

Abstract

Sports Equipments Management System is developed using FLASK which is widely used micro web framework in Python. Sports Equipments Management System is useful in performing all typically needed database operations for the management of sports equipments. For storing the data I have used MongoDB Atlas which is a cloud-based Database System. Sports Equipments Management System user can enter data to create their account through ID, select the equipment, quantity needed , data of taken , data of return and cost of rent. The Admin can monitor record correctly and even can generate softcopy of record.

Acknowledgements

I would like to acknowledge my profound gratitude to all those who have helped in implementing this Internship and project work. I am grateful to CIE, Ekathva Innovations and my institution Jawaharlal Nehru New College of Engineering for imparting us the knowledge with which we can do our best.

I would like to thank our beloved guide Dr. Chetan K R, Associate Professor, Mr. Akshay Kulkarni, CCO, Ekatva, Dr. Poornima K M, HOD dept of CSE and Mr. K Nagendra Prasad, Principal JNNCE who have helped us a lot in making this project and for their continuous encouragement and guidance throughout the Internship and project work.

Thanking you all,

Project Intern,

MADHUSUDHANA B M 4JN18CS043.

Contents

Abstract	i
Acknowledgements	ii
List of Figures	v
List of Tables	vi
1 Company Profile	1
1.1 Center for Innovation and Entrepreneurship (CIE)	1
1.2 Ekathva Innovations Pvt. Ltd.	1
2 About the Company	2
2.1 Initiatives of CIE	2
2.2 Additional Initiatives of CIE	3
2.3 Startups under CIE	5
2.4 Members of CIE	5
2.5 Members of Ekathva Innovations	6
3 Tasks Preformed	7
3.1 Technical Skills	7
3.1.1 Django framework	7
3.1.1.1 Overview	7
3.1.1.2 Django MVT	8
3.1.1.3 Django Views	9
3.1.1.4 Django Templates	9
3.1.1.5 Django Models	10
3.1.1.6 Django Forms	10
3.1.2 MongoDB	11
3.1.2.1 Need of MongoDB	11
3.1.2.2 Mongo Compass and Mongo Atlas	11
3.1.2.3 Djongo and REST API	12
3.1.3 Flask microframework	12
3.1.3.1 Use of Flask	12
3.1.3.2 Flask Environment	13
3.1.3.3 Dynamic Web page	13
3.1.3.4 Jinja2 Templates and static	14
3.1.4 Cassandra	15

3.1.4.1	CQL	15
3.1.4.2	DataStax driver	16
3.1.5	PySpark for Big Data Analytics	16
3.1.5.1	RDD	17
3.1.5.2	DataFrames	17
3.1.5.3	MLLib	17
3.1.5.4	Graph Frames	18
3.2	Assignments	18
3.3	Meetings Conducted	31
3.4	Soft skills	31
4	Reflection Notes	33
4.1	Technical Outcomes	33
4.2	Experiences	40
5	Conclusion	42
	Bibliography	43

List of Figures

3.1	Django MVT Architecturee	9
3.2	Rest APIs	12
3.3	Cassandra Architecture	15
3.4	Snapshots of Assignment 1	19
3.5	Snapshot of Assignmet 2	21
3.6	Snapshot of Assignment 3	22
3.7	Snapshot of Assignment 4	24
3.8	Snapshots of Assignment 5	25
3.9	Snapshots of Assignment 6	27
3.10	Snapshot of Assignment 7	27
3.11	Snapshots of Assignment 8	30
4.1	Snapshot of MongoDB Atlas Connection	34
4.2	Snapshot of Home Page	34
4.3	Snapshot of Data Entry Page	35
4.4	Snapshot of Find Page	36
4.5	Snapshot of Delete Page	37
4.6	Snapshot of Update Page	38
4.7	Snapshot of Record Page	39
4.8	Snapshot of Overall Data Page	40
4.9	Snapshot of Downloaded CSV File	40

List of Tables

2.1 CIE Project Funding	3
3.1 Meeting Details	31

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.

Chapter 2

About the Company

This chapter details on the products, services and organizational details of CIE and Ekathva Innovations Pvt. Ltd. The milestones and departmental subdivisions are also explored.

2.1 Initiatives of CIE

NewGen IEDC: National Science and Technology Entrepreneurship Development Board (NSTEDB), Department of Science & 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 upto 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 upto 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 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

2.2 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,

Table 2.1: 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

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.

2.3 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.

2.4 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

2.5 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 Customised Software Development are provided.

Chapter 3

Tasks Preformed

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 learnt. 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 facilitate 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.

3.1 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 focussed on Flask micro web framework and development of applications using it

3.1.1 Django framework

3.1.1.1 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 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 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.

3.1.1.2 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.

