Chapter 1

Company Profile

The internship was carried at Jawaharlal Nehru New College of Engineering, Shivamogga. It was carried out under Center of Innovation and Entrepreneurship (CIE) in collaboration with Ekathva Innovations Pvt. Ltd. The profile of CIE and Ekathva Innovations Pvt. Ltd. are discussed in following sections.

1.1 Center for Innovation and Entrepreneurship (CIE)

Center for Innovation & Entrepreneurship (CIE) was started in the year 2018 to jointly run various Platforms/Programs at JNNCE with the support of various Govt. Departments & agencies, Universities, Institutions, Industries, Startups, NES Management, JNNCE Faculty, Alumni & Students. CIE aims to create innovation, startup & Entrepreneurial ecosystem at JNNCE and in & around Shivamogga by providing all the required facilities like incubation support, startup space, Mentors connection, Support on IPR & Startup Registration, Investors Connection, assistance in building Business plan, Financial Projection, team building, product launch, marketing, scale-up etc., CIE also provides platform to setup Development Centers & Centers of Excellence by various startups, Industries, Agencies & MNCs at JNNCE Campus.

1.2 Ekathva Innovations Pvt. Ltd.

With the zeal of simplifying the day to day problems in the field of electronics, an Indian start-up company Ekathva Innovations Pvt., Ltd., came into mainstream. Company consists of electronics and computer science engineers who have enthusiasm in making day to day life better with their creative ideas. Company, which was established in mid of 2018 in the JNNCE Campus, Shivamogga in Karnataka state is targeting electronics and information technology field. Company mainly aims in making user friendly and economic products in the areas of design and development of the Embedded Systems, Printed circuit boards, Industrial automation, Home automation and automotive electronics which comes in the horizon of electronics whereas website development and app development both on android and iOS in the field of information technology. Company also banks on some in-house projects and always open for the projects proposed by clients.

1.3 Initiatives of CIE

NewGen IEDC: National Science and Technology Entrepreneurship Development Board (NSTEDB), Department of Science &amp; Technology (DST), Government of India has started New Generation Innovation and Entrepreneurship Development Centre (NewGen IEDC) at JNNCE in 2018 to ***"promote knowledge based and technology-driven start-ups by harnessing young minds and their innovation potential in an academic environment".*** Students will be encouraged to take up innovative projects with possibility of commercialization. Few amongst the "Job-Seekers" would be converted to "Job-Generators" through the entrepreneurial route. Best 10 innovative projects in Batch-1 (2019-20) have been selected recently in which each project team will get up to Rs. 2.5 Lakhs funding support to undergo their projects.

New Age Incubation Network (NAIN) Centre: Department of IT BT, Government of Karnataka has started "New Age Incubation Network" Centre at JNNCE in 2014 to build startup & business ecosystem in and around Shivamogga. Innovative Project proposals will be invited from JNNCE students, Alumni, Research Scholars & the public every year. For Top 10 innovative ideas, there will be a funding of up to Rs. 3.00 Lakhs per Project. Best 43 innovative projects in Batch-1 (2015-16), Batch-2 (2016-17), Batch-3 (2017-18) & Batch-4 (2019-20) have been funded to a maximum of Rs. 3.00 Lakhs per project.

JNNRIC: JNNCE Research & Innovation Centre aims to nurture innovation and encourage entrepreneurial talents among students, faculty and people of the region, and to facilitate inter-disciplinary research and coordinates with Incubation Center to bring up start-ups and promote budding entrepreneurs. JNNRIC might include research proposals addressing the Agricultural Technology, Renewable Energy and Energy Conservation System ICT for Socially relevant problems. JNNRIC will assist students Faculty, staff and students to boost the research and innovation by way of funding to a certain extent (seed money) every year as shown in Table 2.1

JNNCART: Centre for Agriculture & Rural Technology, a Centre for Appropriate and Agricultural Technologies was setup at JNNCE with the objective of serving farming

Table 1.1: CIE Project Funding

Sl.No	Project Type	Funding/Project (INR)	No. of projects	Total Funding (in INR)
1.	Innovative Project	Up to Rs. 25,000/-	10	2,50,000
2.	Research & Innovation Project	Up to Rs. 50,000/-	7	3,50,000
3.	Research Project	Up to Rs. 1,00,000/-	9	9,00,000

community and other people of the Malnad region. Students will be motivated to address problems faced by the farmers through innovative, cost-effective farming implements as a part of their project work as prescribed in academic curriculum.

CART nurtures innovation, and encourages entrepreneurial talents among students, faculty and *people of the region* to help agricultural needs of the farmers. It facilitates inter-disciplinary research and coordinates with incubation centre to bring up start-ups and budding entrepreneurs.

CART aims to Encourage students and faculty for innovation, idea generation and product development, provide seed fund to transform an idea into a product, arrange workshops for students to create awareness in the field of agri-based entrepreneurship, help with the patenting process, Inspire and support staff and students to achieve their potential and meet the challenges of society, facilitate development of at least two innovative products that are socially relevant. The five projects selected and executed for the year 2018-19 are

1. Arecanut Plucker cum Sprayer
2. Multipurpose Hybrid Sprayer
3. Mini Trencher Trenching Machine
4. Hydrophobic sand to combat water scarcity in irrigation
5. Trenching Machin

1.4 Additional Initiatives of CIE

Technology Business Incubator (TBI): Karnataka Council for Technological Upgradation (KCTU), Govt. of Karnataka has approved a TBI & will be started soon. The main objective of this TBI is to promote untapped creativity of individual innovators & to assist them to become technology-based entrepreneurs. JNNCE-TBI aims to provide common instrumentation facility, Mentors & Investors Connection, Co-Working Space, Office Space, Small Manufacturing Units, access to laboratories etc., for all associated individuals, innovators, startups, MSMEs & Companies.

JNNCE-MHRD Institution's Innovation Council: In association with MHRD's Innovation Cell (MIC) of AICTE, CIE JNNCE is working towards establishing an Institution's Innovation Council. Institution's Innovation Council (IIC) at the Institute is a unique model based on Hub-Spoke and coherence approach to align with the innovation and entrepreneurship promotion and support programs are being organized by various departments and ensures round the year activities in the campus for effective engagement, learning and practicing innovation and entrepreneurship among students and faculty community. Ideally, the Institution's Innovation Council is a faculty-led but student-centric body formed by the institute with the active representation of entrepreneurial faculties, students and experts representations from regional ecosystem enablers, pre-incubation and incubation centers within and outside the institute and work in synergize manner towards to provide a platform to encourage, inspire and nurture young students by exposing them to new ideas and process of resulting in innovative activities & entrepreneurial in their formative years.

Intellectual Property Cell (IP Cell): In association with KSCST, VTPC and InovaTree, CIE JNNCE is working towards establishing an IP Cell to provide all required information & guidance on IPR to Students, Alumni, Faculty, Research Scholars, Innovators, Startups, Companies, Industrialists, Academic Institutions & Universities in and around Shivamogga.

Entrepreneurship Cell (E-Cell): In association with E-Cell IIT Madras, CIE JNNCE is working towards establishing an E-Cell with the objective to develop spirit of Entrepreneurship among the students, to motivate students towards self-employment and entrepreneurship, to blend technical skills of students with entrepreneurial skills to convert their innovative ideas into enterprise. Activities of this cell include Conducting Awareness Programmes, Skill Development Programmes, Workshops, Business Plan Competitions, Boot Camps, Entrepreneurial Talks, Mentoring Students, taking up innovative projects, assisting in getting financial support, Company Incorporation, IPR etc., for Startups.

Technology Activities & Programs(TAP): A club of passion driven young students excelling together, either be it in the process of creating innovations, be it being active & rebellious dreamers in turning real time challenges prevailing in academics & to develop an individual in all possible ways by learning & sharing the skillsets. TAP is a knowledge sharing & knowledge gaining platform for all the interested students from different streams & semesters which helps every individual student to be an out-of-the-box thinker which helps to do innovative projects. Here the students can conduct group activities, undergo projects etc., Ideas/Projects developed from TAP teams will be supported by other platforms for further development & commercialization.

NAIN Creative Club: NAIN Creative Club, an initiative of NAIN Centre was started on 23rd September 2017 to provide a open platform for students to learn, excel & train students by conducting workshops & Seminars on various cutting edge technologies & languages. Club supports inter-departmental activities, which ignites the students to think out of box and helps to gain & explore creativity.

1.5 Startups under CIE

Ekathva Innovations Pvt. Ltd. was started in July 2018 by Mr. Vikas H C & team from NAIN first batch

CIE also has following development centres:

1. RootsGoods Development Centre by RootsGoods OPC Pvt. Ltd.
2. Sunrise Digital Media Development Centre by Sunrise Digital Media Pvt. Ltd.
3. Ventalyst Development Centre by Ventalyst Pvt. Ltd.
4. Silfra Tech Development Centre by Silfra Technologies Pvt. Ltd.

Many other Innovators, Project Teams, Startups, MSMEs & Companies are in discussions and in pipeline to setup Startups, Development Centers & Centers of Excellence at JNNCE under CIE. CIE also is in the process of establishing JNNCE Research Park with the objective of pursuing research in niche and cutting-edge technologies and science.

1.6 Members of CIE

1. Mr. Mallesh Kumar K S, Project Manager, Center for Innovation & Entrepreneurship Incubation Centre Manager, New Age Incubation Network Centre Jawaharlal Nehru New College of Engineering, Shivamogga

2. Mr. Nrupatunga C M, Project Coordinator, NewGen IEDC

3. Dr. Manjunath P Prof. & Head, Dept. of ECE College Coordinator, NAIN Centre

4. Dr. Basappaji, Prof. Dept. of Mechanical Engineering, Chief Coordinator, NewGen IEDC

5. Dr. E Basavaraj Professor, Dept. of Mechanical Engineering, Co-Ordinator, JNNCART

6. Dr. Karthika B S, Assoc. Prof. and Head, Dept. of Civil Engineering, Convener, JNNRIC

1.7 Members of Ekathva Innovations

With the ideas of innovation in creating the better future in the ever-changing field of electronics and IT, company wants to be pioneer in some of the products which will change the perspective of everyday life to the layman too. With the motto of invent for betterment, Ekathva aims to strive hard in bringing out the economic, uncomplicated, simple products to the world.

Ekathva Innovations currently having 6 members in team 4 directors and 2 employees

1. Vikas H C Co-Founder, CEO and Embedded Developer

2. Akshay K Kulkarni Co-Founder, COO and Full Stack Software Developer

3. Koushik R Udupa Co-Founder and Embedded Developer
4. Akshay K k Co-Founder and CFO
5. Rudresh P V Full Stack Developer
6. Sachin MS Full Stack Developer

They are currently working on a Embedded Product which is under patent process and on Information Technology services like Website Development, App Development and Customized Software Development are provided.

1.8 Organization of the report

Chapter [1] provides the details about the company, initiatives from of CIE, startups under CIE and its members and also members of the company Ekathva Innovations.
Chapter [2] provides task performed during the internship the tasks include training on technologies like Django, Flask, MongoDB, Cassandra, PySpark, Jinja2 and its templates.
Chapter [3] provides the results of the mini project tasked during the end of the internship and reflection notes, meeting conducted with guide and the experience of the internship.
Chapter [4] provides the conclusion of the internship.
Final chapter is the bibliography.

Summary

The profile of the company Ekathva innovations Pvt Ltd and its members working in the company has been briefed and the initiatives of CIE and the startups has been briefed in chapter.

Chapter 2

Tasks Performed

This internship gave an opportunity to learn various technologies of Python. Python is the popular programming language and has been used in myriad of applications. Full Stack Python is used in Django helps in developing complex web applications and achieving security at the same time. In this internship, a hands-on usage of Python programming in Django framework was used. Further popular NoSQL cloud databases like Firebase and MongoDB were also explored. Python interfacing with these databases were leant. In case of front-end GUI development for different applications like Machine learning, Artificial Intelligence, Computer vision and others are required, micro-framework like Flask can be used. This internship also facilitates learning of developing front-end using Flask microframework with Python. These technologies are detailed in following sections along with assignments/tasks performed, meeting conducted for carrying out project work and soft skills acquired during training.

2.1 Technical Skills

In this internship of four weeks, initial few days laid focus on the learning of Python programming. Different programming/interpretive expressions related tasks on Python were solved in the class. Next 10 days was dedicated on learning of full stack Python using Django framework in Visual Studio Code Environment. Next few days of internship introduced to world of Firebase and MongoDB and their interfacing with Python. The last week of internship was focused on Flask micro web framework and development of applications using it.

2.1.1 Django framework

2.1.1.1 Overview

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. The reasons for selection of this full stack Python framework are:

- 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.

- It makes it easy to display, validate and re-display 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 re-display (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 that one 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 the work to be more productive, at a higher level of abstraction, than if coding against, say, HTTP. But it doesn’t restrict you from going “down” one level of abstraction when needed

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. Finally, Django strictly maintains a clean design throughout its own code and makes it easy to follow best Web-development practices in the applications you create.

2.1.1.2 Django MVT

The MVT (Model View Template) is a software design pattern. It is a collection of three important components Model View and Template. The Model helps to handle database. It is a data access layer which handles the data. The Template is a presentation layer which handles User Interface part completely. The View is used to execute the business logic and interact with a model to carry data and renders a template.There is no separate controller and complete Django application is based on Model View and Template. Hence it is called MVT application. MVT based control flow is depicted in Figure 3.1.

