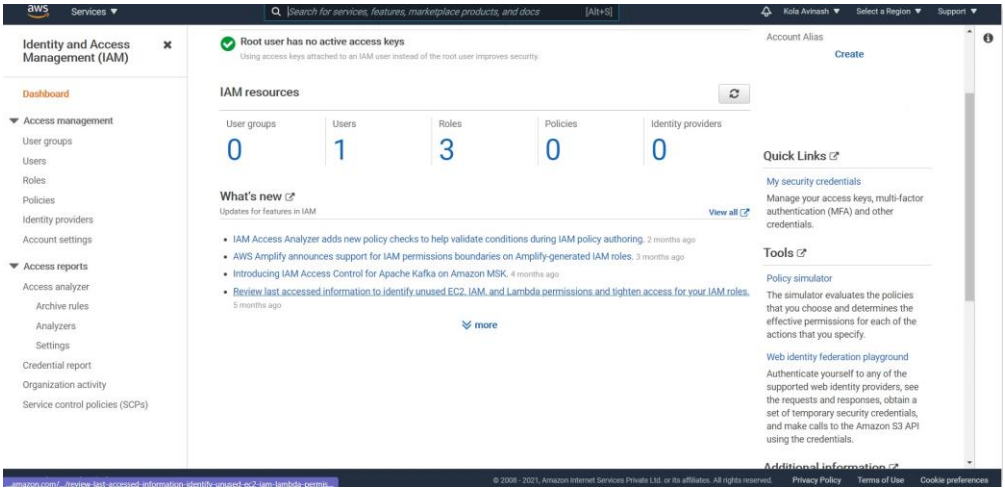


Fig 3.6 AWS Identity and Access Management

3.2 Django

3.2.1 Getting started with Django

Django is a high-level Python Web framework…

A high-level Web framework is software that eases the pain of building dynamic Web sites. It abstracts common problems of Web development and provides shortcuts for frequent programming tasks.

For clarity, a dynamic Web site is one in which pages aren't simply HTML documents sitting on a server's file system somewhere. In a dynamic Web site, rather, each page is generated by a computer program — a so-called "Web application" — that you, the Web developer, create. A Web application may, for instance, retrieve records from a database or take some action based on user input.

A good Web framework addresses these common concerns:

- **It provides a method of mapping requested URLs to code that handles requests.** In other words, it gives you a way of designating which code should execute for which URL. For instance, you could tell the framework, "For URLs that look like /users/joe/, execute code that displays the profile for the user with that username."
- **It makes it easy to display, validate and redisplay HTML forms**. HTML forms are the primary way of getting input data from Web users, so a Web framework had better make it easy to display them and handle the tedious code of form display and redisplay (with errors highlighted).
- **It converts user-submitted input into data structures that can be manipulated conveniently.** For example, the framework could convert HTML form submissions into native data types of the programming language you're using.
- **It helps separate content from presentation via a template system**, so you can change your site's look-and-feel without affecting your content, and vice-versa.
- **It conveniently integrates with storage layers** — such as databases — but doesn't strictly require the use of a database.
- **It lets you work more productively, at a higher level of abstraction**, than if you were coding against, say, HTTP. But it doesn't restrict you from going "down" one level of abstraction when needed.

- **It gets out of your way**, neglecting to leave dirty stains on your application such as URLs that contain ".aspx" or ".php".

Django does all of these things well — and introduces a number of features that raise the bar for what a Web framework should do.

The framework is written in Python, a beautiful, concise, powerful, high-level programming language. To develop a site using Django, you write Python code that uses the Django libraries. Although this book doesn't include a full Python tutorial, it highlights Python features and functionality where appropriate, particularly when code doesn't immediately make sense.

…that encourages rapid development…

Regardless of how many powerful features it has, a Web framework is worthless if it doesn't save you time. Django's philosophy is to do all it can to facilitate hyper-fast development. With Django, you build Web sites in a matter of hours, not days; weeks, not years. This is possible largely thanks to Python itself. Oh, Python, how we love thee, let us count the bullet points:

- Python is an interpreted language, which means there's no need to compile code. Just write your program and execute it. In Web development, this means you can develop code and immediately see results by hitting "reload" in your Web browser.
- Python is dynamically typed, which means you don't have to worry about declaring data types for your variables.
- Python syntax is concise yet expressive, which means it takes less code to accomplish the same task than in other, more verbose, languages such as Java. One line of python usually equals 10 lines of Java. (This has a convenient side benefit: Fewer lines of code means fewer bugs.)
- Python offers powerful introspection and meta-programming features, which make it possible to inspect and add behavior to objects at runtime. Beyond the productivity advantages inherent in Python, Django itself makes every effort to encourage rapid development. Every part of the framework was designed with productivity in mind

3.2.2 The Django template system

The HTML was hard-coded directly in our Python code! This arrangement leads to several problems:

- Obviously, any change to the design of the page would require a change to the Python code. The design of a site tends to change far more frequently than the underlying Python code, so it would be convenient if the frequency of HTML changes were separated from changes to Python code.
- Second, writing backend Python code and designing/coding HTML are two different disciplines, and most professional Web development environments split these responsibilities across separate people (or even separate departments). Designers and HTML/CSS coders shouldn't have to edit Python code to get their job done; they should deal with HTML.
- Similarly, it's most efficient if programmers can work on Python code and designers can work on templates at the same time, rather than one person waiting for the other to finish editing a single file that contains both Python and HTML.

For these reasons, it's much cleaner and more maintainable to separate the design of the page from the Python code itself. We can do this with Django's template system

3.2.3 Interacting with a database: Models

A view is responsible for doing some arbitrary logic, then returning a response. In the example, our arbitrary logic was to calculate the current date and time.

In modern Web applications, the arbitrary logic often involves interacting with a database. Behind the scenes, a database driven Web site connects to a database server, retrieves some data out of it and displays that data, nicely formatted, on a Web page. Or, similarly, the site could provide functionality that lets site visitors populate the database on their own.

Many complex Web sites provide some combination of the two. Amazon.com, for instance, is a great example of a database driven site. Each product page is essentially an extract of Amazon's product database formatted as HTML, and when you post a customer review, it gets inserted into the database of reviews.

Django is very well-suited for making database-driven Web sites, as it comes with easy yet powerful ways of performing database queries using Python.

3.2.4 Views in Django

View is short form of view file. It is a file containing Python function which takes web requests and returns web responses. A response can be HTML content or XML documents or a "404 error" and so on. The logic inside the view function can be arbitrary as long as it returns the desired response. To link the view function with a particular URL we need to use a structure called URLconf which maps URLs to view functions.

Django views are part of the user interface — they usually render the HTML/CSS/JavaScript in your Template files into what you see in your browser when you render a web page. (Note that if you've used other frameworks based on the MVC (Model-View-Controller), do not get confused between Django views and views in the MVC paradigm. Django views roughly correspond to controllers in MVC, and Django templates to views in MVC.)

3.3 Web Technologies

3.3.1 HTML (Hyper Text Markup Language)

HTML was created by Tim Berners-Lee in 1991. The first ever version of HTML was HTML 1.0 but the first standard version was HTML 2.0 which was published in 1999. HTML stands for Hyper Text Markup Language. It is used to design web pages using Markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. Markup language is used to define The text document within tag which defines the structure of web pages.
HTML is a makeup language used to browse the manipulated Text, Image's and other Content to display it on the required format.

Advantages of html:

- HTML is easy to understand and use:

  Most people in the web design and development industry; whether a freelancer or an agency, are familiar with HTML. If your site is made using HTML and you need to take the help of a new designer to update or modify anything on it, it will be much easier to find a designer who is familiar with the language and can make the changes to it for an affordable fee.

- HTML is supported in all browsers:

  HTML is supported by almost all the browsers that are available today. In fact, it is supported by more browsers than any other programming language. As a result, your website will open on almost all the browser around the world if it is created using HTML

- HTML is free:

  One big advantage of HTML is that it does not cost anything. It is free of cost. Since it does not need any plug-ins or software's, you will save a lot of money if you choose to design your site using this language.

- HTML is supported by most web development tools:

  HTML can be used with many web development tools like FrontPage or Dreamweaver. It is supported by almost all of the development tools and hence it will be much easier to create and develop a website using HTML in comparison to other programming languages

- HTML Is most search friendly Language:

  Out of all the programming languages that are available in the industry today, HTML is the most search engine friendly language. It is very easy to create a website using HTML.

HTML VERSION	YEAR
HTML 1.0	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML 5	2014

Fig 3.7 HTML Versions

3.3.2 CSS (Cascading Style Sheet)

Cascading style sheets ,referred as CSS, is a simply designed language intended to simplify the process of making web pages presentable.CSS allows you to apply the styles to webpages. CSS enables you to do this independent of the HTML that makes up each web page.CSS is easy to learn and understood but it provides powerful control over the presentation of an HTML document. CSS describes how html elements are to be displayed on screen, paper, or in other media. CSS saves lot of work .It can control the layout of multiple web pages all at once External style sheets are stored in CSS files

Advantages of CSS:

- CSS SAVES TIME:

  You can write CSS once and reuse same sheet in multiple HTML pages
- EASY MAINTAINENCE:

  To make a global change simply change the style, and all elements in all web pages will be updated automatically.
- OFFLINE BROWSING:

  CSS can store web applications locally with the help of offline Cache using this we can view offline websites
- GLOBAL WEB STANDERDS:

  Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible with future browsers.
- PLATFORM INDEPENDENCE:

  The Script offer consistent platform independence and can support latest browsers as well.

3.3.3 JavaScript

- Java script was developed by Brendan Eich in 1995, which appeared in Net space, a popular browser at that time
- Initially the language is called as live script but latter changed to java script
- There are many programmers who think JavaScript and Java are same but they are very much unrelated .Java is very complex programming language and Java script is just a scripting language.
- The syntax of Java script is most related to c programming language.
- Java script is a lightweight, interpreted, programming language. It is designed for creating network centric applications. It is very easy to implement because it is integrated with html.
- JavaScript helps you create really beautiful and crazy fast websites.
- Due to high demand, there is tons of job growth and high pay for those who know JavaScript with some other related frameworks.

3.3.4 Bootstrap

Bootstrap is a free front-end framework for faster and easier web development. Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, Tables, navigation, modals, image carousels and many other, as well as optional JavaScript Plugins.

Bootstrap also gives you the ability to easily create responsive designs.
Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter, and released as an open Source product in August 2011 on GitHub. In June 2014, Bootstrap was the No.1 project on GitHub!

Advantages of Bootstrap:

- Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap.
- Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops.
- Mobile-first approach: In Bootstrap 3, mobile-first styles are part of the core framework.
- Browser compatibility: Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Edge, Safari, and Opera).
- If you do not want to download and host Bootstrap yourself, you can include it from a CDN (Content Delivery Network).
- Great grid system

3.4 PostgreSQL

PostgreSQL also known as **Postgres**, is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley. In 1996, the project was renamed to PostgreSQL to reflect its support for SQL. After a review in 2007, the development team decided to keep the name PostgreSQL and the alias Postgre.

PostgreSQL features transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures. It is designed to handle a range of workloads, from single machines to data warehouses or Web services with many concurrent users. It is the default database for macOS Server and is also available for Windows, Linux, FreeBSD, and OpenBSD.

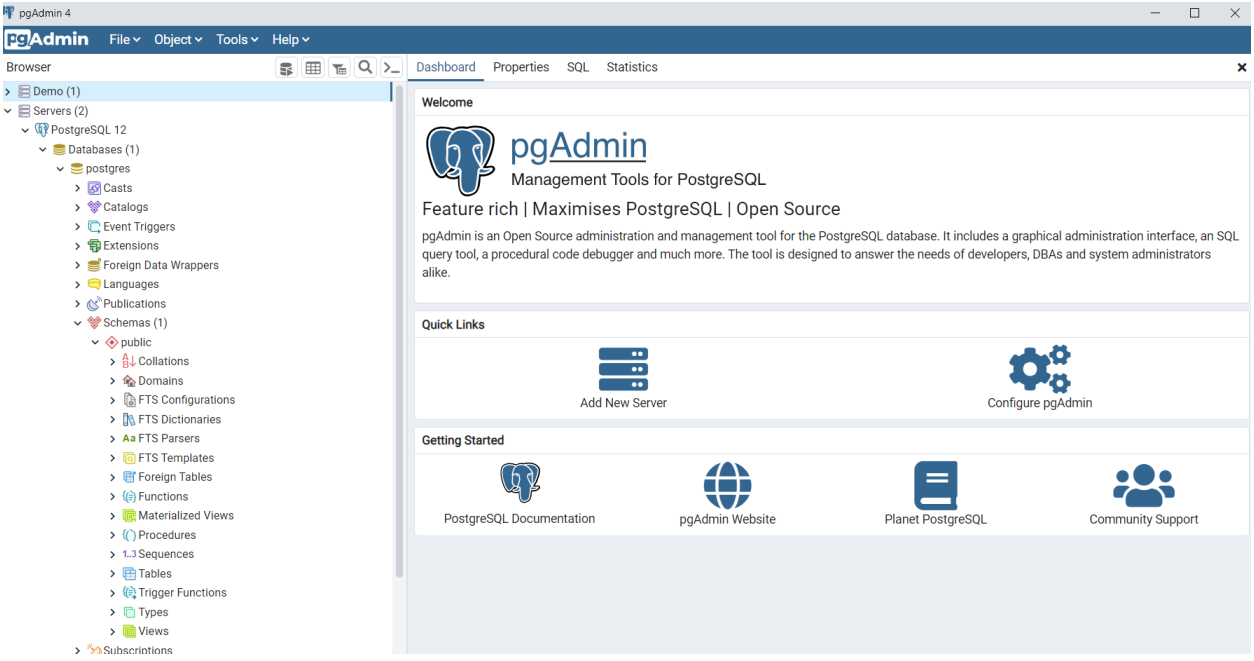


Fig 3.8 PostgreSQL

3.5 Google APIs and Services

Google APIs are application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services. Examples of these include Search, Gmail, Translate or Google Maps. Third-party apps can use these APIs to take advantage of or extend the functionality of the existing services.

The APIs provide functionality like analytics, machine learning as a service (the Prediction API) or access to user data (when permission to read the data is given). Another important example is an embedded Google map on a website, which can be achieved using the Static Maps API Places API or Google Earth API.