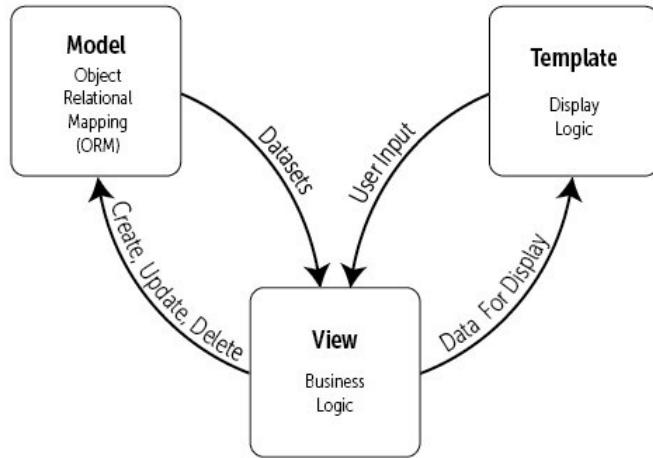


Figure 3.1: Django MVT Architecturee

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.

3.1.1.3 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_list })
  
```

3.1.1.4 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.votes|pluralize }}</li>
{% endfor %}
</ul>
</html>
```

3.1.1.5 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.IntegerField()
```

3.1.1.6 Django Forms

It automatically generates form widgets, Input validation, Redisplay 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>{% endif %}
<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>
```

3.1.2 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.

3.1.2.1 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.

3.1.2.2 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".

3.1.2.3 Django 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:

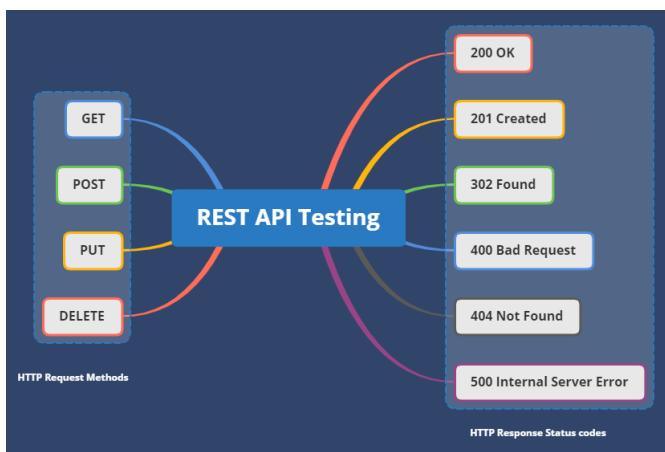


Figure 3.2: Rest APIs

3.1.3 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.

3.1.3.1 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.

3.1.3.2 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.

3.1.3.3 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>')
def user(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'))
```

3.1.3.4 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; }
```

3.1.4 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.

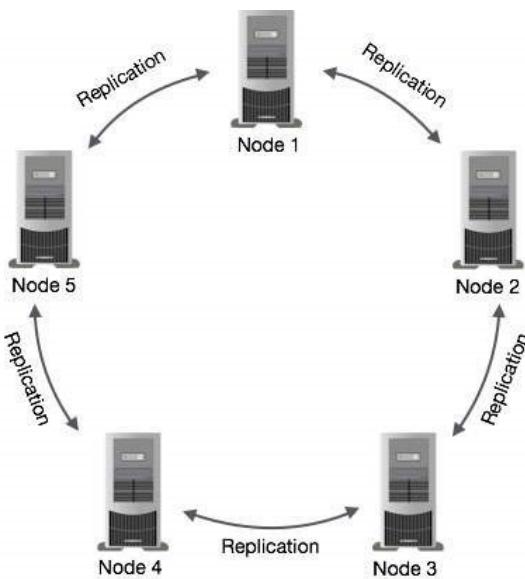


Figure 3.3: Cassandra Architecture

3.1.4.1 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. Clients 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

3.1.4.2 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.

3.1.5 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.

3.1.5.1 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.

3.1.5.2 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.

3.1.5.3 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, chisquare testing, linear algebra (sparse and dense matrices and vectors), and model evaluation methods

3.1.5.4 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'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.