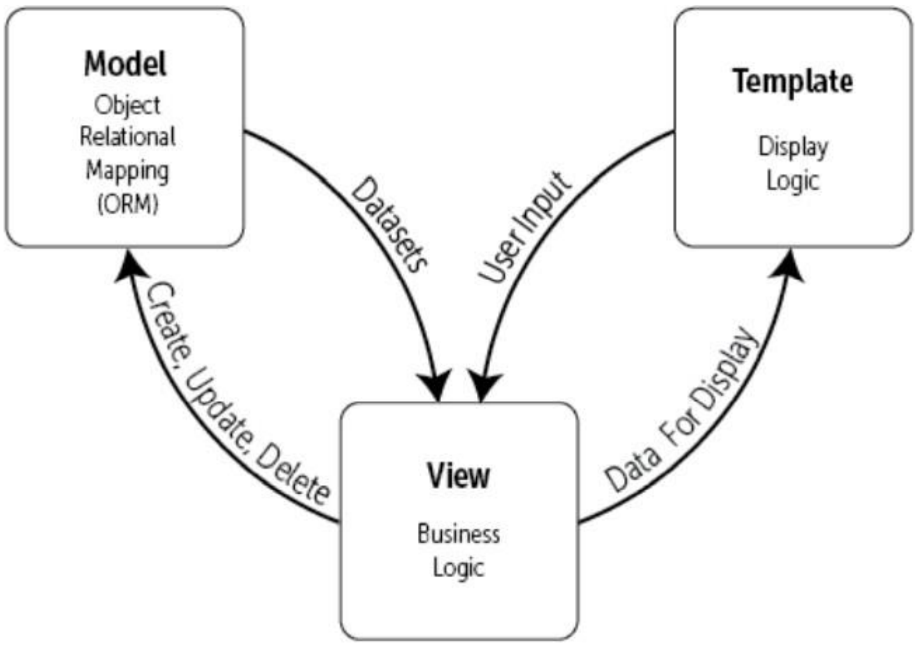


Figure 2.1: Django Architecture

Here, a user requests for a resource to the Django, Django works as a controller and check to the available resource in URL. If URL maps, a view is called that interact with model and template, it renders a template. Django responds back to the user and sends a template as a response.

2.1.1.3 Django Views

Views in the python file where functions/classes in Python is defined. Views can be static or dynamic based on the content they deliver. An example views.py is shown in the following listing:

```
from django.shortcuts import render_to_response
from mysite.polls.models import Poll

def index(request):
latest_poll_list = Poll.objects.all().order_by('−pub_date')[
    :5]    return
render_to_response('polls/index.html', {'poll_list':poll_lis
    t})
```

2.1.1.4 Django Templates

The Django template language consists of tags, which perform many functions and may be embedded in a text file to do neat tricks. An example html file is shown in the following listing:

```
<html >
<h1 >{{ poll . question }} </ h1>
<ul >
{% for choice in poll . choice_set . all %}
< li >{{ choice . choice }} –– {{ choice . votes }} vote {{ choice . v
otes | pluralize }} </ li >
{% endfor %}
</ ul >
</ html >
```

2.1.1.5 Django Models

Models store data for your app and are the key for making dynamic websites. Models are implemented as Python classes, in models.py file. An example models.py is shown in the following listing:

```
from django . db import models class Poll( models . Model ) :
question = models . CharField ( max_length =200)
pub_date = models . DateTimeField ( ' date published ')

class Choice ( models . Model ) :
poll = models . ForeignKey ( Poll )
choice = models . CharField ( max_length =200) votes = models . Inte
gerField ()
```

2.1.1.6 Django Forms

It automatically generates form widgets, Input validation, Re-display a form after invalid input and convert submitted form data to Python data types. An example program is show in the following listing:

```
<h1 >{{ poll . question }} </ h1>
{% if error_message %}<p>< strong >{{ error_message }} </ strong
> </p>{% en
<form action ="/ polls /{{ poll . id }}/ vote /" method =" post ">
{% csrf_token %}
{% for choice in poll.choice_set.all %}
<input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}" />
<label for="choice{{ forloop.counter }}">{{ choice.choice }}</label><br />
{% endfor %}
<input type="submit" value="Vote" />
</form>
```

2.1.2 MongoDB

MongoDB is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document. Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases. Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose. A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

2.1.2.1 Need of MongoDB

MongoDB is always preferred by the developers or project managers when our main concern is the deal with large volume of data with a high performance. If we want to insert thousands of records in a second, then MongoDB is the best choice for that. Also, horizontal scaling (adding new columns) is not so easy process in any RDBMS systems. But in case of MongoDB, it is very much easy since it is a schema less database. Also, this type of work can be directly handled by the application automatically. There is no need to any type of administrative work for perform any type of horizontal scaling in the MongoDB.

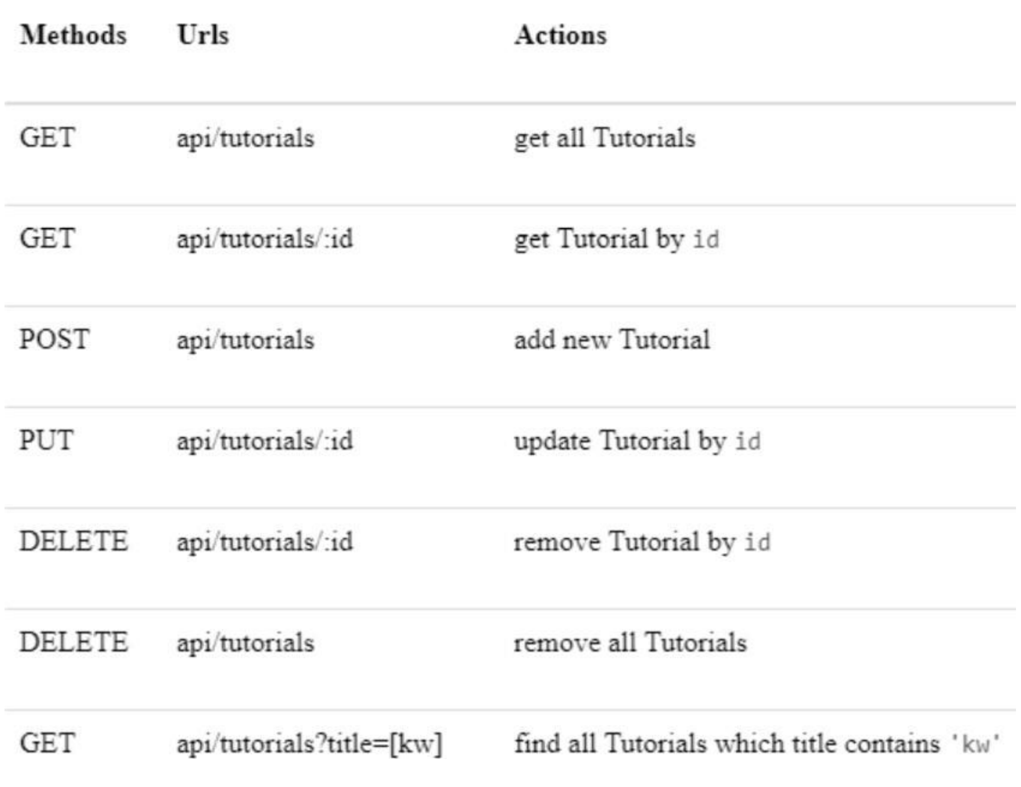
2.1.2.2 Mongo Compass and Mongo Atlas

Developers describe MongoDB Atlas as "Deploy and scale a MongoDB cluster in the cloud with just a few clicks". MongoDB Atlas is a global cloud database service built and run by the team behind MongoDB. Enjoy the flexibility and scalability of a document database, with the ease and automation of a fully managed service on your preferred cloud.

On the other hand, MongoDB Compass is detailed as "A GUI for MongoDB". Visually explore your data. Run ad hoc queries in seconds. Interact with your data with full CRUD functionality. View and optimize our query performance. MongoDB Atlas belongs to "MongoDB Hosting" category of the tech stack, while MongoDB Compass can be primarily classified under "Database Tools".

2.1.2.3 Djongo and REST API

We will build Rest APIs using Django Rest Framework that can create, retrieve, update, delete and find Tutorials by title or published status. First, we setup Django Project with a MongoDB Connector. Next, we create Rest Api app, add it with Django Rest Framework to the project. Next, we define data model and migrate it to the database. Then we write API Views and define Routes for handling all CRUD operations (including custom finder). The following table shows overview of the Rest APIs that will be exported:



Methods	Urls	Actions
GET	api/tutorials	get all Tutorials
GET	api/tutorials/:id	get Tutorial by `id`
POST	api/tutorials	add new Tutorial
PUT	api/tutorials/:id	update Tutorial by `id`
DELETE	api/tutorials/:id	remove Tutorial by `id`
DELETE	api/tutorials	remove all Tutorials
GET	api/tutorials?title=[kw]	find all Tutorials which title contains `'kw'`

Figure 2.2: Rest APIs

2.1.3 Flask microframework

Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by Armin Ronacher who leads an international group of python enthusiasts (POCCO). Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

2.1.3.1 Use of Flask

Companies that use Flask are Netflix, Lyft, Reddit, Zillow, MailGui and so on. It was made by using around 10000 lines of source code. It is used to develop simple web applications, microservices, and "serverless" platforms. It provides URL routing, request & error handling, and a development server. Flask is a simple, lightweight, and easy to use a framework. It includes less built-in functionality than Django. But it provides facility to a web developer to keep the core of a web application extensible and straightforward. Flask doesn't have a database layer, no ORM, supports NoSQL, perform database operations through SQLAlchemy. Flask has built-in security against the number of common threats such as CSRF, XSS, and SQL injection.

2.1.3.2 Flask Environment

In order to set the environment and debug mode reliably, Flask uses environment variables. The environment is used to indicate to Flask, extensions, and other programs, like Sentry, what context Flask is running in. It is controlled with the FLASK_ENV environment variable and defaults to production. Setting FLASK_ENV to development will enable debug mode. flask run will use the interactive debugger and reloader by default in debug mode. To control this separately from the environment, use the FLASK_DEBUG flag. Using the environment variables as described above is recommended. While it is possible to set ENV and DEBUG in your config or code, this is strongly discouraged. They can't be read early by the flask command, and some systems or extensions may have already configured themselves based on a previous value.

2.1.3.3 Dynamic Web page

Flask facilitates us to add the variable part to the URL by using the section. We can reuse the variable by adding that as a parameter into the view function. The url_for() function is used to build a URL to the specific function dynamically. The first argument is the name of the specified function, and then we can pass any number of keyword argument corresponding to the variable part of the URL. This function is useful in the sense that we can avoid hard-coding the URLs into the templates by dynamically building them using this function. Consider the following example:

```
@app.route('/hod')def hod():
return 'HOD'

@app.route('/principal')def principal():
return 'Principal'

@app.route('/student')def student():
return 'student'

@app.route('/user/<name>')defuser(name):
if name == 'HOD':
return redirect(url_for('hod'))if name == 'Principal':
return redirect(url_for('principal'))if name == 'student':
return redirect(url_for('student'))
```

2.1.3.4 Jinja2 Templates and static

Jinja2 is a modern day templating language for Python developers. It was made after Django's template. It is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request. A template contains variables which are replaced by the values which are passed in when the template is rendered. Variables are helpful with the dynamic data. This a example html file:

```
<html>
<head>
<title>print table</title>
</head>
<body>   <h2> printing table of {{n}}</h2>
{% for i  in range(1,11): %}
<h3>{{n}} X {{i}} = {{n * i}} </h3>
{% endfor %}
</body>
</html>
```

The static files such as CSS or JavaScript file enhance the display of an HTML web page. A web server is configured to serve such files from the static folder in the package or the next to the module. The static files are available at the path /static of the application. Example for static file that is Style.css is shown below:

```
body  {          background−color:  powderblue;        }
h2  {          color:  white;        }
h3  {          color:brown;        }
```

2.1.4 Cassandra

Apache Cassandra is an open source, distributed and decentralized/distributed storage system (database), for managing very large amounts of structured data spread out across the world. It provides highly available service with no single point of failure.

Listed are some of the notable points of Apache Cassandra − It is scalable, fault-tolerant, and consistent, It is a column-oriented database, Its distribution design is based on Amazon’s Dynamo and its data model on Google’s Bigtable, Created at Facebook, it differs sharply from relational database management systems, Cassandra implements a Dynamo-style replication model with no single point of failure, but adds a more powerful “column family” data model, Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

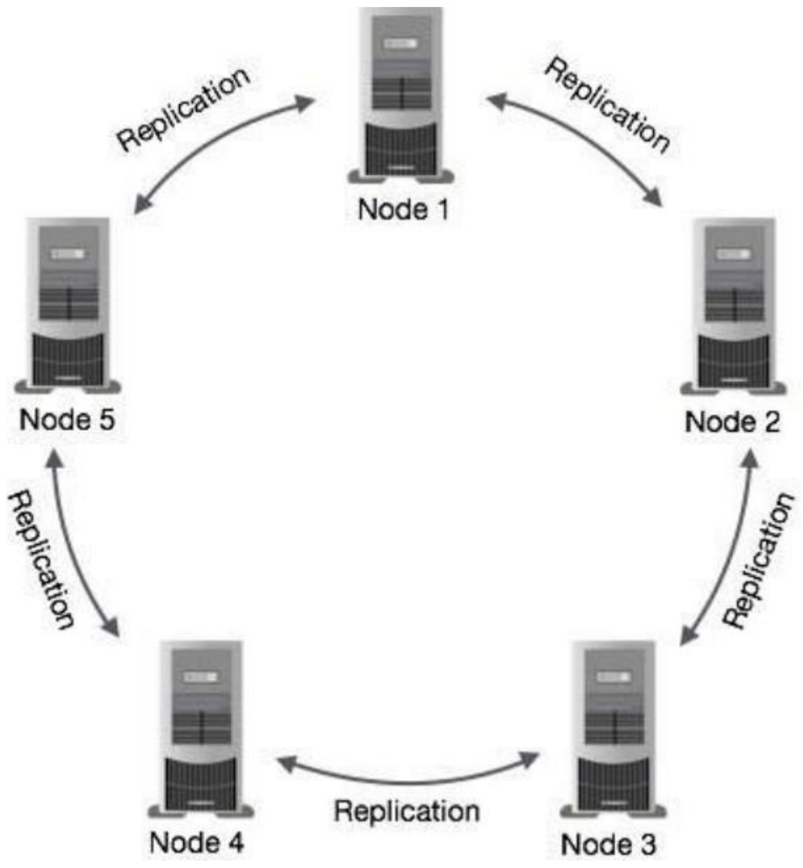


Figure 2.3: Cassandra Architecture

2.1.4.1 CQL

Users can access Cassandra through its nodes using Cassandra Query Language (CQL). CQL treats the database (Keyspace) as a container of tables. Programmers use cqlsh: a prompt to work with CQL or separate application language drivers. Client's approach any of the nodes for their read-write operations. That node (coordinator) plays a proxy between the client and the nodes holding the data.