This service is used to make the user effectively login into the site using Google account and by calling this the Google Oath API service is triggered and used

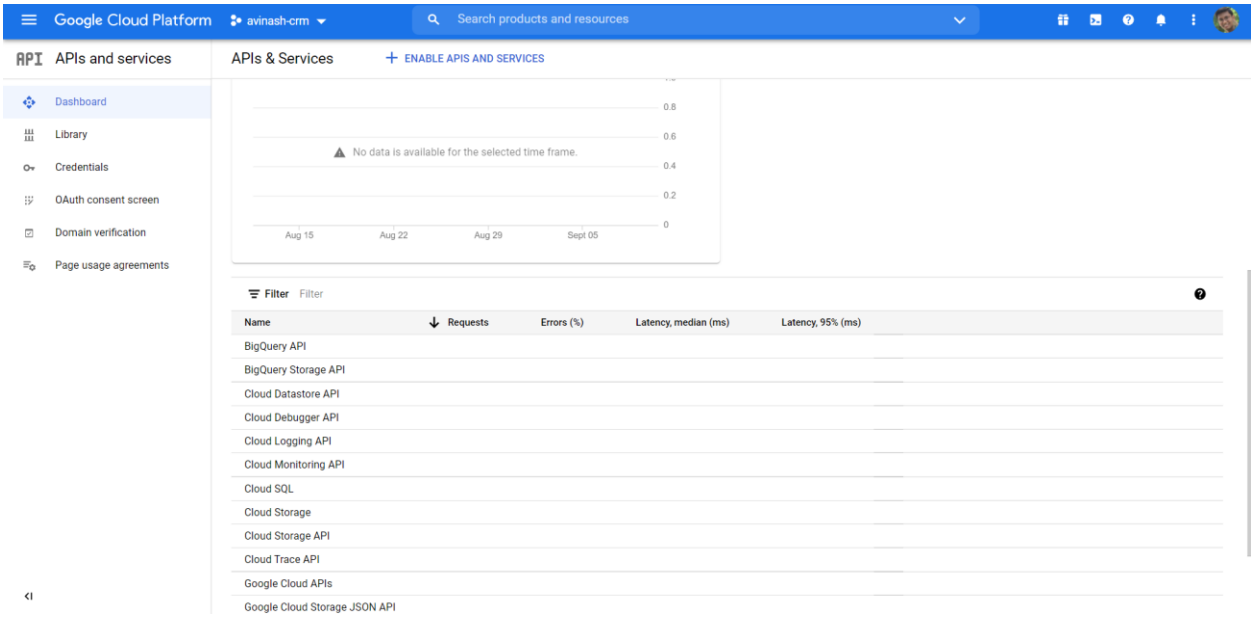


Fig 3.9 Google API Service

3.6 VS Code (Visual Studio)

Visual Studio Code is a code editor in layman's terms. Visual Studio Code is "a free-editor that helps the programmer write code, helps in debugging and corrects the code using the intelli-sense method". In normal terms, it facilitates users to write the code in an easy manner. Many people say that it is half of an IDE and an editor, but the decision is up to the coders. Any program/software that we see or use works on the code that runs in the background. Traditionally coding was used to do in the traditional editors or even in the basic editors like notepad! These editors used to provide basic support to the coders.

There are many advantages over any other IDE; they are as follow:

- Cross-platform support :
    - Windows
    - Linux
    - Mac
- Light-weight
- Robust Architecture
- Intelli-Sense
- Freeware: Free of Cost- probably the best feature of all for all the programmers out there, even more for the organizations.
- Many users will use it or might have used it for desktop applications only, but it also provides great tool support for Web Technologies like; HTML, CSS, JSON.

With advancements in technology day-by-day, Visual Studio Code is going to play a pivotal role in the development of software. With its ever-evolving features and soon-to-be-added new settings, which will enable users to work with it from anywhere, it is certainly "THE THING" to keep one ahead of everyone in this ever-increasing IT market.

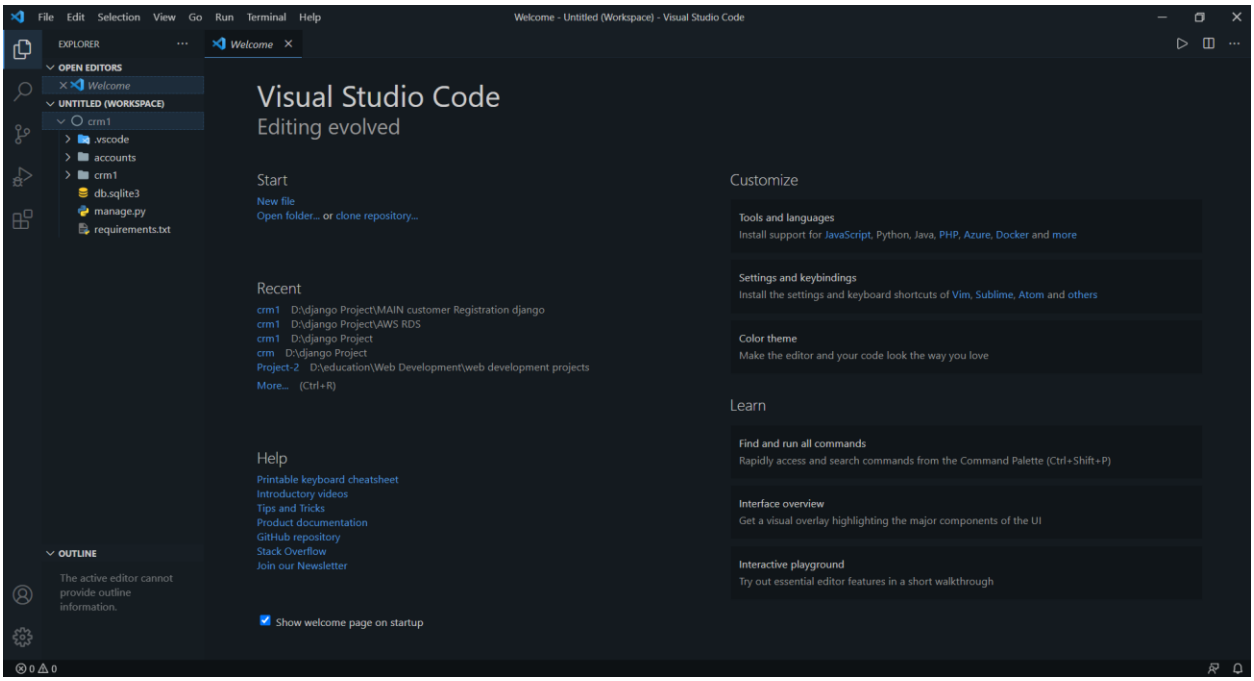


Fig 3.10 Visual Studio Code

CHAPTER 4

DESIGN OF PROJECT

4.1 Hardware Requirements

A computer which is not so old and can work smoothly.

- Processor: Intel core i3 or more
- Hard Drive: 2GB or more
- Memory (RAM): 1GB or more
- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
- Server computer to host your website on to make sure that it is available all the time for anyone trying to access it or simply purchase a hosting package from a web hosting company.

4.2 Software Requirements

4.2.1 Admin Side

- AWS Account with free tier
- VS code for code editing
- Django and Python installed
- Windows 7 or Higher, Linux
- Web browser with active internet connection Preferred Chrome

4.2.2 User Side

- Web browser(Chrome)
- Web connectivity
- Wimdows7 or Higher, Linux

4.3 AWS Infrastructure Diagram

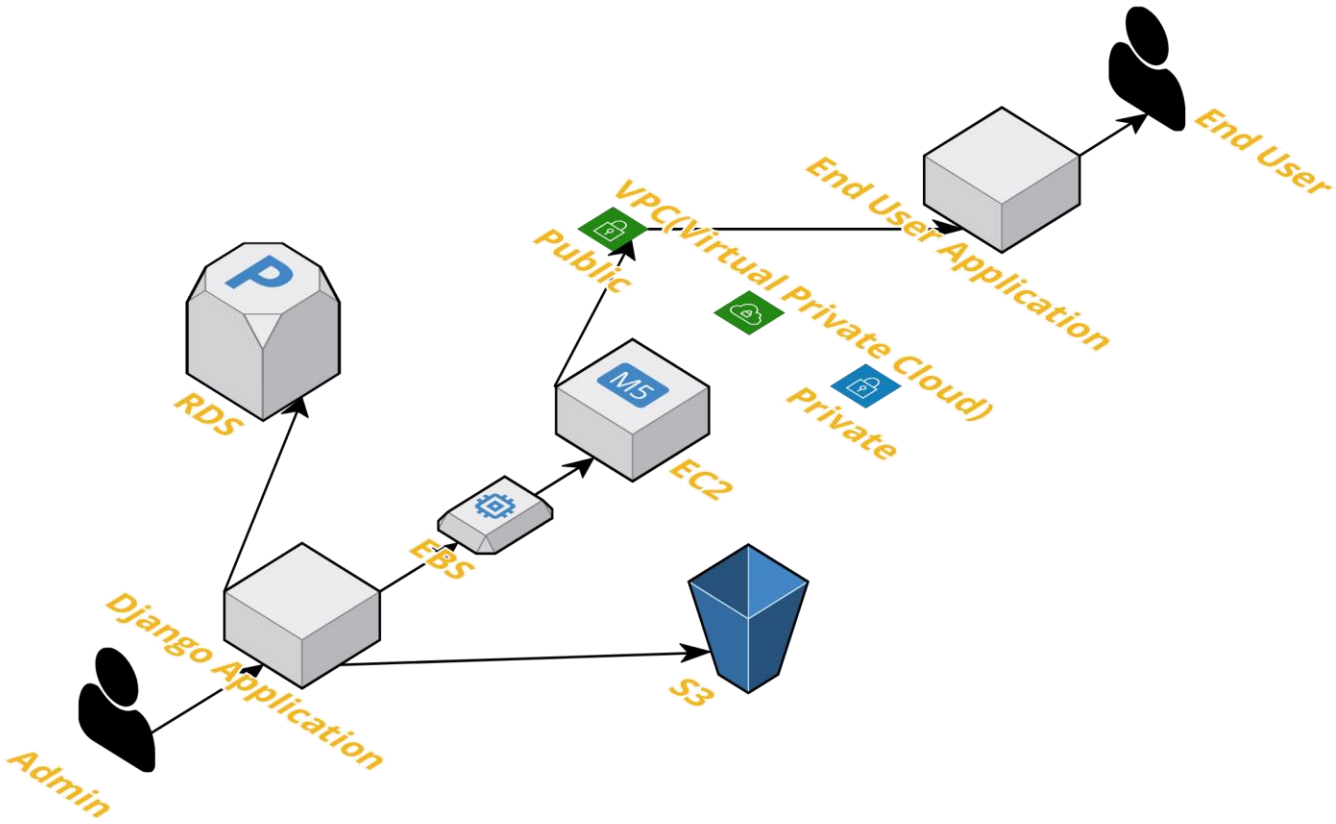


Fig 4.1 AWS Infrastructure Diagram

4.4 Entity Relationship Diagram

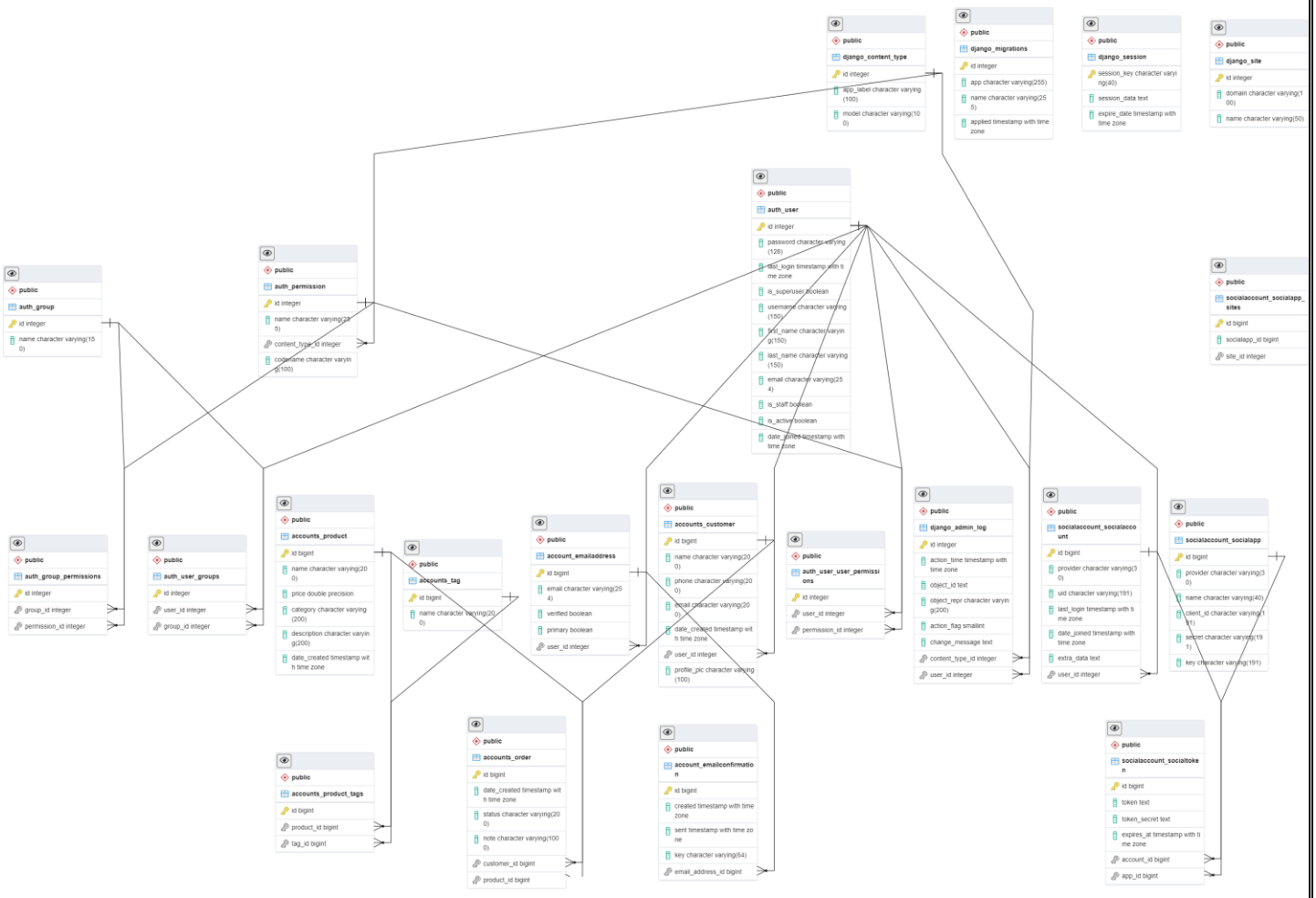


Fig 4.2 ERD of the Project Database

4.5 Use Case Diagram

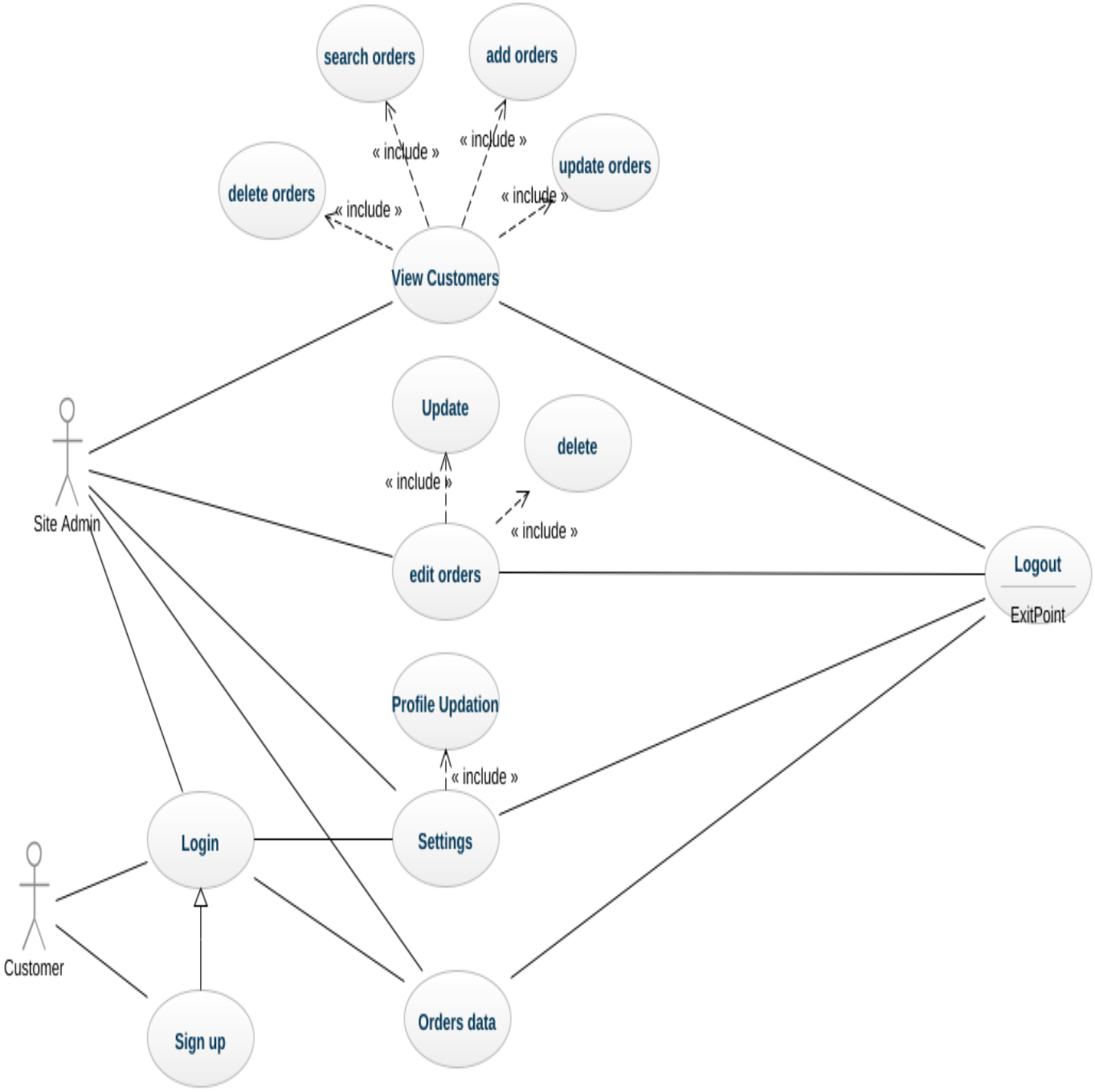


Fig 4.3 Use Case Diagram

CHAPTER 5

IMPLEMENTATION

5.1 Django Application

The Application Used is Customer registration management. This app is totally developed in the django framework.it is a fully functional website with a database. This application stores customer information in a database users will have the ability to create customers and customer orders along with viewing those customers and updating customer information like orders and basic contact information on a customer's profile page we'll also have the ability to search customer information using a multi-parameter search form.

5.1.1 Installing Django and getting a basic Project setup

```
C:\Users\HP>pip install django
Requirement already satisfied: django in d:\python39\lib\site-packages (3.2.6)
Requirement already satisfied: asgiref<4,>=3.3.2 in d:\python39\lib\site-packages (from django) (3.4.1)
Requirement already satisfied: sqlparse>=0.2.2 in d:\python39\lib\site-packages (from django) (0.4.1)
Requirement already satisfied: pytz in d:\python39\lib\site-packages (from django) (2021.1)

C:\Users\HP>
```

Fig 5.1 installing django in Python environment

Creating the Project using the Django command

Django-admin startproject projectname

Initially after creating the project the list of directories that are seen are shown below

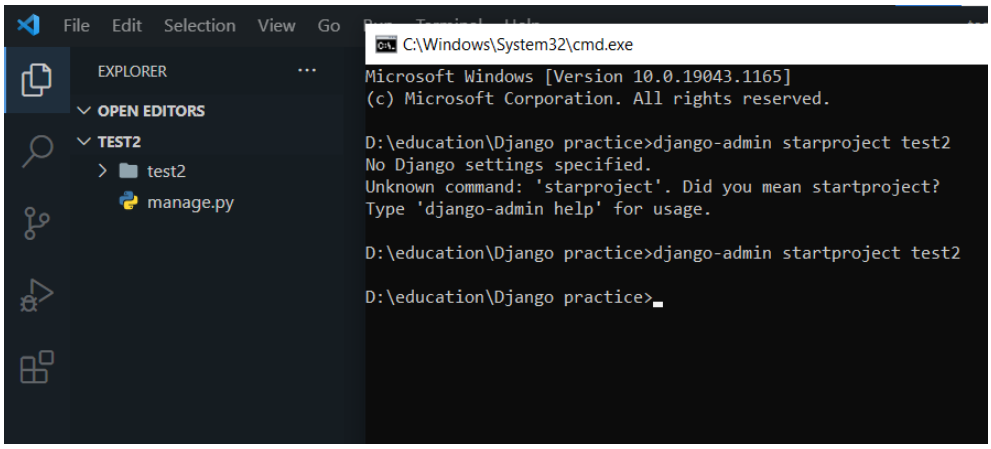


Fig 5.2 New Project file

After successfully running server by using the command *python manage.py run server you* get the following output

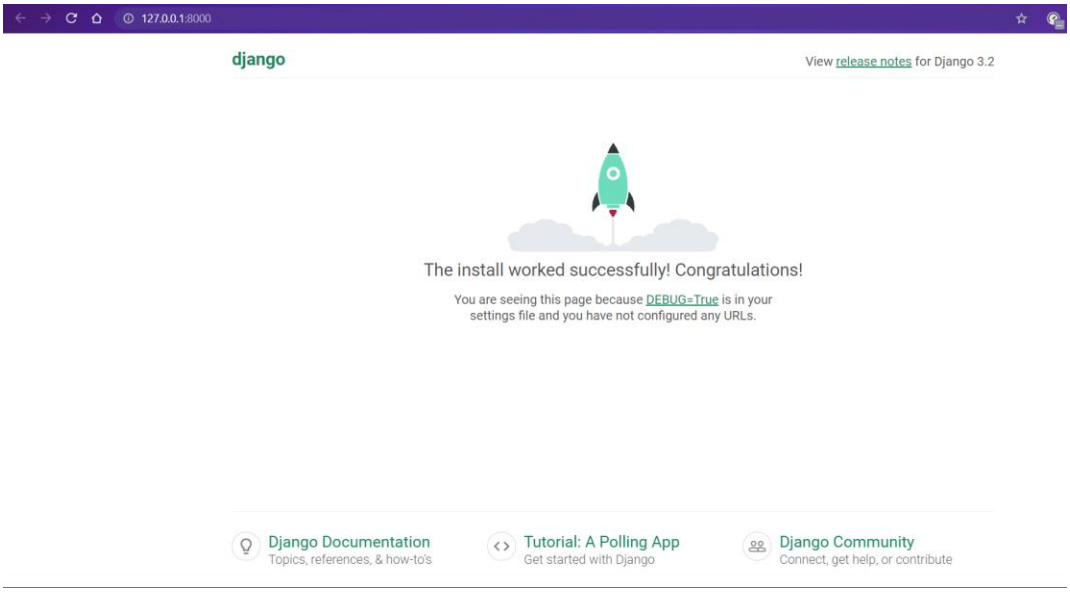


Fig 5.3 Successfully Running server

5.1.2 URL'S and VIEWS

5.1.2.1 URL's

Django stores URLs and this urls.py file in our root directory and whenever we use our types in a URL path it's this files job to find the pattern that the user typed into the browser and return them back a page and Django based on whatever URL pattern was matched it has to trigger a function and it's a functions job to return back the page that the user is looking for

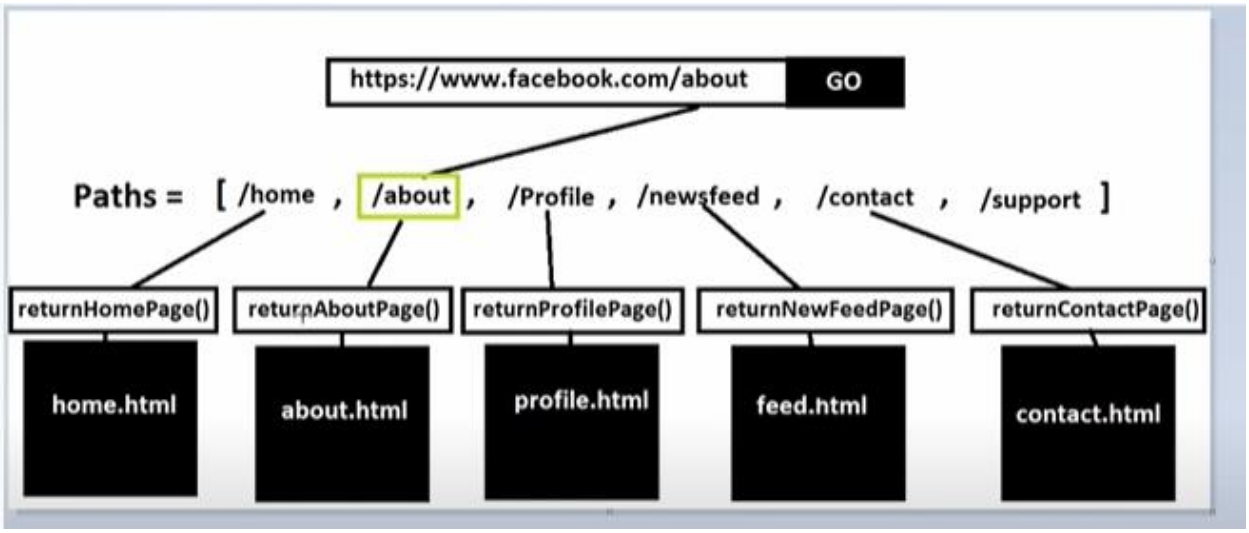


Fig 5.4 Demonstrating URL's

so whenever the user type in a URL path and we'll just use facebook.com/about so whenever the user types us in we have this list of paths right here so these are basically patterns that are matched and that's what you're seeing right here this URLs pattern variable so a lot will be added so you'll actually if you notice it's a list and there's any a lot of patterns like this and each one will have a different path and that pattern is matched so in this case we match to the about page and the about page is supposed to return the function

The URL's used in this project are the Main urls.py file and the file in the app that is created are having the URL routing that is required for the smooth functionality of the application.

These URL's are totally dependent on the function that we write in the views.py These functions return HTTP Response or they can also return the html file using render functionality that is provided by the django web framework.

```
from django.contrib import admin
from django.urls import path, include
from django.http import HttpResponse
from django.conf.urls.static import static
from django.conf import settings
from django.views.generic import TemplateView
from django.contrib.auth.views import LogoutView

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('accounts.urls'))
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Fig 5.5 main project urls.py file

5.1.2.2 Views that are created for the app are

- **Register Page**

  Using Usercreationform Module from the django auth templates we can easily write a function to create by taking the http response from the user in the form of POST and setting it as shown in the below code

```
form = CreateUserForm()
if request.method == 'POST':
    form = CreateUserForm(request.POST)
    if User.objects.filter(username = request.POST['username']).exists():
        messages.error(request, 'Username already taken')
        return redirect('register')
    if form.is_valid():
        user = form.save()
        username = form.cleaned_data.get('username')
        messages.success(request, 'Account was created for ' + username)
        return redirect('login')

context = {'form':form}
return render(request, 'accounts/register.html', context)
```

Fig 5.6 Register Views Function

- **Login Page**

Using the authenticate functionality of django it becomes easy to verify the user and return the login details of the user after successful login into the server. The following functionality is shown in the below code

```
def loginPage(request):

    if request.method == 'POST':
        username = request.POST.get('username')
        password =request.POST.get('password')

        user = authenticate(request, username=username, password=password)
        if user is not None:
                login(request, user)
                return redirect('home')
        else:
            messages.info(request, 'Username OR password is incorrect')
    context = {}
    return render(request, 'accounts/login.html', context)
```

Fig 5.7 Login Views Function

- **Logout**

Simple logout Functionality is enough to logout of the session in django

- **Home**

Using the django data fetching techniques we fetch the data of the orders and all other customer related details and output in the main home page.

- **User page**

Using the django data fetching techniques we fetch the data of the orders and all other customer related details and output in the main home page.

- **Account Settings**

Here the user are allowed to edit the details that are provided during the signup activity. The user can also add pictures to the profile and edit the personal details.

- **Products**

Here the page shows the products that6 are available for the customers.

- **Customer**

It only shows us the products status as they place them

- **Create order**

  The admin can create order to the customers using the functionalities using the django database.

- **Update order**

  Similar to the creation the Updating also works the same

- **Delete order**

  It permanently deletes the order in the database

5.1.3 Templates

Django's template is a simple text file which can generate a text-based format like HTML and XML. The template contains variables and tags. Variables will be replaced by the result when the template is evaluated. Tags control the logic of the template. We also can modify the variables by using filters. For example, a lowercase filter can convert the variable from uppercase into lowercase.

Below are the following templates that are used in the project

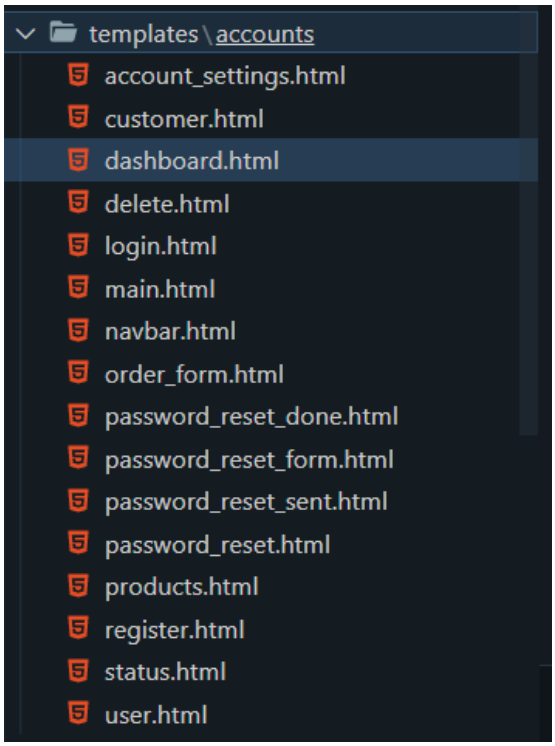


Fig 5.8 Templates in the Project

5.1.4 Static Files

Static files are basically a folder that we can create and store things like CSS JavaScript and images and it basically lets us separate our code from Python code and store any extra things in that particular folder.

For These static files to be loaded into the application and templates we need to load these static files from the application and render it back to the templates. This is done using the command

{% Load Static %}

We also need to configure some settings in the settings.py file to render these files. We need to add two urls

STATIC_URL =”/static”

MEDIA_URL=”/images/”

Presently we are doing everything in the local hard drive, after creation of the application we then migrate it to the AWS.

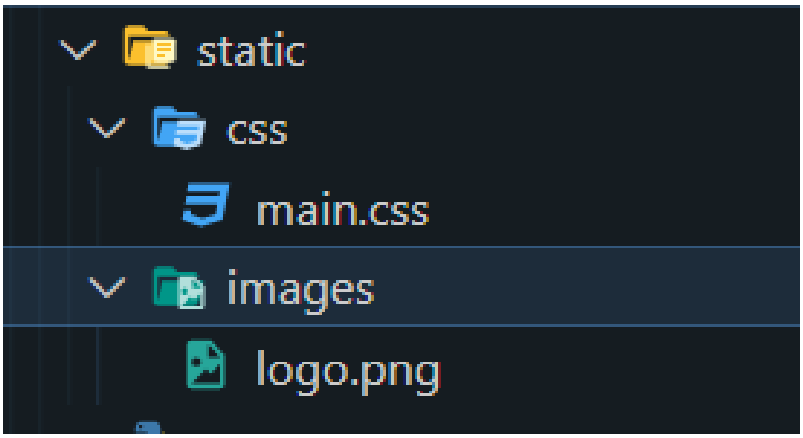


Fig 5.9 Static Files

5.1.5 Database Models and Admin Panel

Models are the modern django forma of creating tables and rendering data to it. Model is a single, definitive data source which contains the essential field and behavior of the data. Usually one model is one table in the database. Each attribute in the model represents a field of a table in the database. Django provides a set of automatically-generated database application programming interfaces (APIs) for the convenience of users.

Models that are created in this application are

- **Customer**

  This is created to store the user's data for the customers that login to the application using the login system provided. The implementation is below