Write Operations: Every write activity of nodes is captured by the commit logs written in the nodes. Later the data will be captured and stored in the mem-table. Whenever the mem-table is full, data will be written into the SStable data file. All writes are automatically partitioned and replicated throughout the cluster. Cassandra periodically consolidates the SSTables, discarding unnecessary data.

Read Operations: During read operations, Cassandra gets values from the mem-table and checks the bloom filter to find the appropriate SSTable that holds the required data.

CQL Data Definition Commands:

CREATE KEYSPACE − Creates a KeySpace in Cassandra. USE − Connects to a created KeySpace.

ALTER KEYSPACE − Changes the properties of a KeySpace. DROP KEYSPACE − Removes a KeySpace

CREATE TABLE − Creates a table in a KeySpace.

ALTER TABLE − Modifies the column properties of a table. DROP TABLE − Removes a table.

TRUNCATE − Removes all the data from a table.

CREATE INDEX − Defines a new index on a single column of a table. DROP INDEX − Deletes a named index.

CQL Data Manipulation Commands:

INSERT − Adds columns for a row in a table. UPDATE − Updates a column of a row.

DELETE − Deletes data from a table.

BATCH − Executes multiple DML statements at once. CQL Clauses:

SELECT − This clause reads data from a table

WHERE − The where clause is used along with select to read a specific data.

ORDERBY − The orderby clause is used along with select to read a specific data in a specific order

2.1.4.2 DataStax driver

A Python client driver for Apache Cassandra®. This driver works exclusively with the Cassandra Query Language v3 (CQL3) and Cassandra's native protocol. Cassandra 2.1+ is supported, including DSE 4.7+. The driver supports Python 2.7, 3.4, 3.5, 3.6, 3.7 and 3.8. This driver is open source under the Apache v2 License. The source code for this driver can be found on GitHub. Before we can start executing any queries against a Cassandra cluster we need to setup an instance of Cluster. As the name suggests, you will typically have one instance of Cluster for each Cassandra cluster you want to interact with.

2.1.5 PySpark for Big Data Analytics

PySpark is a great language for performing exploratory data analysis at scale, building machine learning pipelines, and creating ETLs for a data platform. Apache Spark is written in Scala programming language. To support Python with Spark, Apache Spark community released a tool, PySpark. Using PySpark, you can work with RDDs in Python programming language also. It is because of a library called Py4j that they are able to achieve this. This is an introductory tutorial, which covers the basics of Data-Driven Documents and explains how to deal with its various components and sub-components.

2.1.5.1 RDD

RDD stands for Resilient Distributed Dataset, these are the elements that run and operate on multiple nodes to do parallel processing on a cluster. RDDs are immutable elements, which means once you create an RDD you cannot change it. RDDs are fault tolerant as well, hence in case of any failure, they recover automatically. You can apply multiple operations on these RDDs to achieve a certain task. To apply operations on these RDD’s, there are two ways − Transformation and Action.

Transformation − These are the operations, which are applied on a RDD to create a new RDD. Filter, groupBy and map are the examples of transformations.

Action − These are the operations that are applied on RDD, which instructs Spark to perform computation and send the result back to the driver.

2.1.5.2 DataFrames

A DataFrame is an immutable distributed collection of data that is organized into named columns analogous to a table in a relational database. Introduced as an experimental feature within Apache Spark 1.0 as SchemaRDD, they were renamed to DataFrames as part of the Apache Spark 1.3 release. By imposing a structure onto a distributed collection of data, this allows Spark users to query structured data in Spark SQL or using expression methods (instead of lambdas). By structuring your data, this allows the Apache Spark engine – specifically, the Catalyst Optimizer – to significantly improve the performance of Spark queries. In earlier APIs of Spark (that is, RDDs), executing queries in Python could be significantly slower due to communication overhead between the Java JVM and Py4J. It also shares some common characteristics with RDD:

Immutable in nature: We can create DataFrame / RDD once but can't change it. And we can transform a DataFrame / RDD after applying transformations.

Lazy Evaluations: Which means that a task is not executed until an action is performed.

Distributed: RDD and DataFrame both are distributed in nature.

2.1.5.3 MLLib

MLlib stands for Machine Learning Library. In addition, MLlib is currently the only library that supports training models for streaming. At the high level, MLlib exposes three core machine learning functionalities: Data preparation: Feature extraction, transformation, selection, hashing of categorical features, and some natural language processing methods Machine learning algorithms: Some popular and advanced regression, classification, and clustering algorithms are implemented Utilities: Statistical methods such as descriptive statistics, chi-square testing, linear algebra (sparse and dense matrices and vectors), and model evaluation methods

2.1.5.4 Graph Frames

Graphs are an interesting way to solve data problems because graph structures are a more intuitive approach to many classes of data problems. Whether traversing social networks or restaurant recommendations, it is easier to understand these data problems within the context of graph structures: vertices, edges, and properties. GraphFrames utilizes the power of Apache Spark DataFrames to support general graph processing. Specifically, the vertices and edges are represented by DataFrames allowing us to store arbitrary data with each vertex and edge. While GraphFrames is similar to Spark&#39;s GraphX library, there are some key differences, including: GraphFrames leverage the performance optimizations and simplicity of the DataFrame API. By using the DataFrame API, GraphFrames now have Python, Java, and Scala APIs. GraphX is only accessible through Scala; now all its algorithms are available in Python and Java.

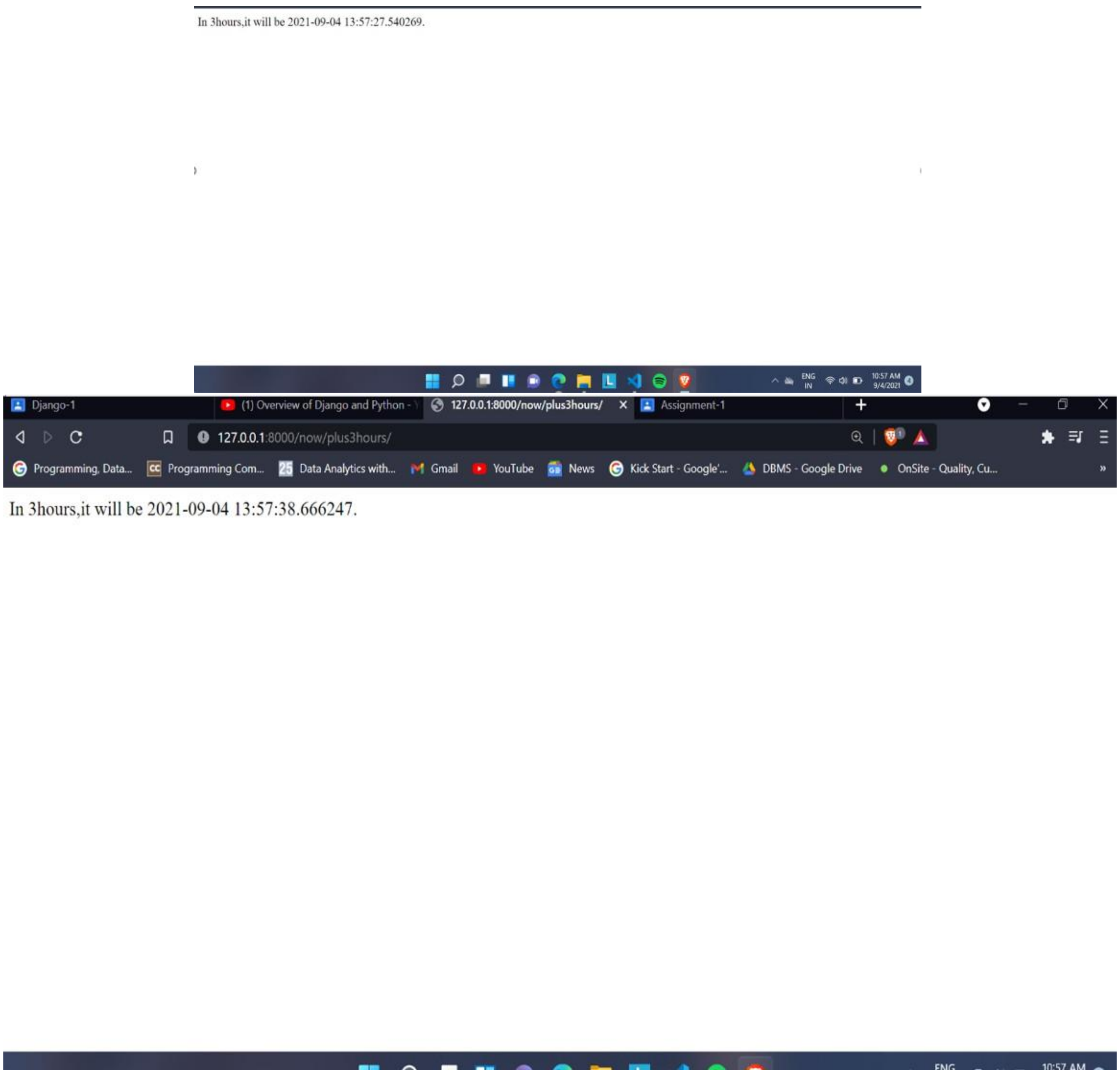
2.2 Assignments

List of assignments:

1) Displaying Date and Time after all modification

```
from django.shortcuts import render
from django.http import HttpResponse import datetime
```

```
# C r e a t e  your  views   here.

 d e f  home( r e q u e s t ) :
r e t u r n  Http Response  ( " Hello ,  Django ! " ) d e f      g r e e t ( r e q u e s t ) :
dt=d a t e t i m e . d a t e t i m e . now()
r e t u r n  Http Response (" < h1 > C u r r e n t d a t e  and  tim e  is :"+ s t r ( dt )
+" </ h1 >") d e f                 h o u r _ o f f s e t ( r e q u e s t ,  p l u s_or_minus
,    o f f s e t ) :
o f f s e t  =  i n t ( o f f s e t ) i f    o f f s e t  ==  1 :
hours  =  ' hour ' e l s e :
hours  =  ' hours '
i f  p l u s_o r_m i n u s  ==  ' p l u s ' :
dt  =  d a t e t i m e . d a t e t i m e . now()  +  d a t e t i m e . t i m e d e l t a ( hours=o f f s
e t ) o u t p u t  =  ' In  %s  %s ,   i t    w i l l                  be %s . '  %  ( o f
f s e t ,                  hours ,   d t )
e l s e :
dt  =  d a t e t i m e . d a t e t i m e . now()  −  d a t e t i m e . t i m e d e l t a ( hours=o f f s
e t ) o u t p u t  =  '%s  %s  ago ,   i t                   was %s . '  %  ( o f f s
e t ,                   hours ,                   d t )
o u t p u t  =  '<html ><body>%s </ body > </ html > '  %  o u t p u t r e t u r n
Http Response ( o u t p u t )
```



```
In 3hours,it will be 2021-09-04 13:57:27.540269.
```

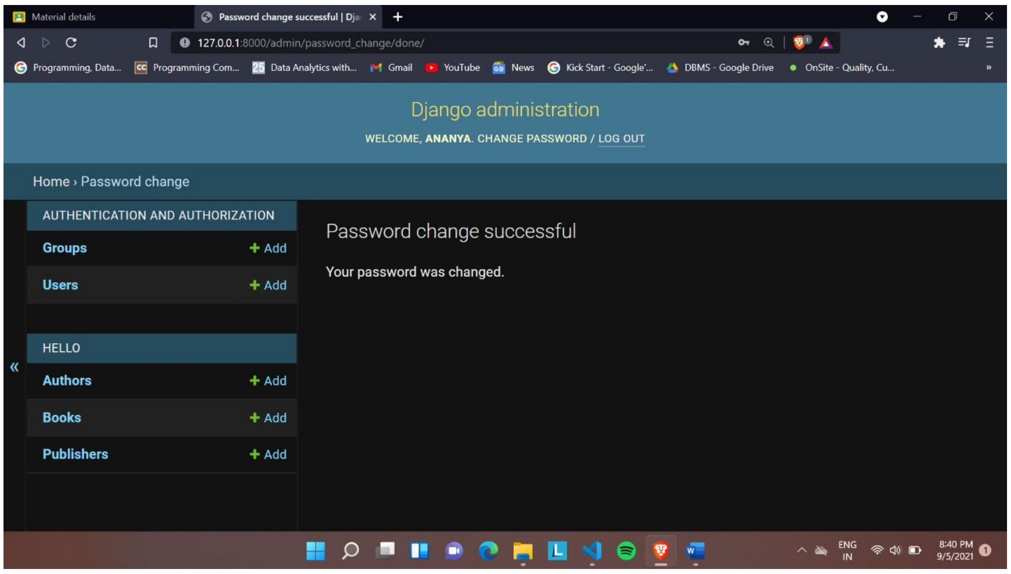
```
In 3hours,it will be 2021-09-04 13:57:38.666247.
```