```
class Customer(models.Model):
    user = models.OneToOneField(User, null=True, on_delete=models.CASCADE)
    name = models.CharField(max_length=200, null=True)
    phone = models.CharField(max_length=200, null=True)
    email = models.CharField(max_length=200, null=True)
    profile_pic = models.ImageField(default="profile1.png", null=True, blank=True)
    date_created = models.DateTimeField(auto_now_add=True, null=True)

    def __str__(self):
        return self.name
```

Fig 5.10 Customer Model

- **Tag**

  This is created to differentiate the type of products by their category which is demonstrated by the tags feature in this model. The implementation is below

```
class Tag(models.Model):
    name = models.CharField(max_length=200, null=True)

    def __str__(self):
        return self.name
```

Fig 5.11 Tag Model

- **Product**

This is the model used to create a table related to the product which consist of the tag that is related the price of the product and the product description for the better understanding of the product the product demonstration of the model is shown below

```
class Product(models.Model):
    CATEGORY = (
            ('Indoor', 'Indoor'),
            ('Out Door', 'Out Door'),
            )

    name = models.CharField(max_length=200, null=True)
    price = models.FloatField(null=True)
    category = models.CharField(max_length=200, null=True, choices=CATEGORY)
    description = models.CharField(max_length=200, null=True, blank=True)
    date_created = models.DateTimeField(auto_now_add=True, null=True)
    tags = models.ManyToManyField(Tag)

    def __str__(self):
        return self.name
```

Fig 5.12 Product Model

- **Order**

This model demonstrates the order ideas that is related to the both the customers and products table it consists of a foreign key of both the tables and relate both these tables using many to many and one to many relationships in the relational database model. The implementation of the model is shown below

```
class Order(models.Model):
    STATUS = (
            ('Pending', 'Pending'),
            ('Out for delivery', 'Out for delivery'),
            ('Delivered', 'Delivered'),
            )

    customer = models.ForeignKey(Customer, null=True, on_delete= models.SET_NULL)
    product = models.ForeignKey(Product, null=True, on_delete= models.SET_NULL)
    date_created = models.DateTimeField(auto_now_add=True, null=True)
    status = models.CharField(max_length=200, null=True, choices=STATUS)
    note = models.CharField(max_length=1000, null=True)
```

Fig 5.13 Order Model

5.2 AWS Migration

5.2.1 AWS RDS (Relation database service)

This service is used to replace the local database and use a universal database for the implementation of the service you need to create aws account. You will a Free tier Access of the first year which provides 750hours of AWS RDS service per month and 20GB SSD for the storage with all these features we can make our database the creation should be with the configuration as shown below.

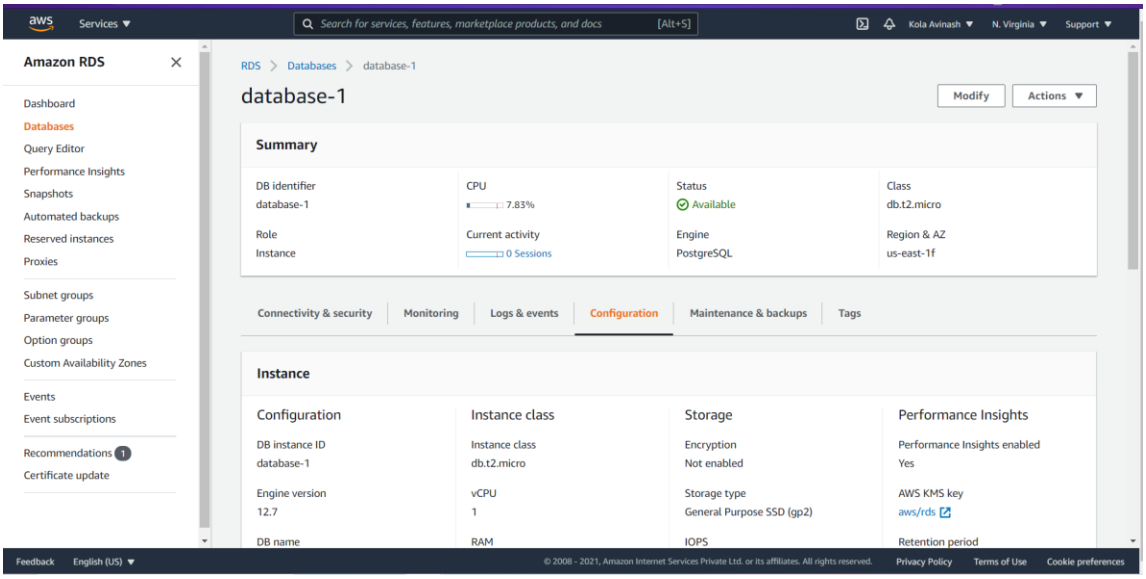


Fig 5.14 AWS RDS Service configuration

The RDS Service Provide the Databases for Oracle, MySQL, PostgreSQL, etc... .For this Project I have used the PostgreSQL database. The configuration is not done yet we need to integrate it with the application for these we need to configure some features in the settings.py file of the application

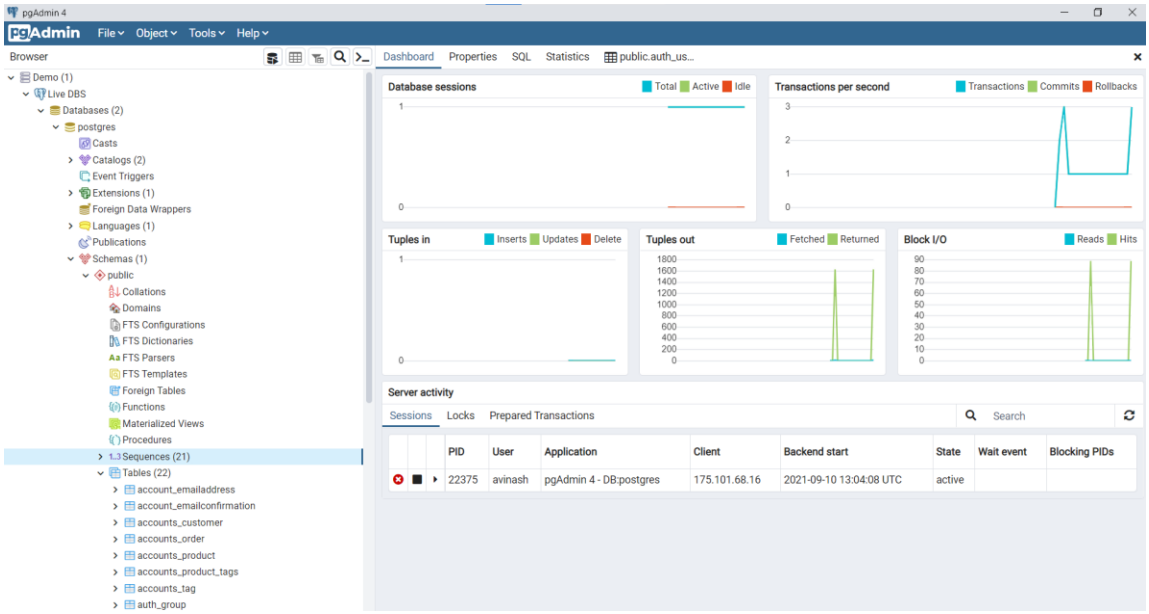


Fig 5.15 PostgreSQL database of AWS RDS

Configuring settings.py file for the database access

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'postgres',
        'USER':'avinash',
        'PASSWORD': [redacted],
        'HOST':'database-1.c5wjhlm[redacted]p.us-east-1.rds.amazonaws.com',
        'PORT':'5432'
        # 'ENGINE': 'django.db.backends.sqlite3',
        # 'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Fig 5.16 AWS RDS Configuration

5.2.2 AWS S3 (Simple Storage Service)

S3 buckets are just a static file storage system on Amazon Web Services that allow us to host and serve our static files like images directly from there rather than our single projects so what I mean by that is if you have a project of this size and you just want to upload this to a AWS EC2 server you could just upload it and have your static files served like this but once you start uploading more pictures maybe you have users with profile pictures you're going to want to serve these outside of Server.

For these purposes we need to create a bucket that stores these static files and these files make the connection to the application. The bucket that is created is as follows.

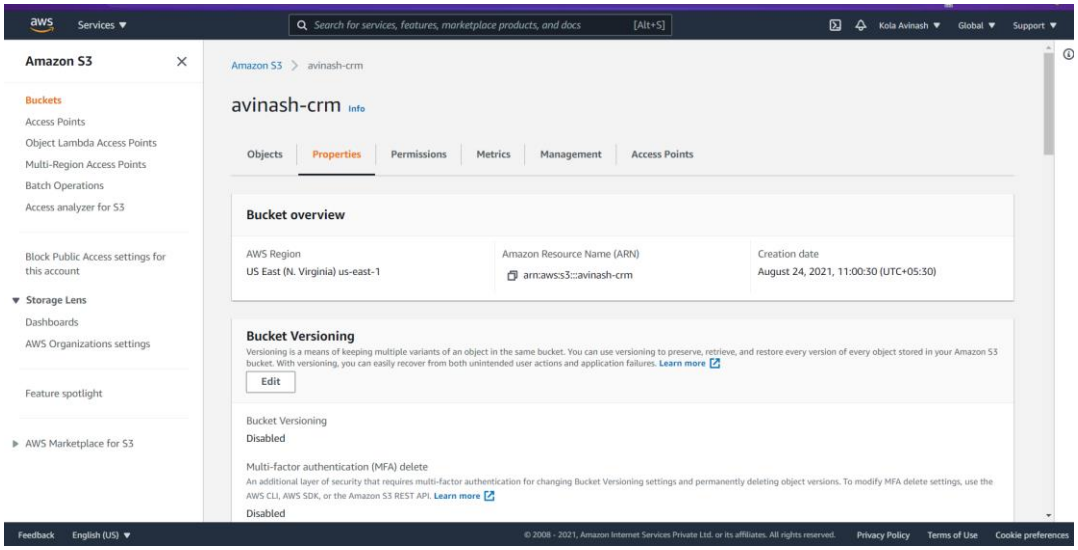


Fig 5.17 AWS S3 Bucket

For the access of this bucket created you need to create an IAM User which have the full access to the s3 bucket and using this user in the django application we can make changes to our bucket from the application using the UI itself. The major changes are we need to fetch our static files like the css styles and other images that are used in our website. All these files are stored in the bucket and made available to the application from the IAM user created.

Settings.py file for the configuration of the s3 bucket is as follows