Figure 2.4: Snapshots of Assignment 1

2) Displaying Django Website after all modification from django.db import models

```
# Create your models here.
class Publisher(models.Model):
name = models.CharField(max_length=30)
address = models.CharField(max_length=50) city = models.CharField(max_length=60)
state_province = models.CharField(max_length=30) country = models.CharField(max_length=50)
website = models.URLField()

class Author(models.Model):
salutation = models.CharField(max_length=10)
```

```
first_name = models . CharField ( max_length =30) last_name =
models . CharField ( max_length =40) email = models . EmailField ( bl
ank =True )
headshot = models . ImageField ( upload_to = ’img ’)

class Book ( models . Model ):
title = models . CharField ( max_length =100) authors = models .
ManyToManyField ( Author )
publisher = models . ForeignKey ( Publisher , on_delete=models .
CASCADE) publication_date = models . DateField ()

Admin . py :
from django . contrib import admin
from hello . models import Book , Author , Publisher# Register
    your models         here .
admin . site . register (Book)
#admin . site . register ( Author ) @admin . register ( Author )
class AuthorAdmin ( admin . ModelAdmin ):
#fields = (( ’first_name ’, ’last_name ’) , ’email ’) ordering = (
’first_name ’ ,)
search_fields = ( ’first_name ’, ’email ’) list_filter     = ( ’fi
rst_name ’, ’email ’)
fieldsets = (           ( ’Required Information ’,
{ ’description ’: "These fields are required for each event."
,
 ’fields ’: (( ’first_name ’, ’last_name ’) , ’salutation ’, ’headsh
 ot ’)
    }) ,         ( ’Optional Information ’, {
’classes ’: ( ’collapse ’ ,) ,             ’fields ’: ( ’email ’ ,)
        }) ,     )
admin . site . register ( Publisher )
```


Django administration

WELCOME, ANANYA. CHANGE PASSWORD / LOG OUT

Home › Password change

AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add

HELLO
Authors + Add
Books + Add
Publishers + Add

Password change successful

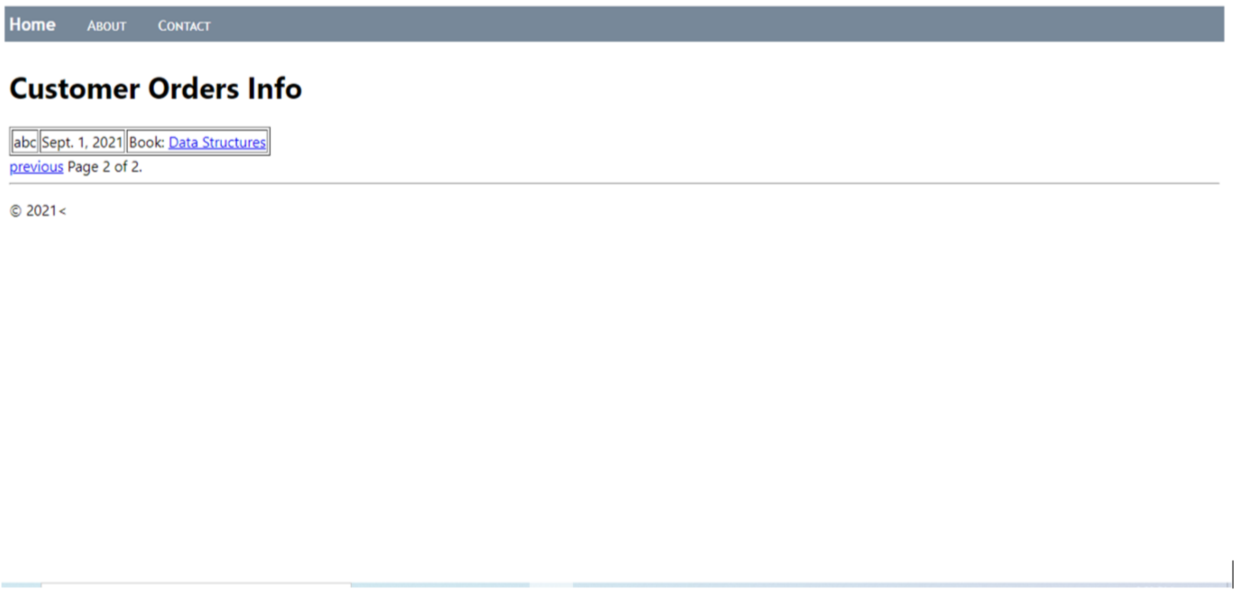
Your password was changed.

Figure 2.5: Snapshot of Assignment 2

3) Displaying Customer orders list and Book Information from django.shortcuts

```
  import              render
from django.http import HttpResponse
from django.template import Template, Contextimport  dateti
me
from django.views import generic
from hello.models import Book, Author, Ordersfrom django.htt
p    import     HttpResponseRedirect
from hello.contact import ContactForm

class OrdersListView(generic.ListView):model = Orders
template_name = 'orders_list.html'paginate_by = 1

class BookDetailView(generic.DetailView):model = Book
template_name = 'book_detail.html'

class BookListView(generic.ListView):model = Book
template_name = 'book_list.html'paginate_by = 1

class AuthorDetailView(generic.DetailView):
```

```
model  =  Author
template_name  =  ’author_detail.html’
```



Customer Orders Info

abc	Sept. 1, 2021	Book: Data Structures

previous Page 2 of 2.

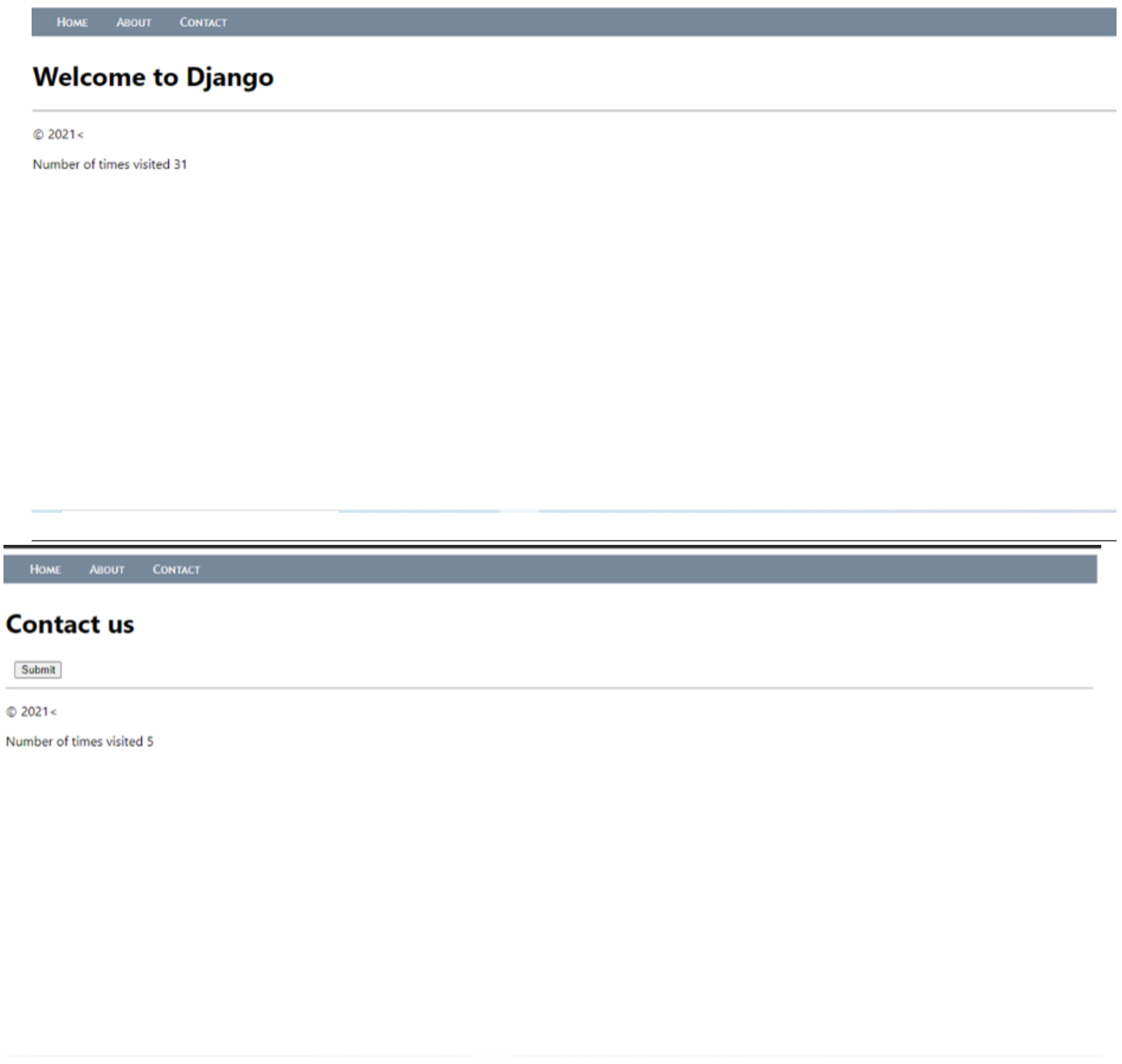
© 2021<

Figure 2.6: Snapshot of Assignment 3

4) Displaying View count for All Pages

```
from  django.shortcuts importrender, redirectfrom  django.htt
p                       import HttpResponse
from  django.template  import  Template ,  Context
importdatetimefrom  django.views  import genericfrom    hello.
models  import  Book,                  Author ,  Orders
from  django.http  import HttpResponseRedirectfrom  hello.cont
act   import          ContactForm
from  django.core.files.storage  import FileSystemStoragefrom h
ello.oforms           import  DocumentForm
def  home(request):
nv  =  request.session.get(’nv’,  0)request.session[’nv’]  =  nv
+  1
return render(request,"home.html",{’nv’:request.session[’nv’
]    })def                                                     about(re
av  =  request.session.get(’av’,  0)request.session[’av’]  =  av
+ 1
return render(request,"about.html",{’nv’:request.session[’av
’]    })def                                                     contact
cv  =  request.session.get(’cv’,  0)
```

```
request.session['cv'] = cv + 1
return render(request,"contact.html",{'nv':request.session['
cv']})def                                                    visito
if request.COOKIES.get('lv') is not None:value = request.
COOKIES.get('lv')
html = HttpResponse
("<center><h1><br>You have last visited on "+value)html.set_
cookie('lv',          datetime.datetime.now())
return html else:
value=1
html=HttpResponse("<center><h1>First Time</h1>") html.set_co
okie('lv',          value)
return html
def simple_upload(request):
if request.method == 'POST' and request.FILES['myfile']:myfi
le = request.FILES['myfile']
fs = FileSystemStorage()
filename = fs.save(myfile.name, myfile)uploaded_file_url = f
s.url(filename)
return render(request, 'supl.html',
{
'uploaded_file_url':uploaded_file_url        })return    r
ender(request,        'supl.html')
def model_form_upload(request):
if request.method == 'POST':
form = DocumentForm(request.POST, request.FILES)if      form.i
s_valid():
form.save()
return redirect('/up')else:
form = DocumentForm()
return render(request, 'model_form_upload.html', {'form':
       form        })
```



Welcome to Django

© 2021<

Number of times visited 31

Contact us

© 2021<

Number of times visited 5

Figure 2.7: Snapshot of Assignment 4

5) Displaying Mongo Shell Commands

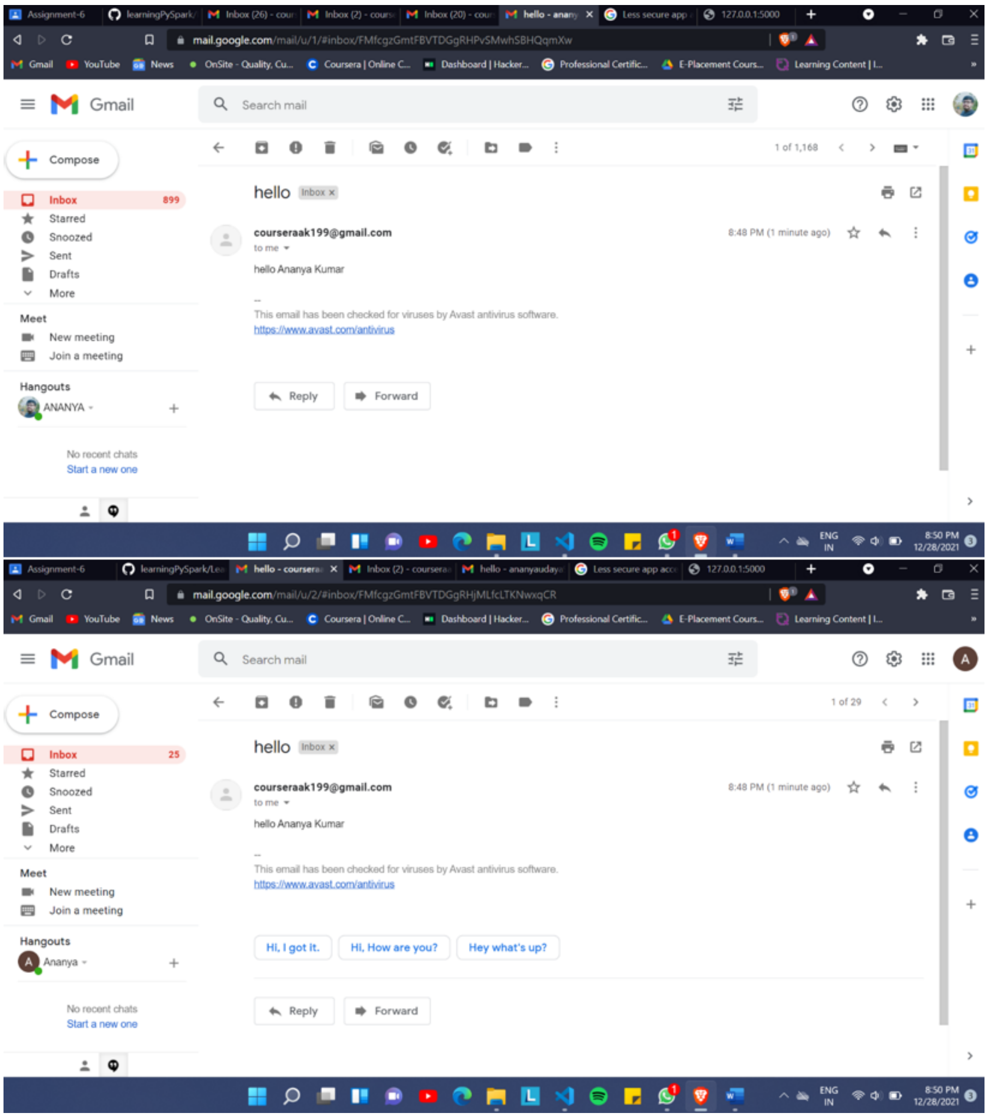
```
>use corona
>db.createCollection(Karnataka)
>db.Karnataka.insertMany([{ District:"Shimoga",
Total:29618,Active:283,Discharged:9656,Death:221},
{ District:"Belgaum",Total:35618,Active:283,Discharged:50400
,Death:243
{District:"Bengaluru Urban",Total:798321,Active:0
,Discharged:167122,Death:3402},{ District:"Dharwad",Total:
33488,Active:404,Discharged:12147,Death:448}, { District:"
Mysuru",Total:84078,Active:31,Discharged:21543,Death:611}])
>db.Karnataka.findOne()
>db.Karnataka.find({},{District:1,_id:0,Active:1}).pretty();
>db.Karnataka.find({},{District:1,Active:1,_id:0})
```

```
.sort({Active:−1}).limit(1)
>db.Karnataka.find({},{District:1,Active:1,_id:0})
.sort({Active:−1})
> db.Karnataka.remove({$expr:{$gt:["$Discharged","$Active"]}
  },1)
> db.Karnataka.remove({$expr:{$lt:["$Death",20]}})
Since all the documents got removed in last command, Create
new docume
> db.Karnataka.insertMany([{District:"Mandya",Active:12000
3,Discharged:50400,Death:10},{    District:"Bagalkot",Active:
20023,Discharged:100000,Death:20},
{ District:"Udupi",Active:220032,Discharged:150000,Death:30},
{ District:"Mysore",Active:20000,Discharged:100000,Death:20},
{ District:"Tumkur",Active:325400,Discharged:100302,Death:10
},
{ District:"Chikballapur",Active:420000,Discharged:105000,
Death:2},
{ District:"Bidar",Active:24003,Discharged:103400,Death:30}])
>db.Karnataka.updateOne({"District":"Udupi"},
{$set:{"Active":20000,"Death":10}})
>db.Karnataka.find({"District":"Udupi"}).pretty()
```

6) Bulk Emailing in Flask

```
from flask import *
from flask_mailimport *app = Flask(__name__)
app.config["MAIL_SERVER"]='smtp.gmail.com' app.config["
MAIL_PORT"] = 465
app.config["MAIL_USERNAME"] = 'courseraak199@gmail.com' app.c
onfig['MAIL_PASSWORD'] = 'coursera'
app.config['MAIL_USE_TLS'] = Falseapp.config['MAIL_USE_SSL']
= True
users = [{'name':'Ananya Kumar','email':'ananyaudaya99@gmail.
com'},
{'name':'Ananya Kumar','email':'courseraak2@gmail.com'},
{'name':'Ananya Kumar','email':'courseraak3@gmail.com'}]mail
= Mail(app)
@app.route("/")def index():
with mail.connect() as con:
for user in users:
message = "hello %s" %user['name']
```

```
msgs = Message(recipients=['email']], body = message, subject = 'hello',  sender = 'courseraak199@gmail.com')
```



hello Inbox

courseraak199@gmail.com 8:48 PM (1 minute ago)
to me

hello Ananya Kumar

--
This email has been checked for viruses by Avast antivirus software.
https://www.avast.com/antivirus

hello Inbox

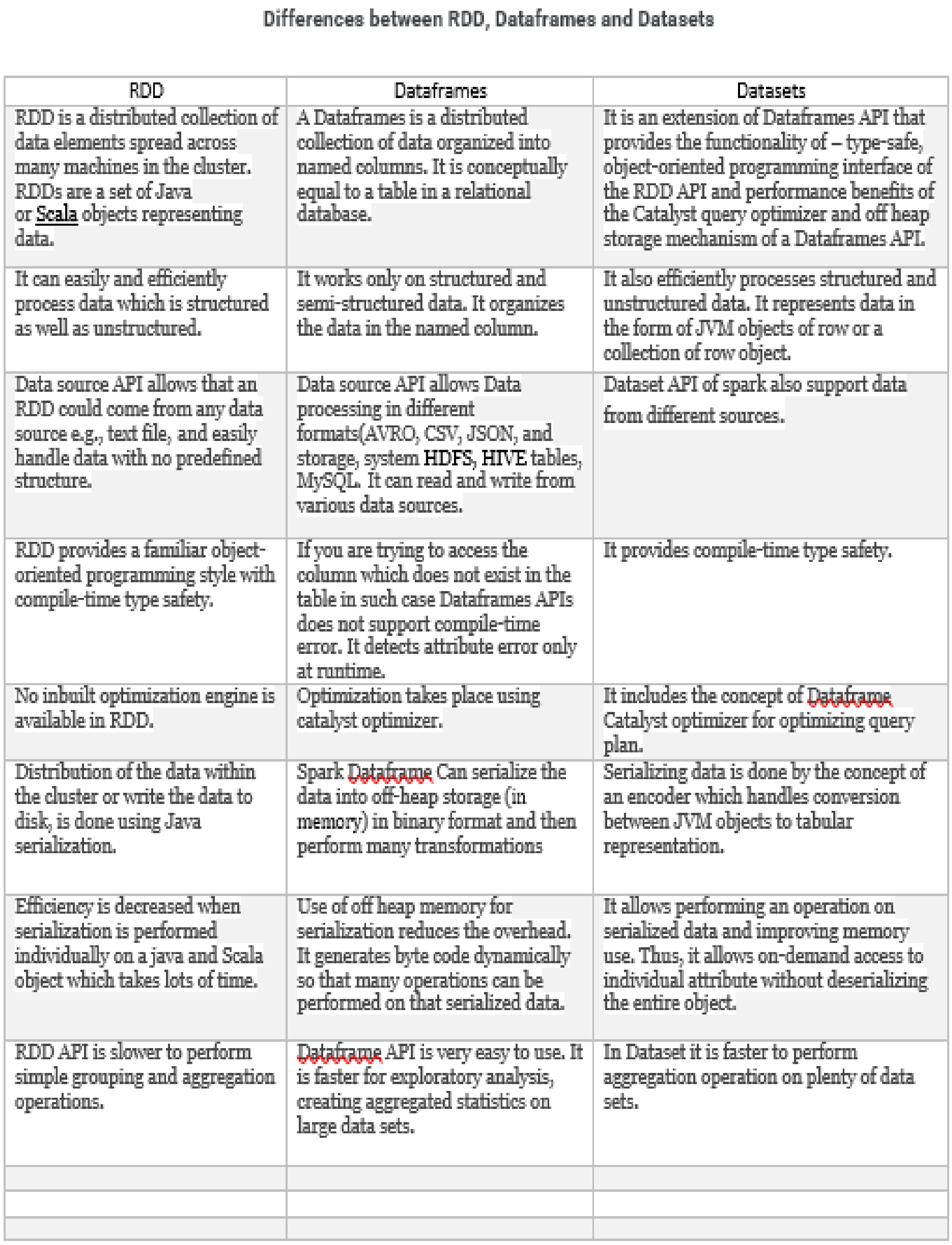
courseraak199@gmail.com 8:48 PM (1 minute ago)
to me

hello Ananya Kumar

--
This email has been checked for viruses by Avast antivirus software.
https://www.avast.com/antivirus

Figure 2.8: Snapshots of Assignment 5

7) Difference between RDD, Dataframes and Datasets



Differences between RDD, Dataframes and Datasets

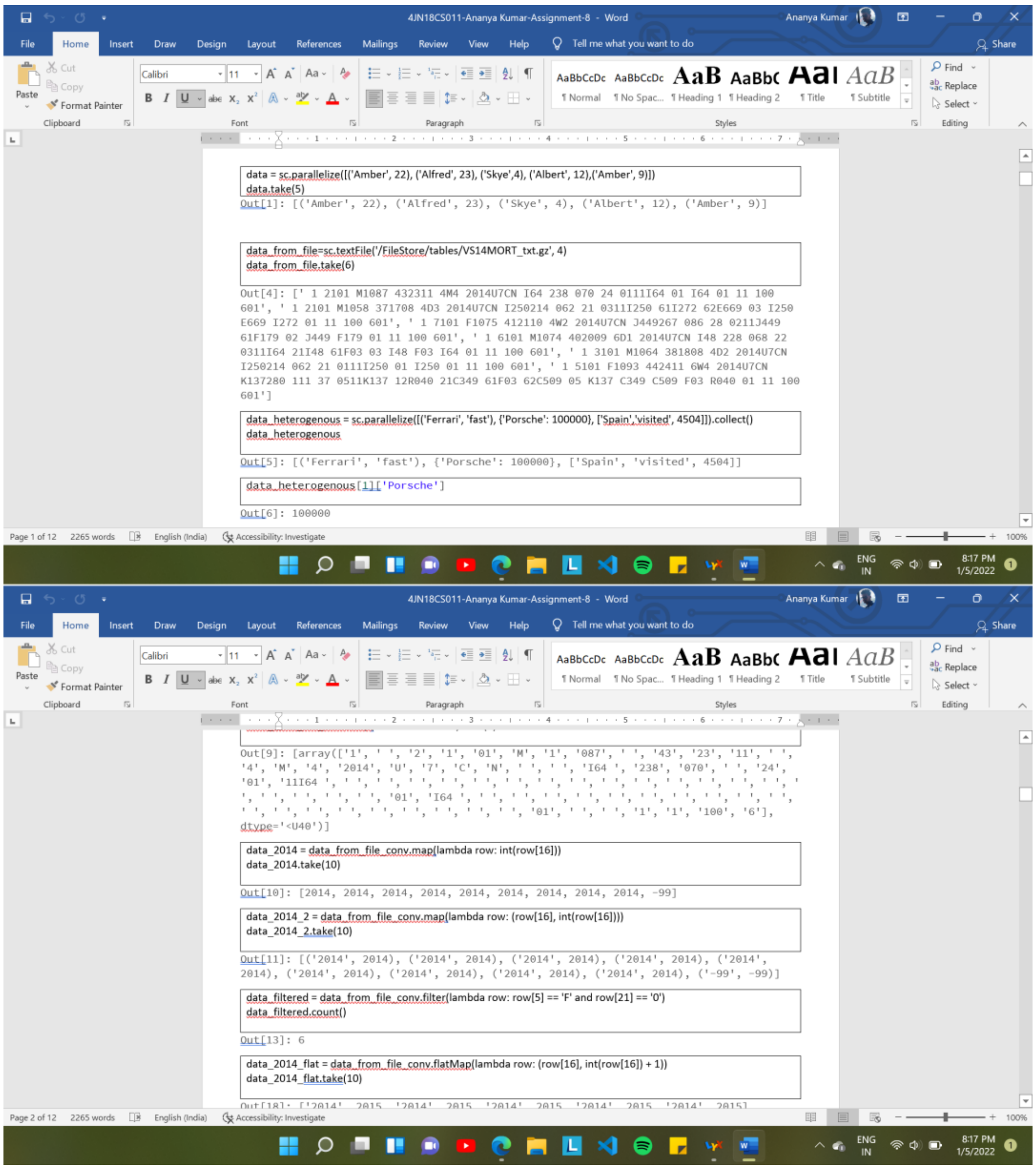
RDD	Dataframes	Datasets
RDD is a distributed collection of data elements spread across many machines in the cluster. RDDs are a set of Java or Scala objects representing data.	A Dataframes is a distributed collection of data organized into named columns. It is conceptually equal to a table in a relational database.	It is an extension of Dataframes API that provides the functionality of – type-safe, object-oriented programming interface of the RDD API and performance benefits of the Catalyst query optimizer and off heap storage mechanism of a Dataframes API.
It can easily and efficiently process data which is structured as well as unstructured.	It works only on structured and semi-structured data. It organizes the data in the named column.	It also efficiently processes structured and unstructured data. It represents data in the form of JVM objects of row or a collection of row object.
Data source API allows that an RDD could come from any data source e.g., text file, and easily handle data with no predefined structure.	Data source API allows Data processing in different formats(AVRO, CSV, JSON, and storage, system HDFS, HIVE tables, MySQL. It can read and write from various data sources.	Dataset API of spark also support data from different sources.
RDD provides a familiar object-oriented programming style with compile-time type safety.	If you are trying to access the column which does not exist in the table in such case Dataframes APIs does not support compile-time error. It detects attribute error only at runtime.	It provides compile-time type safety.
No inbuilt optimization engine is available in RDD.	Optimization takes place using catalyst optimizer.	It includes the concept of Dataframe Catalyst optimizer for optimizing query plan.
Distribution of the data within the cluster or write the data to disk, is done using Java serialization.	Spark Dataframe Can serialize the data into off-heap storage (in memory) in binary format and then perform many transformations	Serializing data is done by the concept of an encoder which handles conversion between JVM objects to tabular representation.
Efficiency is decreased when serialization is performed individually on a java and Scala object which takes lots of time.	Use of off heap memory for serialization reduces the overhead. It generates byte code dynamically so that many operations can be performed on that serialized data.	It allows performing an operation on serialized data and improving memory use. Thus, it allows on-demand access to individual attribute without deserializing the entire object.
RDD API is slower to perform simple grouping and aggregation operations.	Dataframe API is very easy to use. It is faster for exploratory analysis, creating aggregated statistics on large data sets.	In Dataset it is faster to perform aggregation operation on plenty of data sets.