```
#S3 BUCKETS CONFIG

AWS_ACCESS_KEY_ID = '[redacted]
AWS_SECRET_ACCESS_KEY = '[redacted]
AWS_STORAGE_BUCKET_NAME = 'avinash-crm'
AWS_S3_FILE_OVERWRITE = False
AWS_DEFAULT_ACL = None
DEFAULT_FILE_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
STATICFILES_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
```

Fig 5.18 AWS S3 settings.py configuration

5.2.3 AWS EC2 (Elastic Cloud Compute)

After you create an AWS account, login with that username and password, and choose Elastic Compute Cloud from the list of services available. Then select sign up for EC2 from the EC2 homepage. Amazon providers ask for your credit card details (but is not charged till you use their resources) and then you can continue with the process. After you successfully gained access to EC2 service, go to AWS Management Console, which is a web-based, point-and-click, graphical user interface that makes it even easier to access and manage AWS Infrastructure Web Services. Information about cost per hour for an instance, bandwidth, data transfer rate and others can be known from https://aws.amazon.com//ec2/ i.e., ec2 homepage.

The EC2 instance created for this project is part of the Free Tier and consists of a windows server with 1GB Ram and 30GB Storage of SSD which is more than enough for this project. The EC2 instance created is as follows

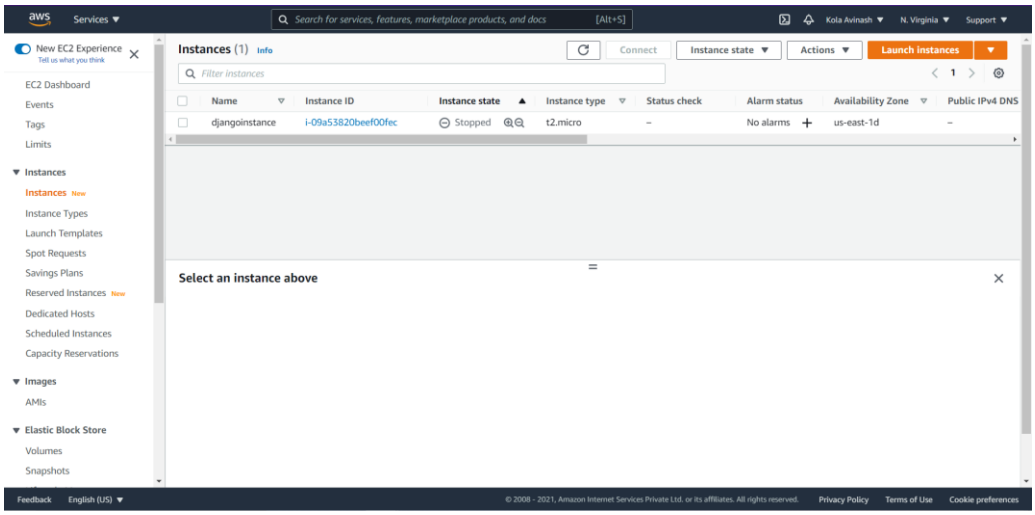


Fig 5.19 EC2 Instance in AWS

For using this service we need to connect to the desktop/server that is assigned using amazon secure connect and access the server after accessing the server we need to install the django service in to the server and copy the files of the project into a zip folder and move it to the server in aws and unzip it. After that we need to configure the security credential and should allow the server to connect remotely through any device. And we should run our server at port 80 so that the server is accessible to the users from the amazon EC2 public DNS address.

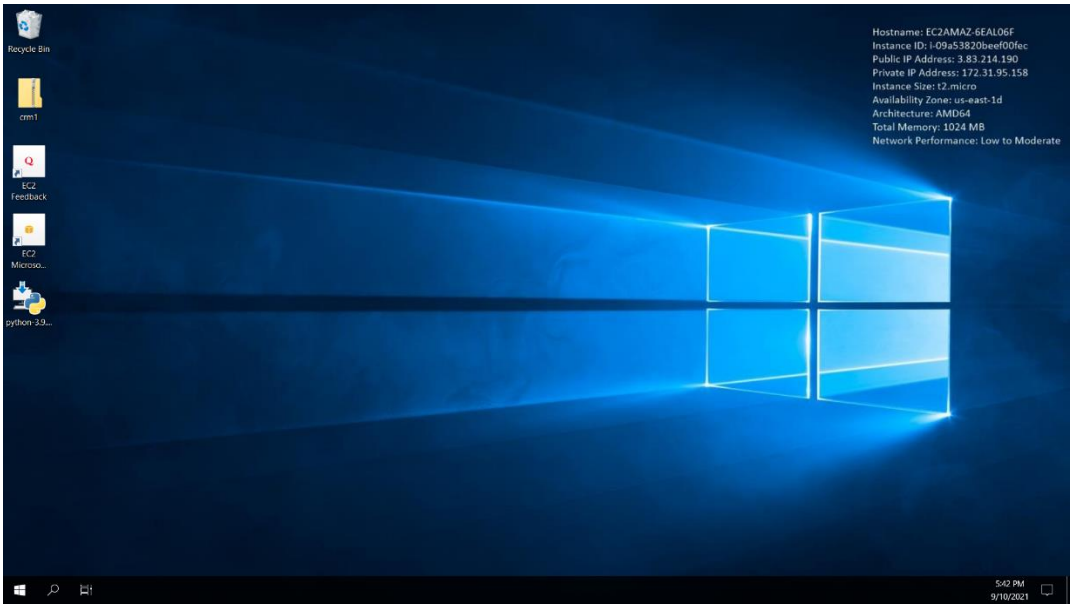


Fig 5.20 AWS EC2 Server

```
C:\crm1>python manage.py runserver 0.0.0.0:80
Performing system checks...