Figure 2.9: Snapshot of Assignment 6

8) Working in DataBricks using practical examples

```
 import     urllib.request
 f= urllib.request.urlretrieve
 ("http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_1
0_percent.gz""kddcup.data_10_percent.gz")     data_file
=
 "/FileStore/tables/kddcup_data_10_percent.gz"

 raw_data = sc.textFile(data_file)    raw_data.count(raw_da
ta.take(5)

 a = range(100)
 data = sc.parallelize(a)data.count()

 data.take(5)

data_file = "/FileStore/tables/kddcup_data_10_percent.gz"
raw_data = sc.textFile(data_file)
 normal_raw_data = raw_data.filter(lambda x:        '
normal.'       in x)
 from time import time
 t0 = time()
 normal_count = normal_raw_data.count()tt = time() − t0
 print("There are {}'normal 'interactions".format(normal_coun
t))
 print(“Count completed in {} seconds".format(round(tt,3)))

from pprint        import    pprint
csv_data = raw_data.map(lambda x: x.split(","))t0 = time()
head_rows = csv_data.take(5)
tt = time() − t0     print    ("Parse    completed    in {}
seconds".format(round(tt,3)))    pprint(head_rows[0])

t0 = time()
head_rows = csv_data.take(100000)
tt = time() − t0     print    ("Parse    completed    in {} s
econds".format(round(tt,3)))
```

```
data_file = "/FileStore/tables/kddcup_data_10_percent.gz"raw
_data = sc.textFile(data_file)
key_csv_data = raw_data.map(parse_interaction)normal_key_in
teractions =
key_csv_data.filter(lambda x: x[0] == "normal.")t0 = time()
all_normal = normal_key_interactions.collect()tt = time() − t
0
normal_count = len(all_normal)

print ("Data collected in {} seconds".format(round(tt,3)))
print ("There are {} 'normal' interactions".format(normal_co
unt))

csv_data = raw_data.map(lambda x: x.split(","))
protocols = csv_data.map(lambda x: x[1]).distinct()protocols
.collect()

services = csv_data.map(lambda x: x[2]).distinct()services.c
ollect()

data = sc.parallelize([('Amber', 22), (
'Alfred', 23), ('Skye',4), ('Albert', 12),('Amber', 9)])data
.take(5)

product = protocols.cartesian(services).collect()print("
There are {} combinations          of
protocol X service".format(len(product)))
```



```
data = sc.parallelize([('Amber', 22), ('Alfred', 23), ('Skye',4), ('Albert', 12),('Amber', 9)])
data.take(5)
```

```
Out[1]: [('Amber', 22), ('Alfred', 23), ('Skye', 4), ('Albert', 12), ('Amber', 9)]
```

```
data_from_file=sc.textFile('/FileStore/tables/VS14MORT_txt.gz', 4)
data_from_file.take(6)
```

```
Out[4]: [' 1 2101 M1087 432311 4M4 2014U7CN I64 238 070 24 0111I64 01 I64 01 11 100
601', ' 1 2101 M1058 371708 4D3 2014U7CN I250214 062 21 0311I250 61I272 62E669 03 I250
E669 I272 01 11 100 601', ' 1 7101 F1075 412110 4W2 2014U7CN J449267 086 28 0211J449
61F179 02 J449 F179 01 11 100 601', ' 1 6101 M1074 402009 6D1 2014U7CN I48 228 068 22
0311I64 21I48 61F03 03 I48 F03 I64 01 11 100 601', ' 1 3101 M1064 381808 4D2 2014U7CN
I250214 062 21 0111I250 01 I250 01 11 100 601', ' 1 5101 F1093 442411 6W4 2014U7CN
K137280 111 37 0511K137 12R040 21C349 61F03 62C509 05 K137 C349 C509 F03 R040 01 11 100
601']
```

```
data_heterogenous = sc.parallelize([('Ferrari', 'fast'), {'Porsche': 100000}, ['Spain','visited', 4504]]).collect()
data_heterogenous
```

```
Out[5]: [('Ferrari', 'fast'), {'Porsche': 100000}, ['Spain', 'visited', 4504]]
```

```
data_heterogenous[1]['Porsche']
```

```
Out[6]: 100000
```

```
Out[9]: [array(['1', ' ', '2', '1', '01', 'M', '1', '087', ' ', '43', '23', '11', ' ',
'4', 'M', '4', '2014', 'U', '7', 'C', 'N', ' ', ' ', 'I64 ', '238', '070', ' ', '24',
'01', '11I64 ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '
', ' ', ' ', ' ', ' ', '01', 'I64 ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '01', ' ', ' ', '1', '1', '100', '6'],
dtype='<U40')]
```

```
data_2014 = data_from_file_conv.map(lambda row: int(row[16]))
data_2014.take(10)
```

```
Out[10]: [2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, -99]
```

```
data_2014_2 = data_from_file_conv.map(lambda row: (row[16], int(row[16])))
data_2014_2.take(10)
```

```
Out[11]: [('2014', 2014), ('2014', 2014), ('2014', 2014), ('2014', 2014), ('2014',
2014), ('2014', 2014), ('2014', 2014), ('2014', 2014), ('2014', 2014), ('-99', -99)]
```

```
data_filtered = data_from_file_conv.filter(lambda row: row[5] == 'F' and row[21] == '0')
data_filtered.count()
```

```
Out[13]: 6
```

```
data_2014_flat = data_from_file_conv.flatMap(lambda row: (row[16], int(row[16]) + 1))
data_2014_flat.take(10)
```

```
Out[18]: ['2014', 2015, '2014', 2015, '2014', 2015, '2014', 2015, '2014', 2015]
```

Figure 2.10: Snapshots of Assignment 7

2.3 Meetings Conducted

A meeting was conducted every week to track the progress of mini-project work assigned. The meeting details are as shown in Table 3.1.

Table 2.1: Meeting Details

Sl. No.	Date	Name with USN	Meeting contents	Members present	Outcome
1	02/09/2021	Ananya Kumar 4JN18CS011	Django	16	Full stack python with Django working with relational, MongoDB and REST API using Django
2	09/09/2021	Ananya Kumar 4JN18CS011	Flask	16	Micro web framework Flex with interfaces to open-source cloud databases
3	14/09/2021	Ananya Kumar 4JN18CS011	Cassandra	16	Interfaces to open-source cloud databases- Cassandra, MongoDB and Elastic search
4	23/09/2021	Ananya Kumar 4JN18CS011	Pyspark	12	Exploring Big data and Machine learning on big data using PySpark

2.4 Soft skills

The following soft skills were acquired during four weeks of internship.

- Time Management : Time management skills are those that help us use our time effectively and achieve desired results. Time management skills can help you allocate your time properly and accomplish tasks efficiently. Some of the most important skills related to successful time management skills include: Organization, Prioritization, Goal setting, Communication, Planning, Delegation, Stress management, Flexibility.

- Personality Development :Personality development refers to how the organized patterns of behavior that make up each person's unique personality emerge over time. Many factors go into influencing personality, including genetics, environment, parenting, and societal variables. Perhaps most importantly, it is the ongoing interaction of all of these influences that continue to shape personality over time.

- Communication: Effective Communication is significant for managers in the organizations so as to perform the basic functions of management, i.e., Planning, Organizing, Leading and Controlling. Communication helps managers to perform their jobs and responsibilities. Communication serves as a foundation for planning. All the essential information must be communicated to the managers who in-turn must communicate the plans so as to implement them.

- Email writing: Email communication is important type of written communication. Today, communications are conducted among business firms, organizations and companies mostly via emails. Moreover email provides most authentic and secure means of communication. The records of past emails can be retrieved very easily in moments.

- Group Discussion practice: A little practice on GD was done and we were taught how to start the GD conversation and how to end it effectively to clear the round.

- Teamwork: Teamwork is important because it enables your team to share ideas and responsibilities, which helps reduce stress on everyone, allowing them to be meticulous and thorough when completing tasks. This will enable them to meet sales goals quickly.

- Problem Solving skills: Problem-solving skills allow us to find candidates who are cognitively equipped to handle anything their jobs throw at them. Problem solvers can observe, judge, and act quickly when difficulties arise when they inevitably do.

- Work ethics: Employees who are ethically positive, honest, hardworking, and driven by principles of fairness and decency in the workplace, increases the overall morale and enhances the performance of an organization. A company that has established behavioral policies can improve its reputation and help ensure its long-term success.

- Adaptability Skills: Adaptability expands our capacity to handle change, no matter how serious it might be. Instead of throwing away our energy trying to change our circumstance, we will change our-self right from within, thus making us thrive in whatever situation we find yourself.

Summary

This chapter provides the tasks undertaken during the internship and also some information regarding the technologies used and the details of the assignments completed.

Chapter 3

Reflection Notes

This internship provided an environment to carry out a real time min-project and implement the requirements submitted during the initial meeting conducted on Project Work. The reflection on technical outcomes achieved and my learning experience in the training are explained in this chapter.

3.1 Technical Outcomes

- Python and Django Overview, Django Templates and Admin Interfaces
- Django working with Structured database like MySQL, Model forms, Uploading files, Cookies and Sessions, Class based and Generic Views in Django
- Djongo – REST API with MongoDB (NoSQL database)
- Python Flask Microweb Overview
- Flask Mail, Uploading files, Cookies and sessions
- Flask WTF with Atlas MongoDB
- Flask with DataStax Cassandra PySpark APIs for Big data analytics with Databricks
- Transformations and Actions in PySpark
- Dataframe API in PySpark
- Checking for duplicates, missing observations, and outliers, Graph Frames, Facebook circles – A Gentle Introduction to Apache Spark GraphFrames , Introducing MLlib,
- Documentation using Lyx

Using these skills a Mini project was done on Boat Service Booking Online using Django and MongoDB.At first create a base and then login, register,logout template:base.html:

```
<!doctype html>
<html lang="en">
<head>
<link rel="stylesheet"
 href="https://maxcdn.bootstrapcdn.com
```

```
/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384−
Gn5384xqQ1aoWXA+058 RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW /
dAiS6JXm " crossorigin="anonymous">
<title>BSB auth </title> </head>
<body>
<div class="container">
<div class="row justify−content−center">
<div class="col −4">
<h1 class="text−center"> Welcome to BSB! </h1>
{% block main %}
{% endblock %}
</div> </div> </div> </body> </html> login.html:
```

```
{% extends "registration\base.html" %}
{% load crispy_forms_tags %}
{% block main %}
<div class="card">
<div class="card −body">
<h4 class="card−title">Login to your account </h4>
<form method="POST">
{% csrf_token %}
<input type="hidden" name="next" value="{{ next }}">
{{ form|crispy }}
<buttontype="submit" class="btn btn−primary btn−block">Login
</button>
Don’t have an account?
<a href="/register" target="blank"><strong>register here </st
rong></a>
</form>
{% csrf_token %}
</table>
</div> </div>
{% endblock %} register.html:
```

```
{% extends "registration\base.html" %}
{% load crispy_forms_tags %}
{% block main %}
<div class="card">
```

```
<div class="card-body">
<h4 class="card-title">Register your account</h4>
<form method="POST">
<input type="hidden" name="next" value="{{ next }}">
{% csrf_token %}
{{ form|crispy }}
<buttontype="submit" class="btn btn-primary btn-block">Regi
ster</button>
<div> If you already have an account <a href="/login"target
="blank">
<strong>login</strong></a> instead.
</div> </form></div> </div> {% endblock %} logout.html:
```

```
{% extends 'registration/base.html' %}
{% block main %}
<p>You are logged out!</p>
<a href="{% url 'registration/login' %}">Log in again</a>
{% endblock %}
```

In forms.py create a form for login and register:

```
from django.contrib.auth.forms importUserCreationForm from d
jango.contrib.auth.models import User
class UserRegisterForm(UserCreationForm):
email = forms.EmailField(required=True)classMeta:
model = User
fields = ('username', 'email', 'password1', 'password2')def
    save(self, commit=True):
user = super(UserRegisterForm, self).save(commit=False)user.
email = self.cleaned_data['email']
if commit:
user.save()return user
```

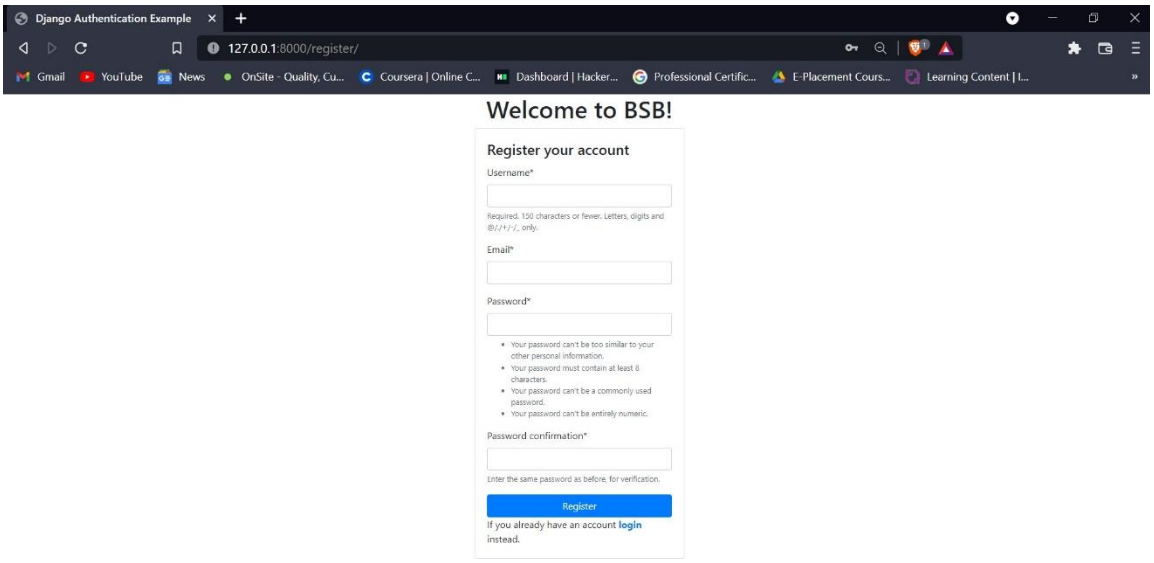
Then at views.py create a view function for login and register and logout using the built-in functions Django has,

```
from django.contrib.auth import authenticate,login, logout
from django.contrib.auth.forms importAuthenticationForm
from django.contrib import messages
from  .forms  import  UserRegisterForm

def  register(request):
if  request.method  ==  'POST':
form  =  UserRegisterForm(request.POST)if form.is_valid():
          user  =  form.save()
          username  =  form.cleaned_data.get('username')
          messages.success(request, f'Account  created for {
          username}!')login(request,user)
          return  redirect('/services/?submitted=True')
else:
          for  msg  in  form.error_messages:
          messages.error(request, f"{msg}:  {form.error_message
          s[msg]}")return           render(request = request,
          template_name  =  'registration/register.html,context
          ={"form":form})
          form  =  UserRegisterForm
return  render(request  =  request,
          template_name  =  'registration/register.html',context
          ={"form":form})


def  login_view(request):
if  request.method  ==  'POST':
form  =  AuthenticationForm(request=request, data=request.POST
)if form.is_valid():
username  = form.cleaned_data.get('username')password  =  form.c
leaned_data.get('password')
user  =  authenticate(username=username,  password=password)if u
seris     not  None:
          login(request,  user)
          messages.info(request, f"You are now logged   in  as  {
          username}")return        redirect('/services/')
else:
          messages.error(request,  "Invalid  username  or  password.")
else:
          messages.error(request,"Invalid username  or  password.")form  =
          AuthenticationForm()
```

```
        return render(request = request,
template_name = 'registration/login.html',context={"form":
form })

def logout_view(request):
logout(request)
messages.info(request, "Logged out successfully!")return r
edirect('/services/')
```

In urls.py, add these paths:

```
path('register/',register, name='register'),path('login/',
                login_view, name='Login'),
path('logout/', logout_view, name='logout'),
```



Django Authentication Example

127.0.0.1:8000/register/

Welcome to BSB!

Register your account

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

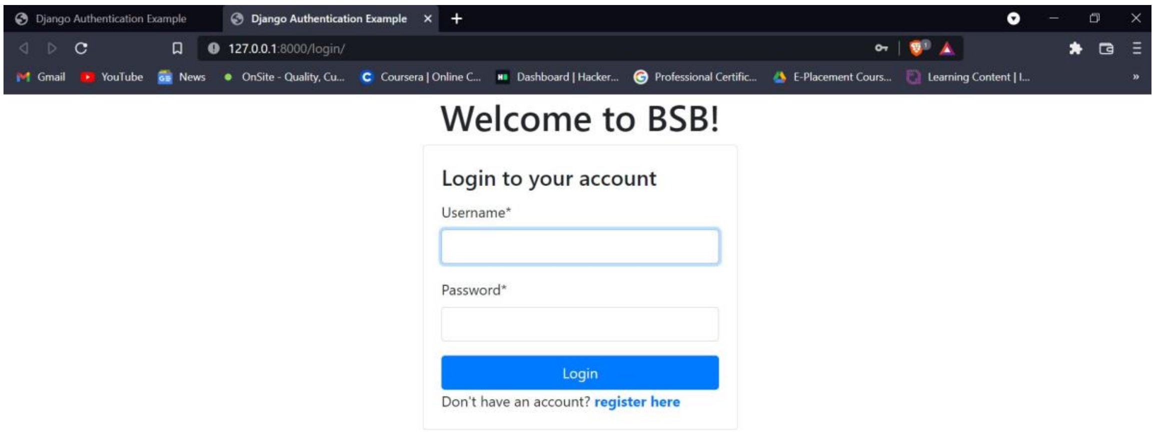
Password confirmation*

Enter the same password as before, for verification.

Register

If you already have an account login instead.

Figure 3.1: Snapshot of Register Page



Django Authentication Example

127.0.0.1:8000/login/

Welcome to BSB!

Login to your account

Username*

Password*

Login

Don't have an account? register here

Figure 3.2: Snapshot of Login Page

Then after some front end templates created for the details of the company, models were created for storing booking information and then register them in the admin page,

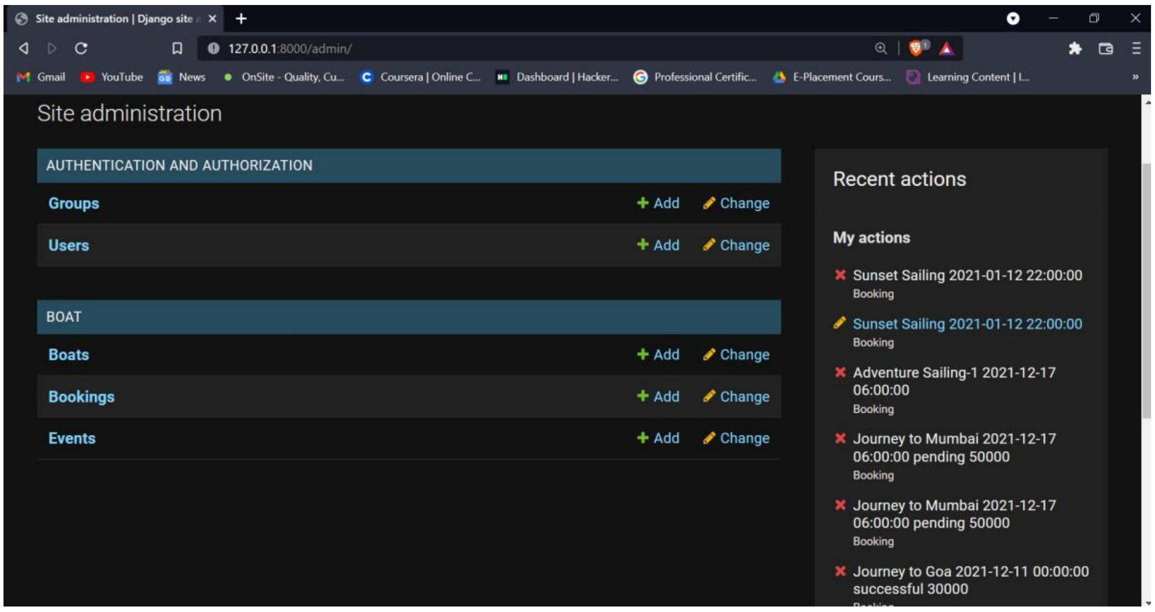
models.py:

```
from django.db import models from decimal import Decimal
from django.forms import widgetsclass

Boat (models . Model )
boat_name = models . CharField ( max_length =100 , primary_ key =True
, ) boat_type = models . CharField ( max_length =100)
boat_location = models . CharField ( max_length =100) boat_capacit
y = models . IntegerField ()
boat_price = models . IntegerField () def __str__( self ):
        return self . boat_name +" "+ self . boat_type


class Event ( models . Model ):
event_name=models . CharField ( max_length =50 , primary_ key =True , u
nique=False )
boat=models . ForeignKey ( Boat , on_delete=models .CASCADE, null
=True , default =None )
event_location =models . CharField ( max_length =30)
event_capacity =models . DecimalField ( max_digits =10000 , decimal
_places =0) event_price =models . DecimalField ( max_digits =
100000 , decimal_places =0)
def __str__( self ):
return self . boat . boat_name +"−size :"+
str( self . event_capacity )+" price :"+ str ( self . event_price )
```

```
class  Booking(models.Model):
name  =  models.CharField(max_length =200) email  =  models.EmailF
ield()
phone_no  =  models.CharField(max_length =10)
event  =  models.ForeignKey(Event ,on_delete=models.CASCADE)
booking_date  =  models.DateField(  auto_now=False ,
     auto_now_add=False) booking_time  =  models.TimeField (
auto_now=False ,
     auto_now_add= False ) def                          __str__( self
):
return self.event.event_name +"  "+ str(self.booking_date)+ "  "+
str(self.booking_time)
```

admin.py,

```
from  django.contrib  import  admin
from  boat.models  import  Boat ,   Event ,Booking #  Register your
models        here .
@admin.register(Boat)
class  BoatAdmin(admin.ModelAdmin):
          list_display  =  ('boat_name ','boat_type ')
          search_fields  =  ('boat_name ',   'boat_type ')

@admin.register(Event)
class  EventAdmin(admin.ModelAdmin):
          list_display  =  ('event_name ','event_price ')
          search_fields  =  ('event_name ',   'event_price ')

@admin.register(Booking)
class  BookingAdmin(admin.ModelAdmin):
          list_display  =  ('booking_date ','booking_time ')
          search_fields  =  ('booking_date ',      'booking_time ')
```



Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	Change
Users	+ Add	Change

BOAT

Boats	+ Add	Change
Bookings	+ Add	Change
Events	+ Add	Change

Recent actions

My actions

- Sunset Sailing 2021-01-12 22:00:00
  Booking
- Sunset Sailing 2021-01-12 22:00:00
  Booking
- Adventure Sailing-1 2021-12-17 06:00:00
  Booking
- Journey to Mumbai 2021-12-17 06:00:00 pending 50000
  Booking
- Journey to Mumbai 2021-12-17 06:00:00 pending 50000
  Booking
- Journey to Goa 2021-12-11 00:00:00 successful 30000

Figure 3.3: Snapshot of Models registered in the admin

A form is created using templates and a view function is created to store the user input in the database called Booking:

Booking.html:

```
{% extends "layout.html" %}
{% block title %}Booking
{% endblock %}
{% block content %}
{% load crispy_forms_tags %}
<h1>Book a boat </h1>
<div class="container" style="position: inherit;">
{% if user.is_authenticated %}
<form method="post">


{% csrf_token %}
<div class="form-group" style="padding: 2px 2px 2px 2px;">
<label>Name</label><br><strong>{{user.username}}</strong><br
>
<label>Email </label><br><strong>{{user.email}}</strong>
{{form | crispy}}
</div>
<button type="submit" class="btn btn-primary">Submit </button
>
```

```
</ form >
{% else %}
<div class="alert alert−danger">
<strong>You are not logged in!</strong> Please login to book
a boat.
{% endif %}
</ div > {% endblock %}
```

forms.py:

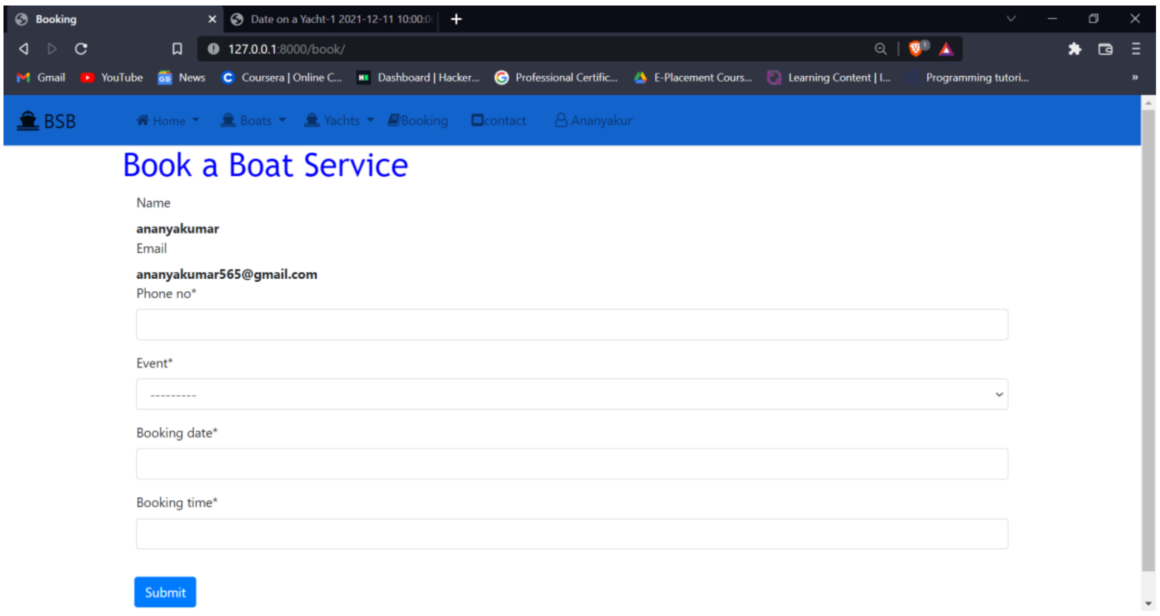
```
from django.forms import ModelForm from boat.models import
Booking
class BookingForm ( ModelForm ):
required_css_class = 'required'class    Meta:
model = Booking
fields = '__all__'
exclude = ('name', 'email')
widget={'booking_date': forms.DateInput(attrs={'class': 'date
picker'}),'booking_time': forms.TimeInput(attrs={'class': 'ti
mepicker'})}
```

views.py:

```
from boat.models import Booking def booking ( request ):
if request.method == 'POST':
          form = BookingForm(request.POST) if    form.is_valid()
            :
                    book=form.save(commit=False)
                    book.name = request.user.username book.email
                    = request.user.email
                    book.save()
                    return redirect('/order/?submitted=True')
            else:


          else:

            return        render(request, ' booking.html ',{'
        form':   form})
             form = BookingForm()
             return render(request, 'booking.html', {'
             form: ' form})
```