System check identified no issues (0 silenced).
September 10, 2021 - 17:45:53
Django version 3.2.6, using settings 'crm1.settings'
Starting development server at http://0.0.0.0:80/
Quit the server with CTRL-BREAK.
```

Fig 5.21 AWS EC2 Running Server

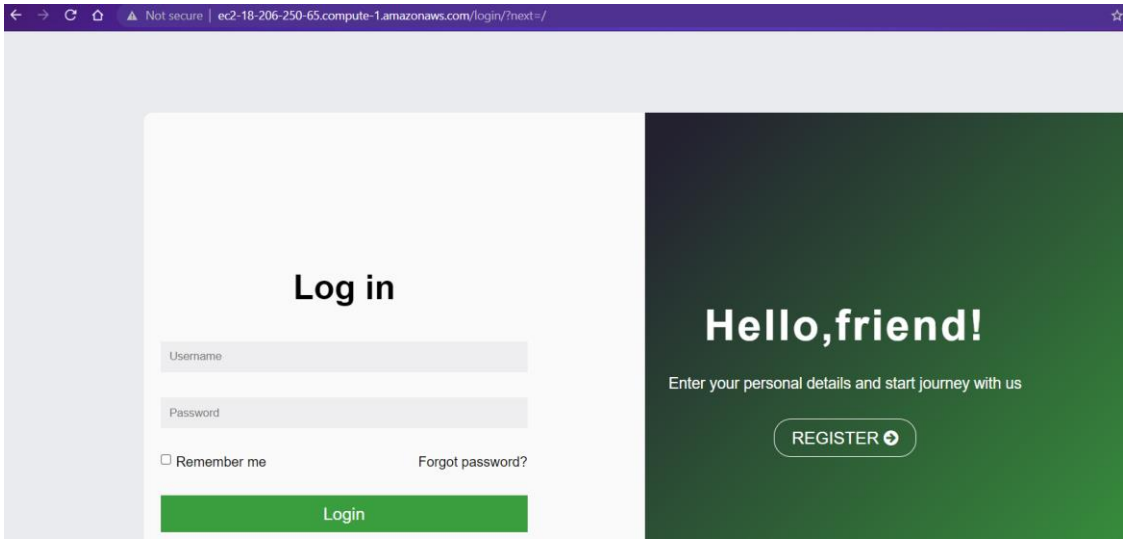


Fig 5.22 accessing the application from Public DNS Address

From the above images we can see that we have connected to the server from the AWS UDP client and accessed our server that is being assigned to us. After that we have installed python and launched our django site at port 80 as shown in fig 5.21. from that our server is launched and the we can access our application from the public DNS address of the EC2 instance created.

Chapter 6

SCREENSHOTS

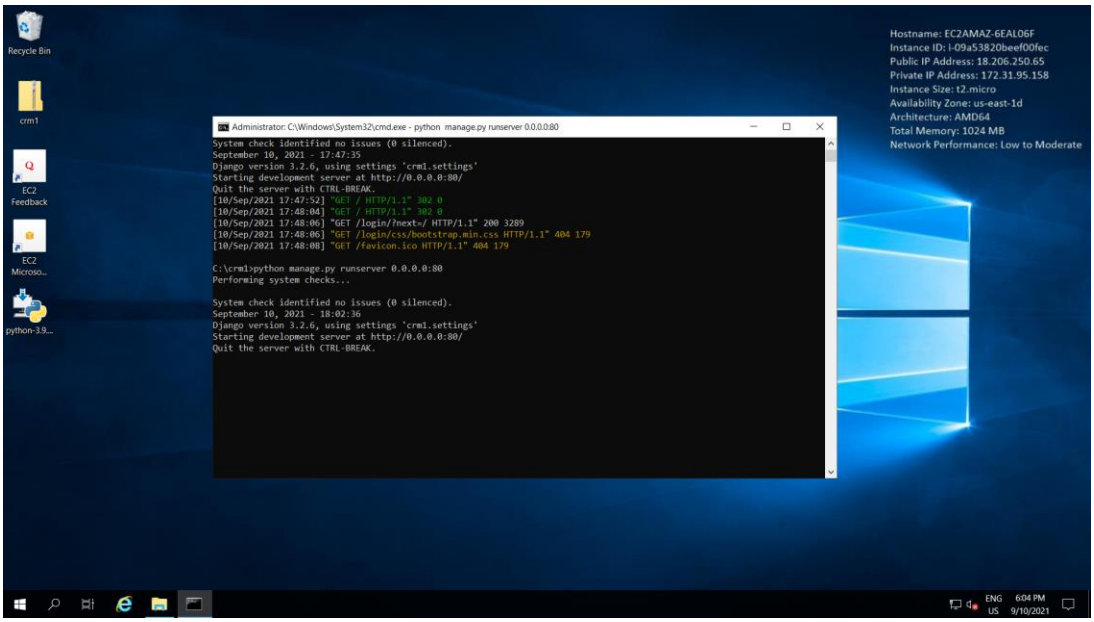


Fig 6.1 EC2 Server Started

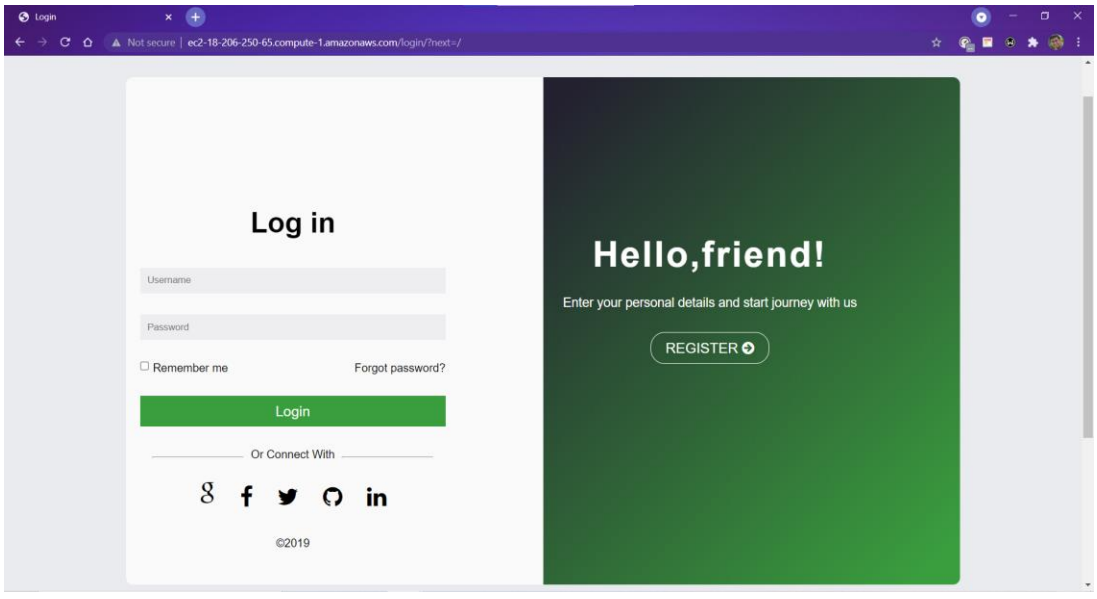


Fig 6.2 Main Login Page

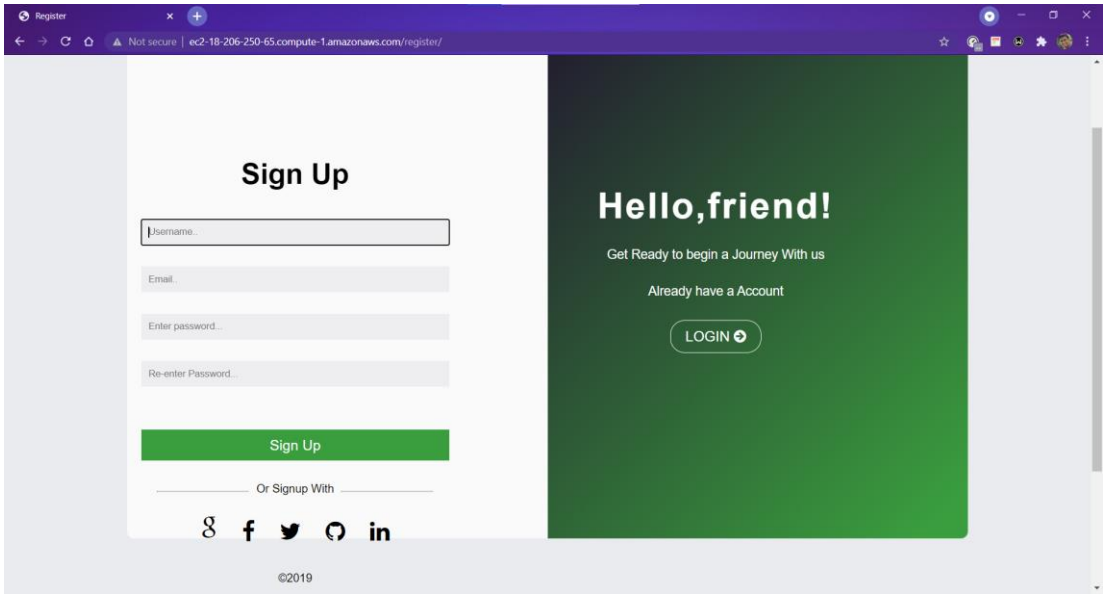


Fig 6.3 Main Sign up Page

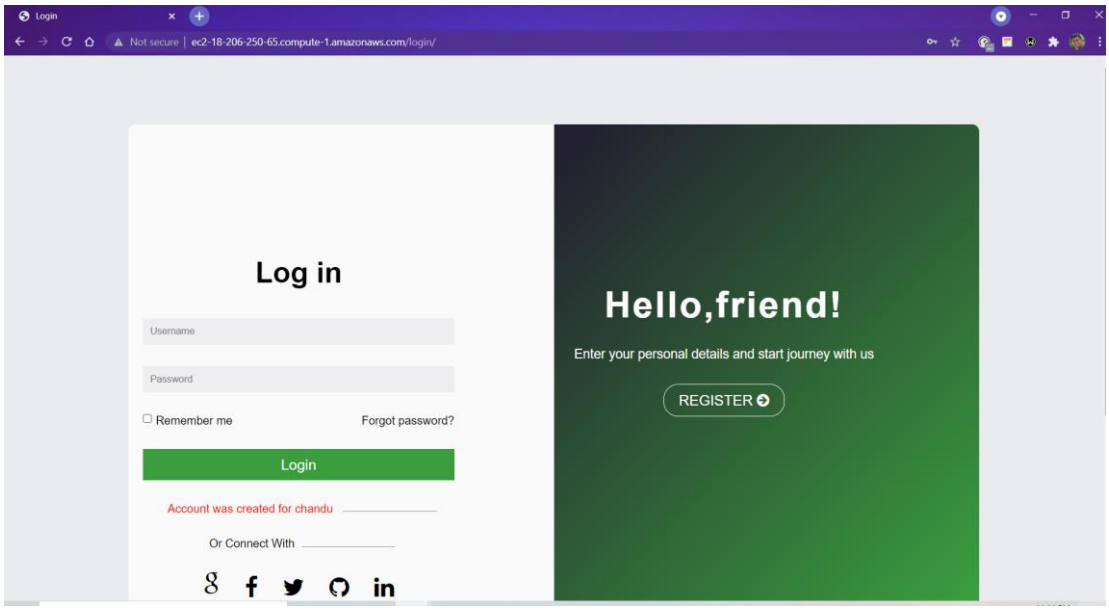


Fig 6.4 after Successful Sign Up

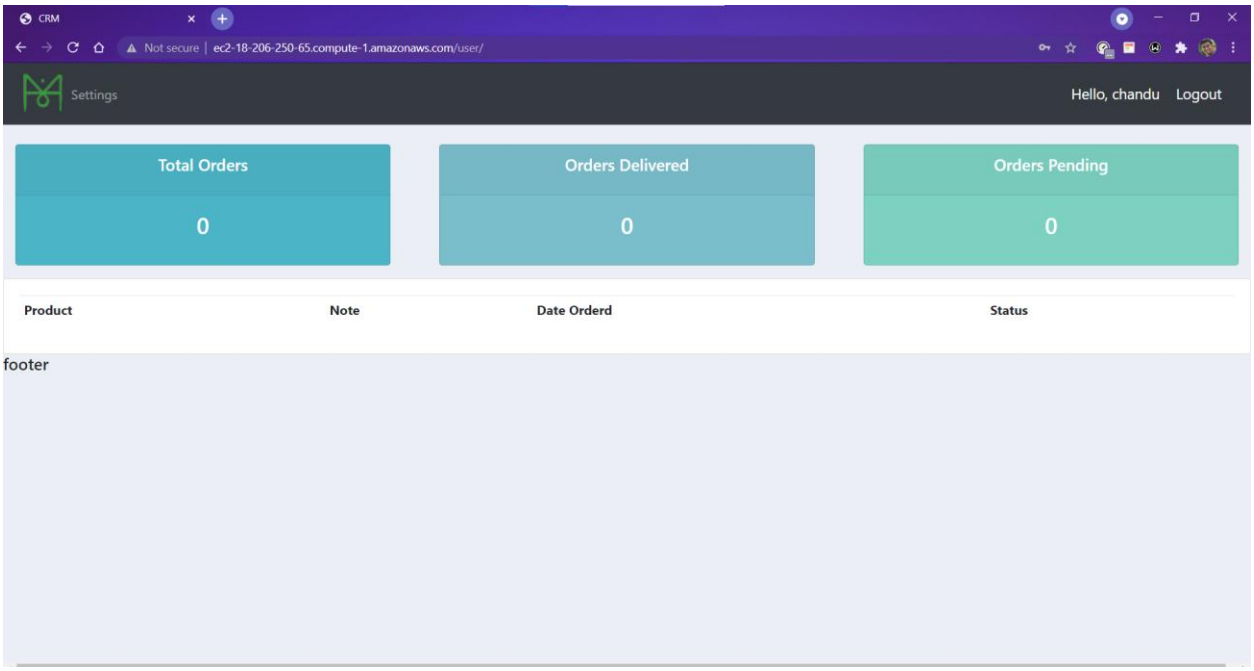


Fig 6.5 after Successful Customer Login

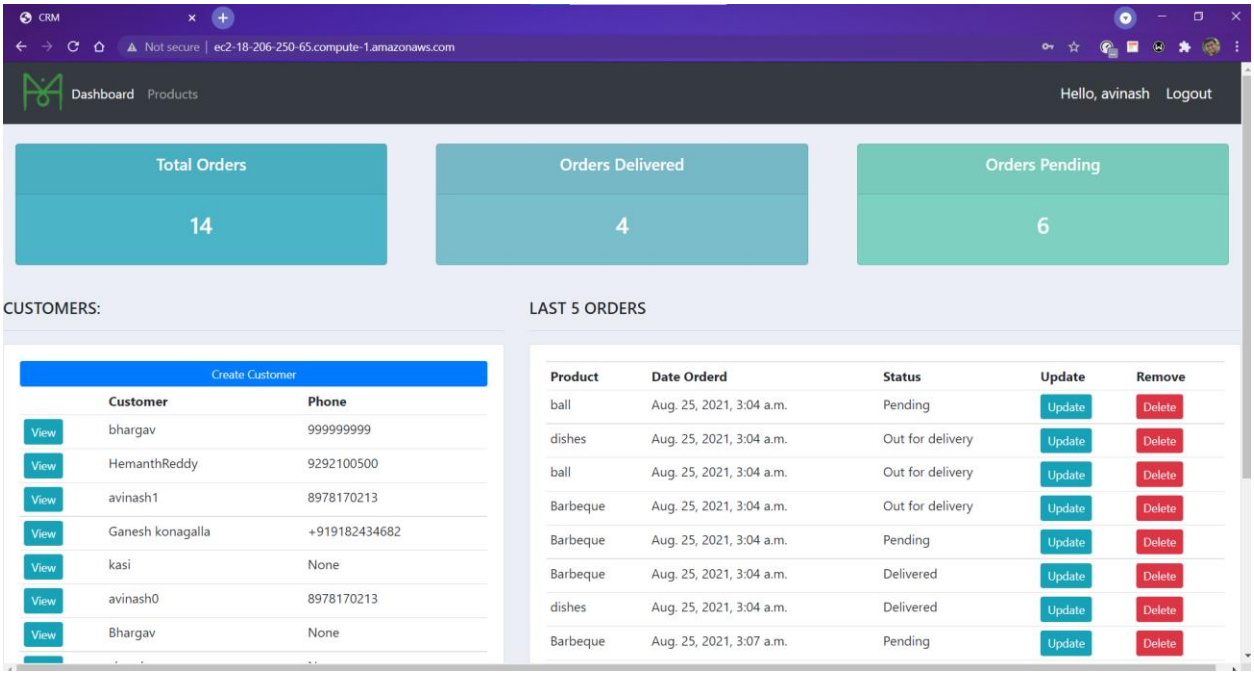


Fig 6.6 after Successful Admin Login

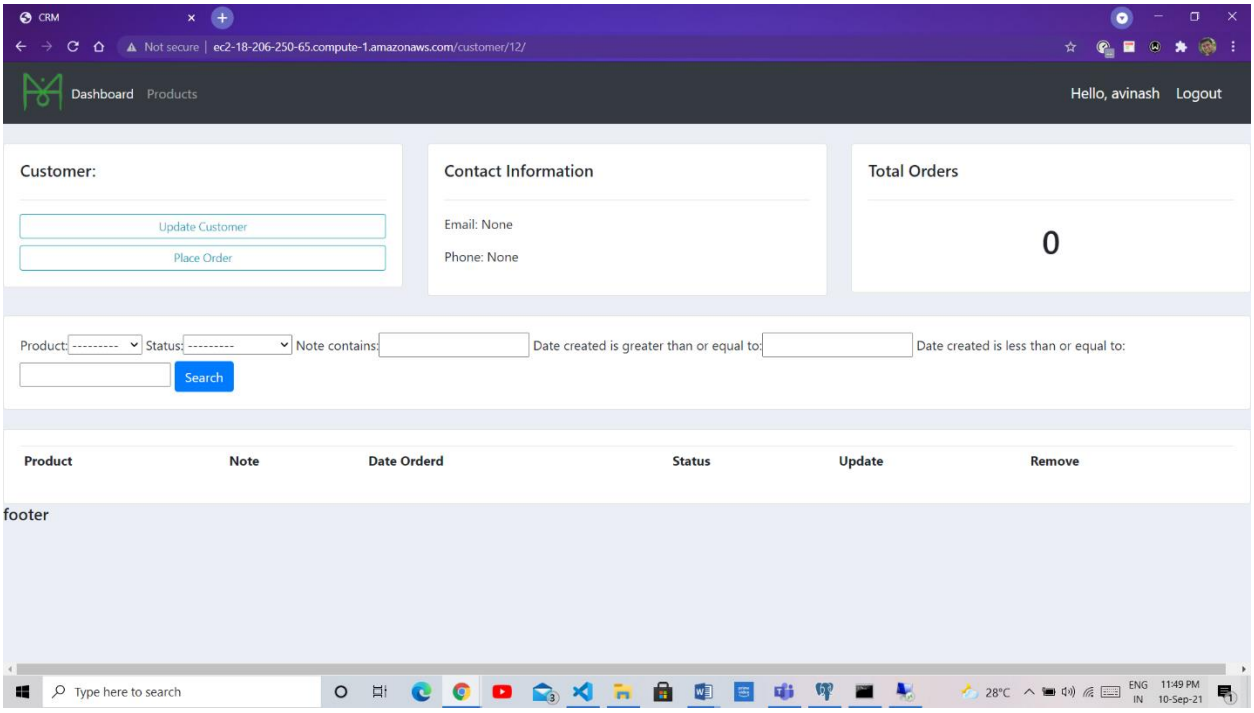


Fig 6.7 Admin Viewing Customer

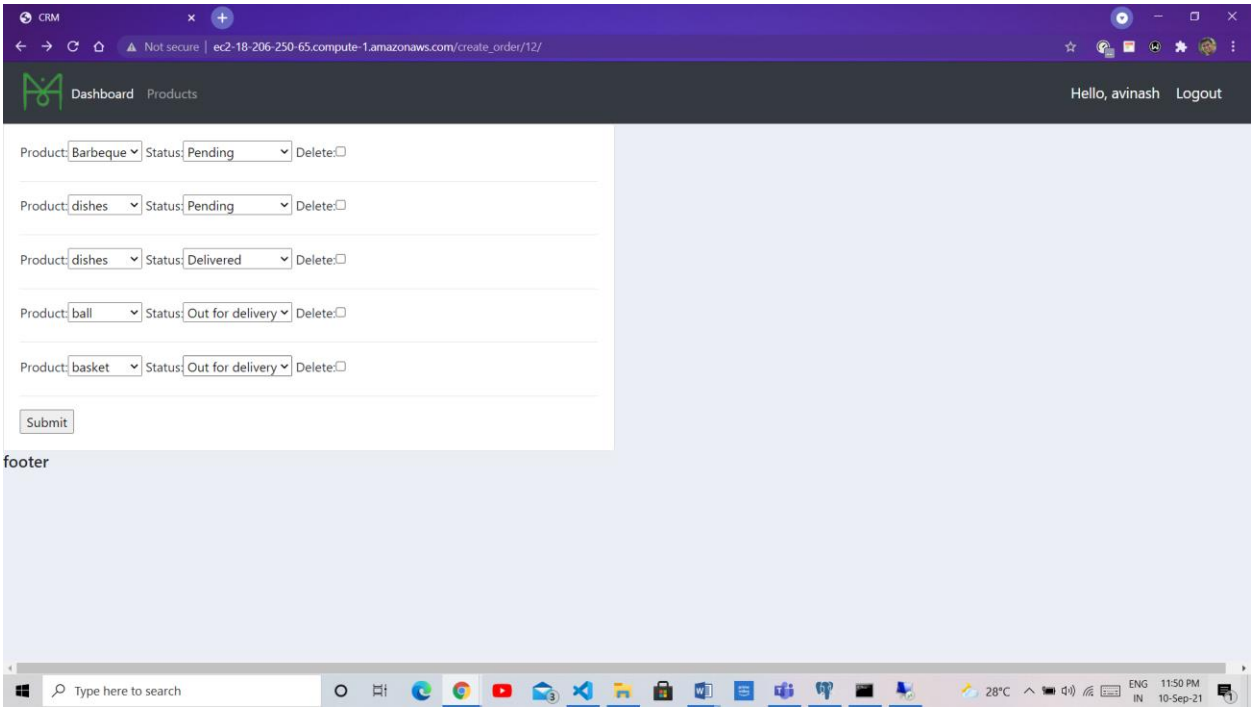


Fig 6.8 Admin placing order for Customer

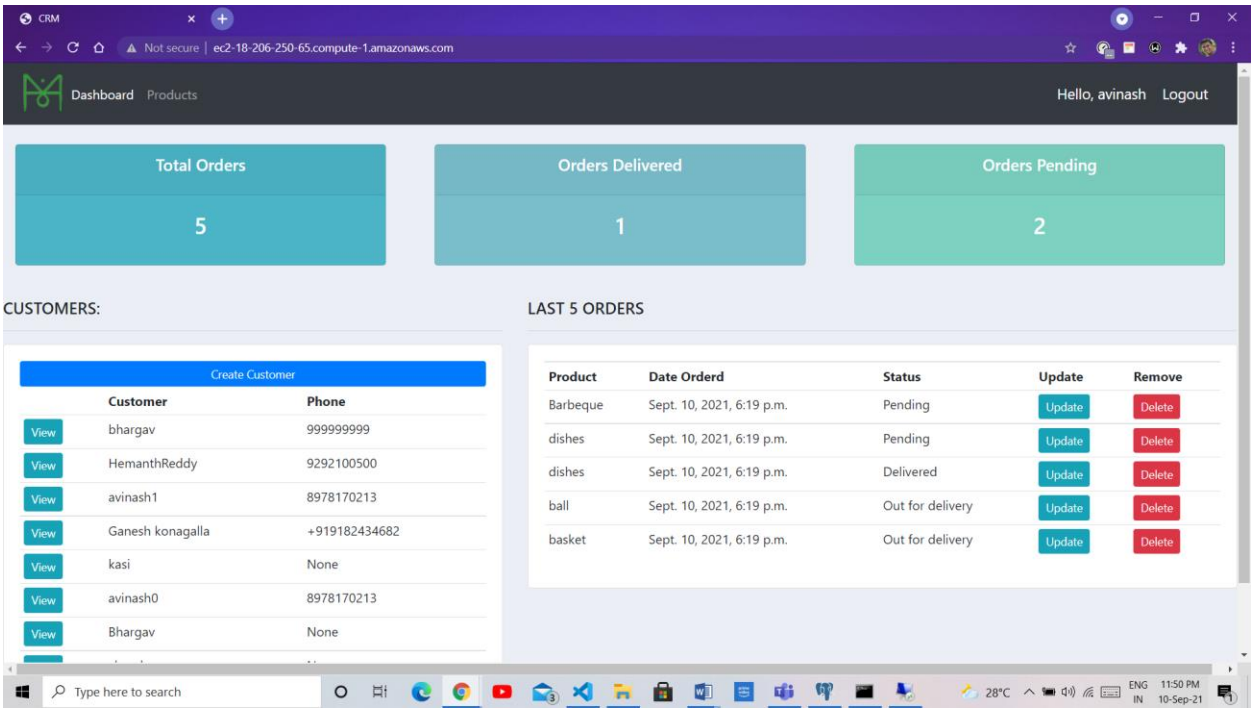


Fig 6.9 Admin page after placing order

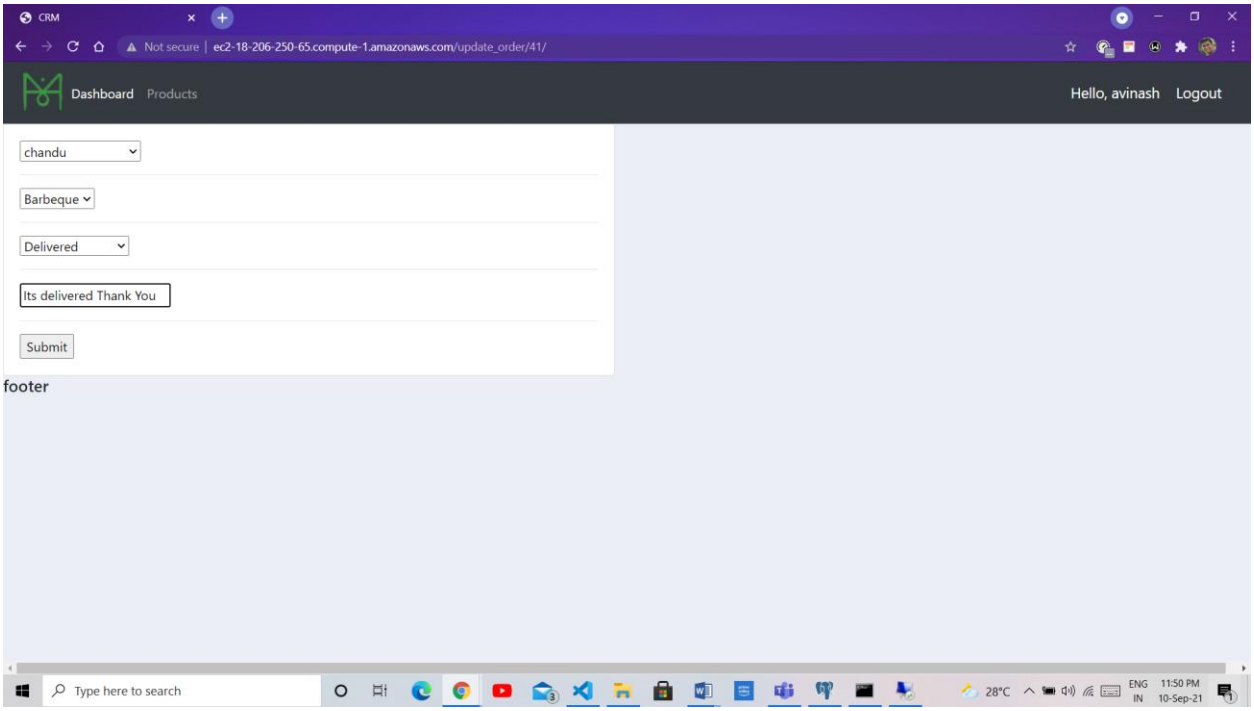


Fig 6.10 Admin Updating Order status

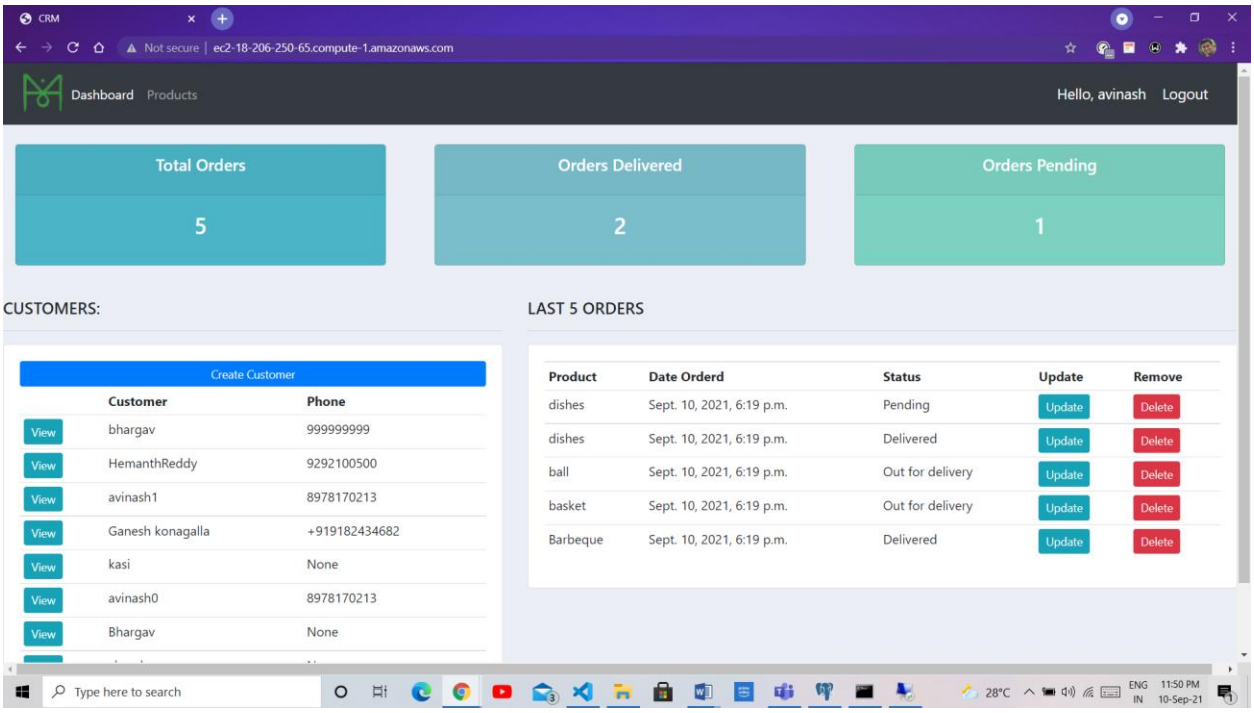