In urls.py, add this path:

```
path('book/', booking, name='booking')
```



Book a Boat Service

Name

ananyakumar

Email

ananyakumar565@gmail.com

Phone no*

Event*

Booking date*

Booking time*

Figure 3.4: Snapshot of Booking form

Now the booked details can be displayed in the booked list template page, which contains delete and update option along with the details:

order_listing.html:

```
{% extends "layout.html" %}
{% load static %}
{% block title %}Your Orders
{% endblock %}
{% block content %}
<div class="container">
{% if order_list is not empty %}        ,
<div class="row">
<div style="padding-left:350px;">
<h1>Your Orders </h1>
 </div>
 </div>
{% for order in order_list %}
<div class="col-13">
```

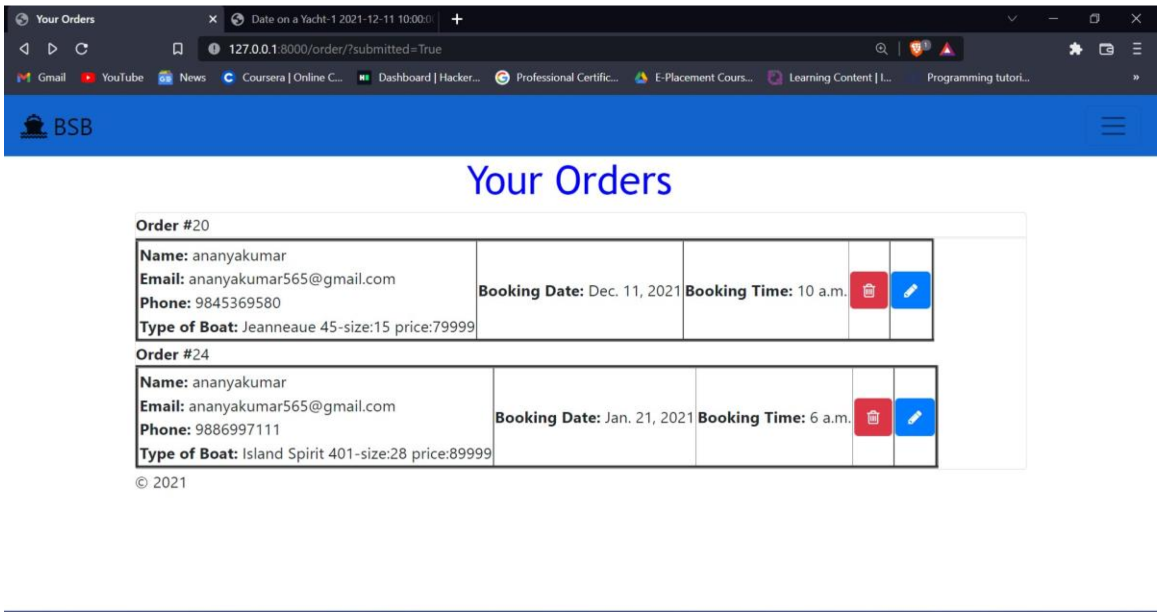
```
<div class="card">
<div class="card-content">
<table border="3" >
<thead >
<tr id="result_tr" style="display: none;"><span class="card-
title">
<strong>Order #</strong>{{order.id}} </span></tr>
<tr>
<td>
<strong>Name:</strong> {{order.name}}<br>
<strong>Email:</strong> {{order.email}}<br>
<strong>Phone:</strong> {{order.phone_no}}<br>
<strong>Type of Boat:</strong> {{order.event}}<br></td>
<td><strong>Booking Date:</strong> {{order.booking_date}}<br
></td>
<td><strong>Booking Time:</strong> {{order.booking_time}} </td
>
<!--give a button for delete-->
<td>
<a href="/delete/{{order.id}}"><button type="button" class="b
tn btn-danger">
<i class="fa fa-trash-o"></i></button></a></td>
<!--give a button for update-->
<td>
<a href="/update/{{order.id}}"><button type="button" class="b
tn btn-primary">
<i class="fa fa-pencil"></i></button></a></td>
</tr>
</thead>
{% endfor %}
</div>
</table>
</div>
</div>
{% else %}
<div class="alert alert-danger">
<strong>You have not booked</strong> please book a boat!</
div>
</div>
{% endif %} {% endblock %}
```

views.py:

```
def orderlisting(request):

#code to send authhenticated user's
#order list to the order listing page if request.user.is_auth
enticated:
order_list = Booking.objects.filter(name=request.user.
username)
return render(request,'order_listing.html',{'order_list': or
der_list})else:
return redirect('/login/')

def delete_order(id):
order = Booking.objects.get(id=id)
order.delete()
return redirect('/order/')

def update_order(request,id):
order = Booking.objects.get(id=id)
form = BookingForm(request.POST or None, instance=order)if
    form.is_valid():
form.save()
return redirect('/order/')
return render(request,'booking.html',{'form': form})
```

In urls.py, add these paths:

```
path('order/', orderlisting, name='order')
path('delete/<int:id>', delete_order, name='delete'),path('u
pdate/<int:id>',          update_order, name='update'),
```



Your Orders

Order #20

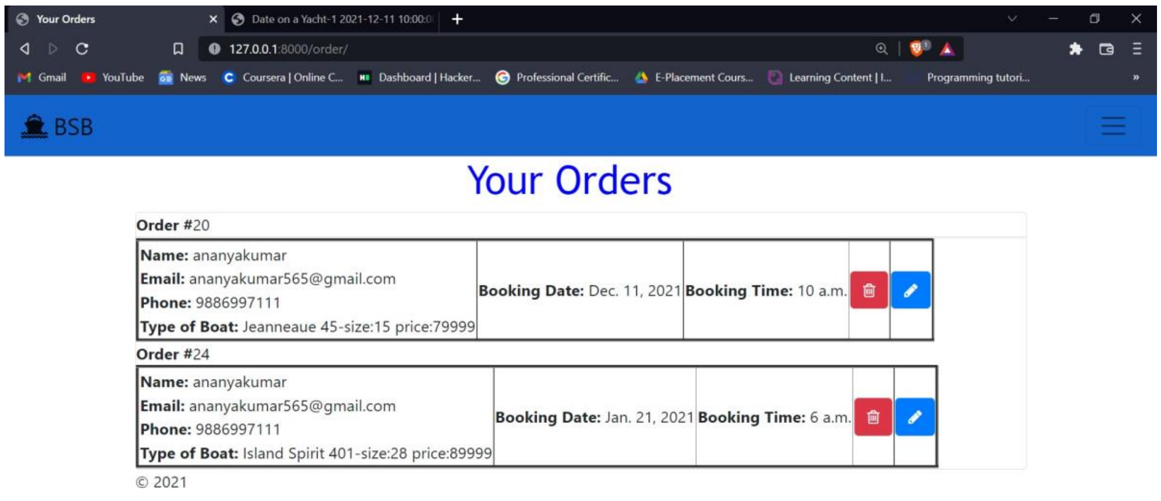
Name: ananyakumar **Email:** ananyakumar565@gmail.com **Phone:** 9845369580 **Type of Boat:** Jeanneaue 45-size:15 price:79999	**Booking Date:** Dec. 11, 2021	**Booking Time:** 10 a.m.		

Order #24

Name: ananyakumar **Email:** ananyakumar565@gmail.com **Phone:** 9886997111 **Type of Boat:** Island Spirit 401-size:28 price:89999	**Booking Date:** Jan. 21, 2021	**Booking Time:** 6 a.m.		

© 2021

Figure 3.5: Snapshot of Booked List



Your Orders

Order #20

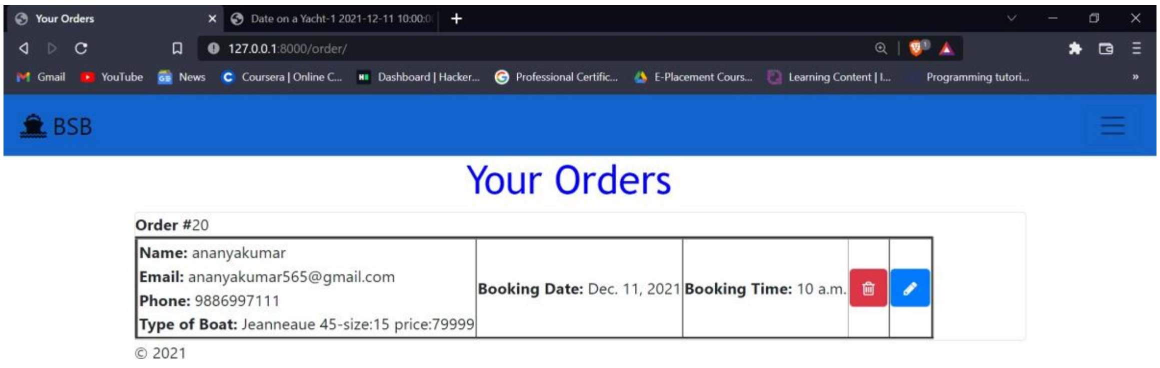
Name: ananyakumar **Email:** ananyakumar565@gmail.com **Phone:** 9886997111 **Type of Boat:** Jeanneaue 45-size:15 price:79999	**Booking Date:** Dec. 11, 2021	**Booking Time:** 10 a.m.		

Order #24

Name: ananyakumar **Email:** ananyakumar565@gmail.com **Phone:** 9886997111 **Type of Boat:** Island Spirit 401-size:28 price:89999	**Booking Date:** Jan. 21, 2021	**Booking Time:** 6 a.m.		

© 2021

Figure 3.6: Snapshot of Booked List after update



Your Orders

Order #20

Name: ananyakumar **Email:** ananyakumar565@gmail.com **Phone:** 9886997111 **Type of Boat:** Jeanneaue 45-size:15 price:79999	**Booking Date:** Dec. 11, 2021	**Booking Time:** 10 a.m.		

© 2021

Figure 3.7: Snapshot of Booked List after delete

The user can click those buttons if the user wants to update the booking or delete it, overall this is the booking mechanism of this project, where the user has to first register and login before booking a boat service and after the booking is done the user can display the booking details for confirmation.

3.2 Experiences

My internship in Ekathva Innovations has been the most rewarding and motivational experiences I have had during my time as a student at Jawaharlal Nehru New college of Engineering. Because of the lessons I learned from my supervisor and cooperating mentors, I am confident that I will continue to grow and develop professionally and in my personal endeavors. I leant about many trending topics of which industries are using in a very efficient way. I got a chance to learn about soft skills and interview cracking techniques which is gonna help me in my future placements and to work individually on project.

Summary

This chapter provides the detail of the reflected notes derived from this internship and provides the results of the mini project tasked in this internship and the experience of the internship is briefed.

Chapter 4

Conclusion

This internship, I have been provided an opportunity to work on a real-time project and also had the opportunity to explore technologies like Django, Flask, Python, MongoDB, PySpark. The project developed has insertion, deletion, update, view operations demonstrated using a Django framework and MongoDB. The project demonstrates the authentication and login for the user where without it they can’t book a boat service. This internship provided an opportunity to gain valuable experience in my upcoming career field. Having additional work experience before applying for job gives an edge over other candidates in a competitive job market.

Bibliography

[1] Antonio Mele, Django 2 by Example: Build powerful and reliable Python web applications from scratch, Packt Publishing, 2nd Edition, 2018

[2] Daniel Rubio, Beginning Django: Web Application Development and Deployment with Python, APress, 1stEdition, 2017

[3] https://djangobook.com/

[4] https://developer.wordpress.org/rest-api/reference/

[5] Jeff Carpenter, Eben Hewitt, Cassandra – The Definitive Guide, 3e: Distributed Data at Web Scale, O'Reilly Media, Inc., 3rd Edition, 2020

[6] https://www.analyticsvidhya.com/blog/2020/08/a-beginners-guide-to-cap-theorem-for-data-engineering/

[7] Plugge, Eelco Hawkins, Tim Membrey, Peter, The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing, APress, 1stEdition, 2017

[8] https://api.mongodb.com/python/current/tutorial.html

[9] Grinberg and Miguel, Flask Web Development: Developing Web Applications with Python, O'Reilly Media, Inc., 1stEdition, 2014

[10] https://realpython.com/tutorials/flask/

[11] Tomasz Drabas, Denny Lee,Learning Pyspark: Build data-intensive applications locally and deploy at scale using the combined powers of Python and Spark 2.0, Ingram Publishers, 1st Edition, 2017

[12] https://towardsdatascience.com/building-an-ml-application-with-mllib-in-pyspark-part-1-ac13f01606e2

[13] https://www.kaggle.com/datasets