Fig 6.11 Admin after Updating Barbeque to delivered

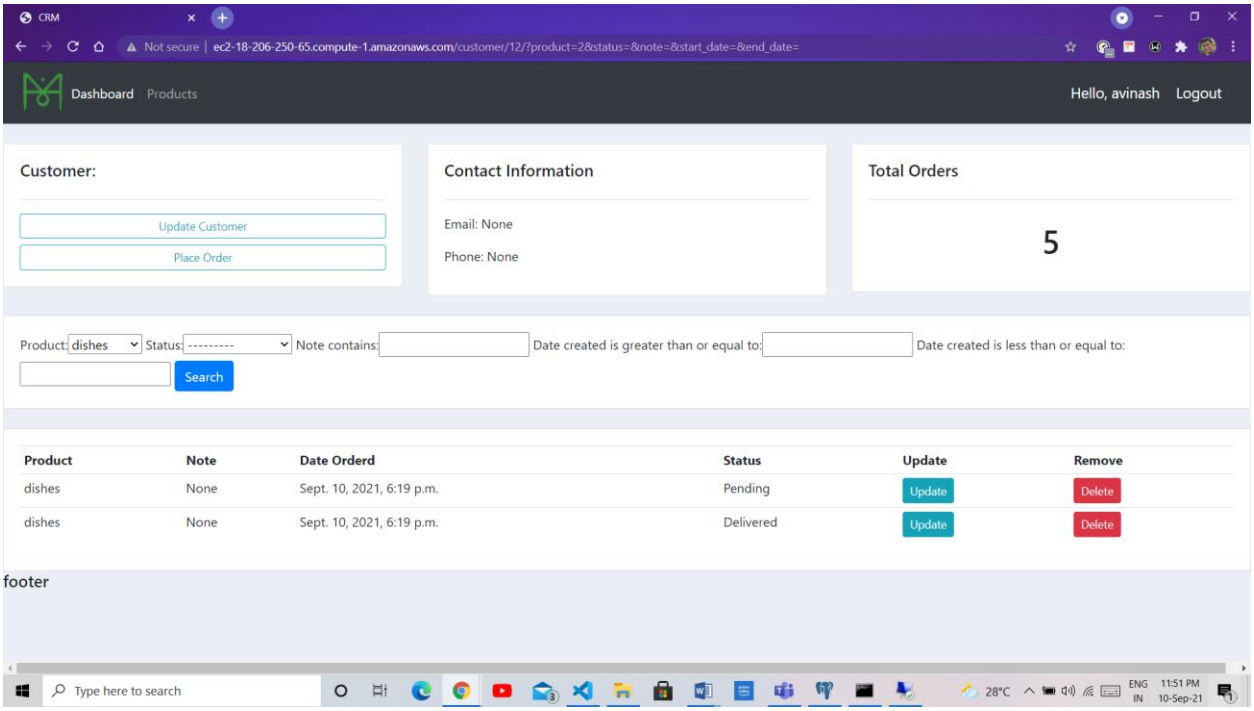


Fig 6.12 Searching By the product

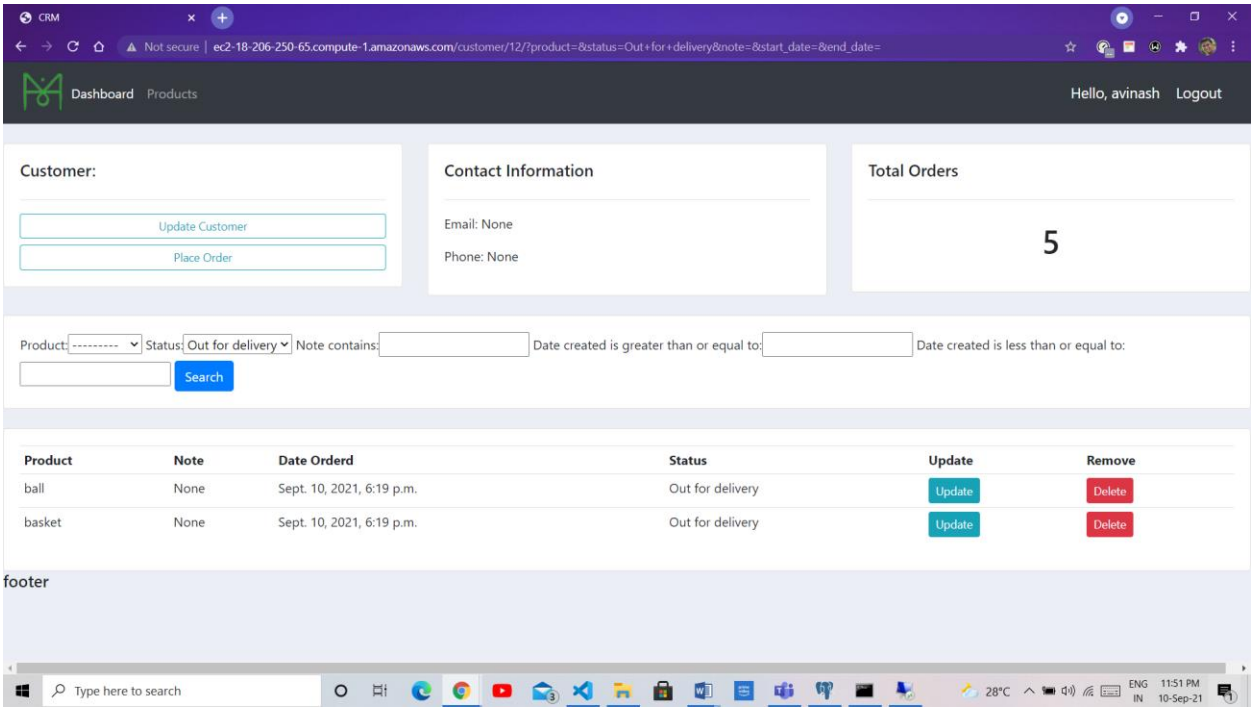


Fig 6.13 Searching by the status of the product

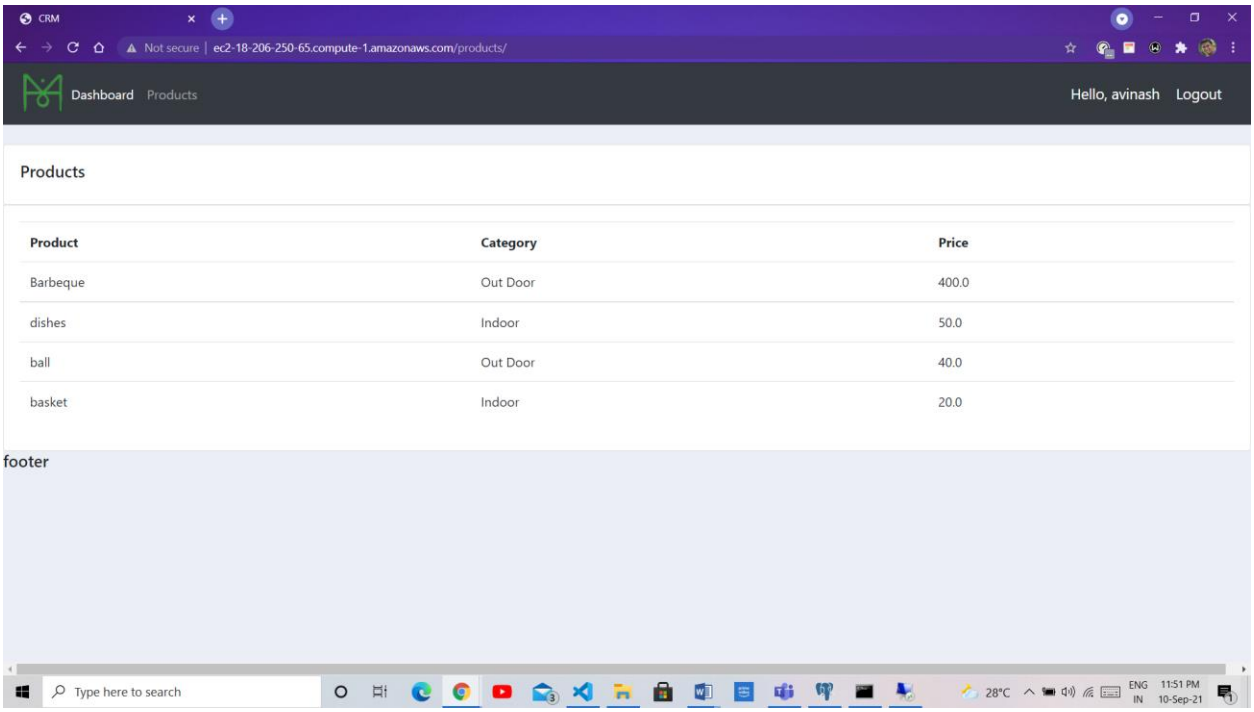


Fig 6.14 Products Page in Admin

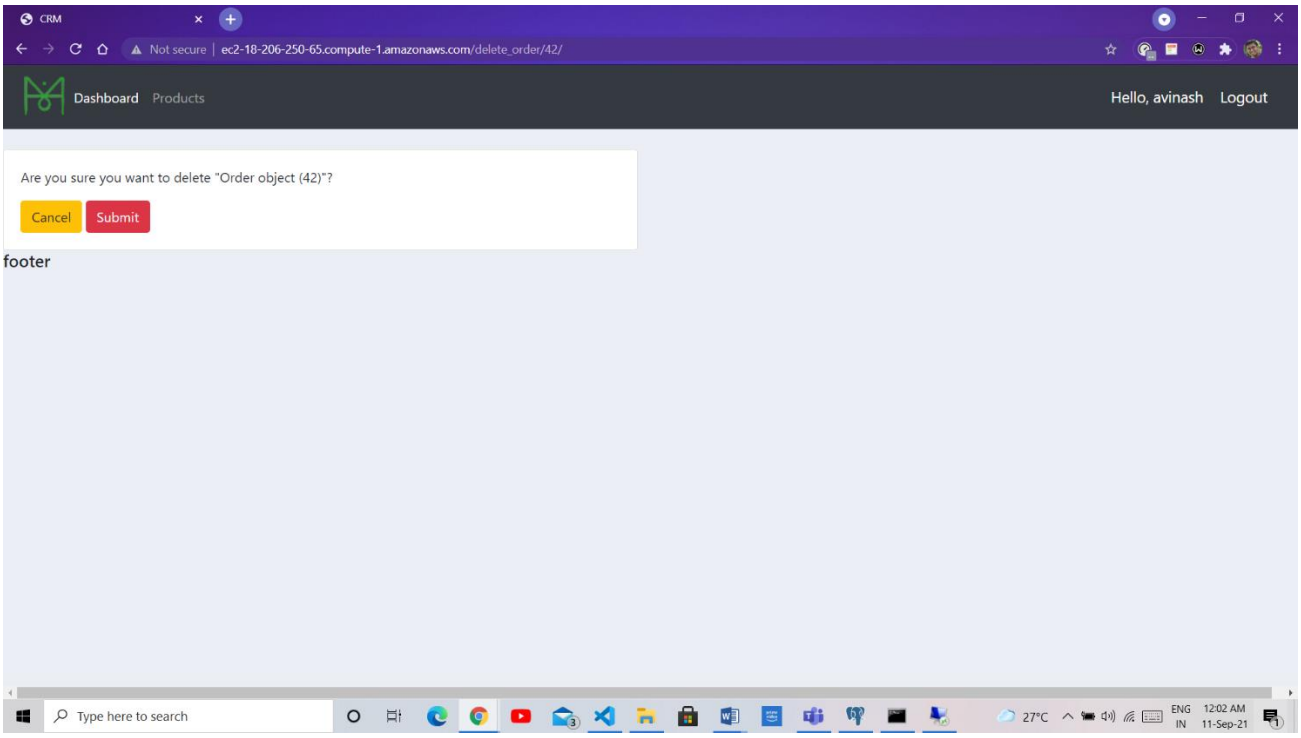


Fig 6.15 Deleting Order by Admin

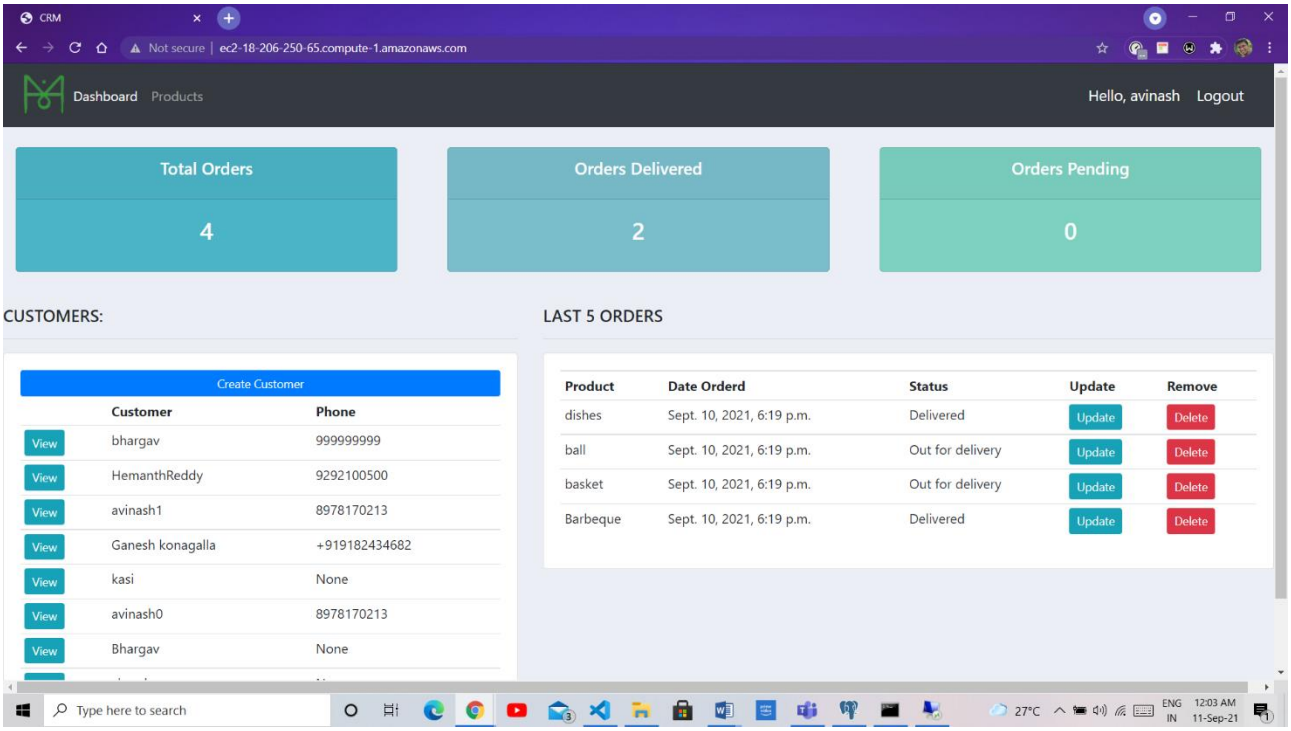


Fig 6.16 After Deleting Order by Admin

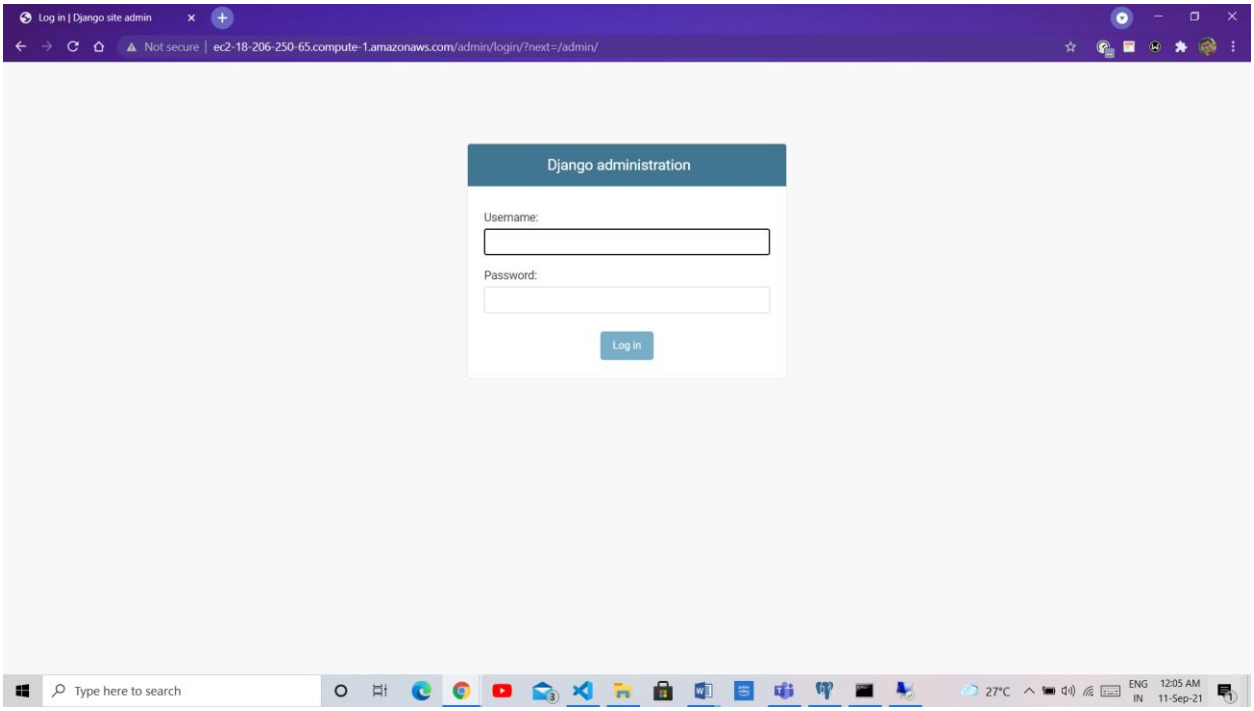


Fig 6.17 Django Admin panel

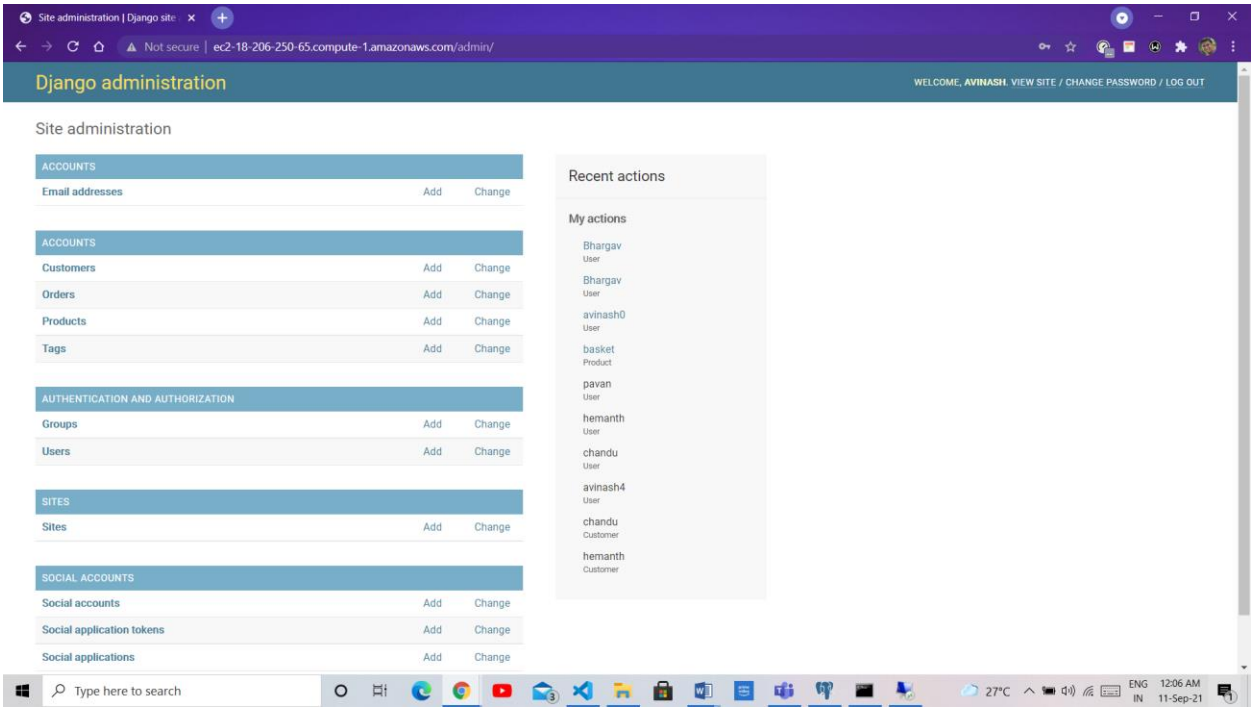


Fig 6.18 Django admin panel after Login

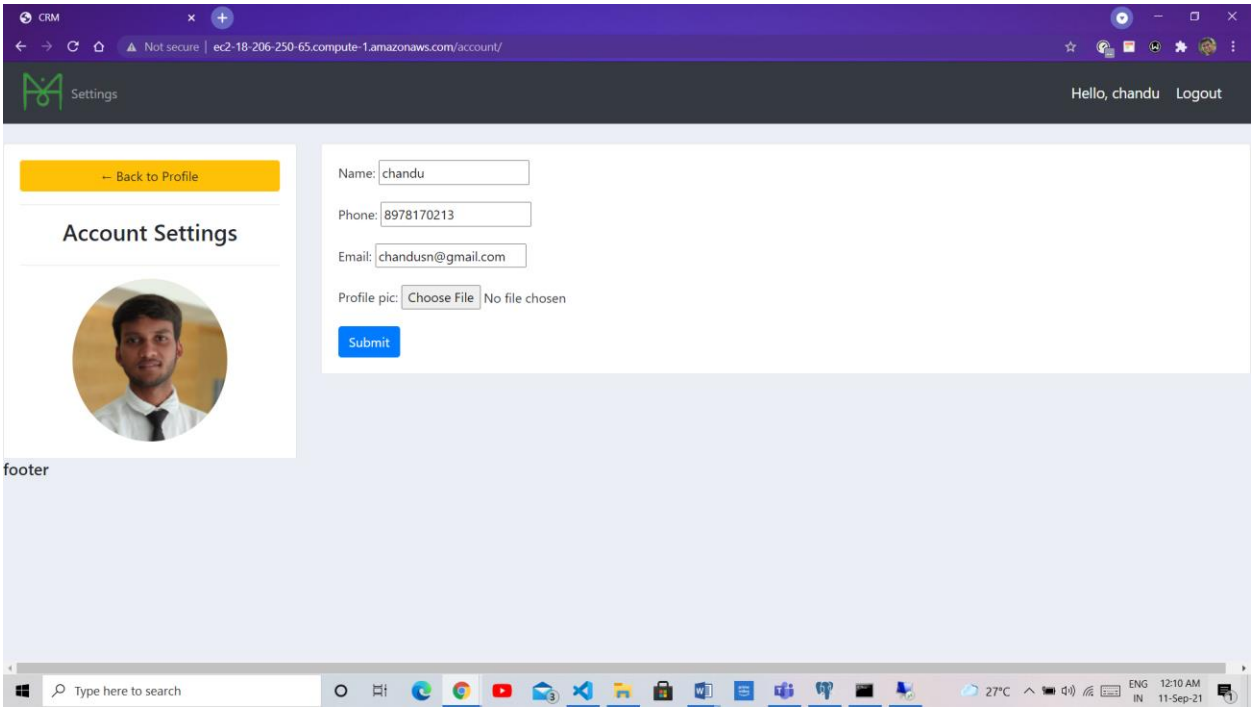


Fig 6.19 User Profile Page

Chapter 7

FUTURE SCOPE OF THE PROJECT

This project evaluates different services provided by the leading cloud provider, AMAZON. It shows how to create/own/access elastic compute cloud instance in all the possible ways. It deals with issues like bringing the machine up and down at any time and creating users in a secure way with authenticated keys, which are very useful for an organization like universities, non-IT based companies that are willing to join Amazon Cloud.

One of the future works of this project is to provide more information on better usage of the available resources on the cloud. Other works can be to incorporate all the services in different combinations (like EC2 with Cloud Front, S3, etc.) and then benchmark the best performance, Create machine images from scratch, Security of data on the cloud, etc.

Implementing Django related applications using just 3 basic services proves the migration of the cloud is possible in a very effective way in future.

Chapter 8

CONCLUSION

The Django framework gives us a simple and reliable way to create customer management system. It provides powerful functionalities and concise syntax to help programmers deal with the database, the web page and the inner logic. The experience of developing the group component in the system also helped me learning a lot of website development with Django. Within the Django framework, we have successfully accomplished the requirements of the system.

And also the usage of the cloud Resources makes this project most viable and it lead to the most intrigued solutions of the implementing and hosting the application. By this I conclude that the cloud platform makes the application secure and faster with minimal efforts and less cost with high efficient service.

Chapter 9

References

[1]Amazon Web Services (AWS). http://aws.amazon.com/

[2] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/

[3] http://clouddb.info/

[4]http://docs.amazonwebservices.com/AWSEC2/2009-10- 31/GettingStartedGuide/

[5]http://docs.amazonwebservices.com/AWSEC2/2009-10- 31/UserGuide/

[6] http://developer.amazonwebservices.com/connect/entry.jspa?e xternalID=351&categoryID=88

[7] http://developer.amazonwebservices.com/connect/entry.jspa?e xternalID=1233&categoryID=174

[8] Amazon Elastic Block Store. http://aws.amazon.com/ebs/

[9] http://docs.amazonwebservices.com/AWSEC2/2007-08- 29/GettingStartedGuide/putty.html

[10] https://d0.awsstatic.com/whitepapers/cloud-migration-main.pdf