3.2 Assignments

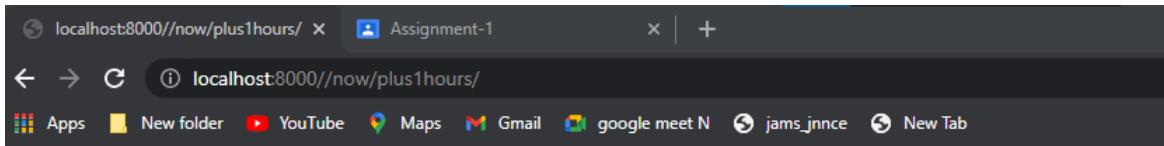
List of assignments:

- 1) Displaying Date and Time after all modification

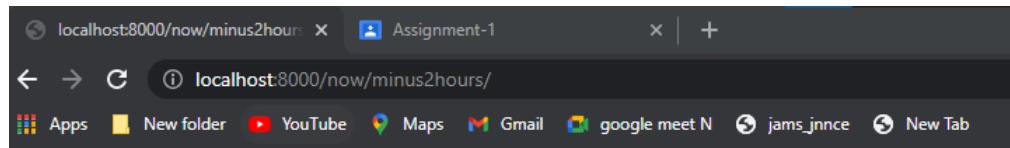
```
from django.shortcuts import render
from django.http import HttpResponse
import datetime
# Create your views here.

def home(request):
    return HttpResponse("Hello , Django !")
def greet(request):
    dt=datetime.datetime.now()
    return HttpResponse
("<h1>Current date and time is:"+str(dt)+"</h1>")
def hour_offset(request , plus_or_minus , offset):
    offset = int(offset)
    if offset == 1:
        hours = 'hour'
    else:
        hours = 'hours'
    if plus_or_minus == 'plus':
```

```
dt= datetime.datetime.now() + datetime.timedelta(hours=offset)
output= 'In %s %s , it will be %s.' %(offset , hours , dt)
else :
    dt= datetime.datetime.now() - datetime.timedelta(hours=offset)
    output = '%s %s ago , it was %s.' % (offset , hours , dt)
    output = '<html><body>%s</body></html>' % output
return HttpResponse(output)
```



In 1 hour(s), it will be 2021-09-05 21:08:44.007005.



2 hour(s) ago, it was 2021-09-05 17:55:57.596712.

Figure 3.4: 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(blank=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 = ('first_name', 'email')
    fieldsets = (( 'Required Information',
        {'description': "These fields are required for each event.",
        'fields': ('first_name', 'last_name'), 'salutation',
        'headshot')}),
        ('Optional Information', {
        'classes': ('collapse',), 'fields': ('email',)}),
    admin.site.register(Publisher)
```

Question 1

Musician	Genre
Django Reinhardt	jazz
Jimi Hendrix	rock
Louis Armstrong	jazz
Pete Townsend	rock
Yanni	new age
Ella Fitzgerald	jazz
Wesley Willis	casio
John Lennon	rock
Bono	rock
Garth Brooks	country
Duke Ellington	jazz
William Shatner	spoken word
Madonna	pop

Question 2

Musician	Genre
Django Reinhardt	jazz
Jimi Hendrix	rock
Louis Armstrong	jazz
Pete Townsend	rock
Yanni	new age
Ella Fitzgerald	jazz
Wesley Willis	casio
John Lennon	rock
Bono	rock
Garth Brooks	country
Duke Ellington	jazz
William Shatner	spoken word
Madonna	pop

Figure 3.5: Snapshot of Assignment 2

3) Displaying Customer orders list and Book Information

```
from django.shortcuts import render
from django.http import HttpResponseRedirect
from django.template import Template, Context
import datetime
from django.views import generic
from hello.models import Book, Author, Orders
from django.http 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'
```

The screenshot shows a web browser window with the title 'Orders' and the URL 'localhost:8000/orders/'. The page features a navigation bar with links for Home, ABOUT, SEARCH BOOK, ORDER, ORDERS, CONTACT, and LOGOUT. Below the navigation bar is a section titled 'Orders Info' containing a table with four rows of data. The table columns are 'Customer Name', 'Book', 'Nos Of Copies', and 'Address'. The data is as follows:

Customer Name	Book	Nos Of Copies	Address
Navaneeth	Data Structures	4	Thirthahalli
Varun Kumar	Django	2	Banglore
Mahi	Design and Analysis of Algorithms	1	Shimoga

The screenshot shows a web browser window with the title 'Book Info' and the URL 'localhost:8000/orders/book/3'. The page features a navigation bar with links for Home, ABOUT, SEARCH BOOK, ORDER, ORDERS, CONTACT, and LOGOUT. Below the navigation bar, the page displays the following information for a book:

Book Title: Design and Analysis of Algorithms
Publisher: Apress
Authors : Coremen

Figure 3.6: Snapshot of Assignment 3

4) Displaying View count for All Pages

```
from django.shortcuts import render, redirect
from django.http import HttpResponseRedirect
from django.template import Template, Context
import datetime
from django.views import generic
from hello.models import Book, Author, Orders
from django.http import HttpResponseRedirect
from hello.contact import ContactForm
from django.core.files.storage import FileSystemStorage
```

```
from hello.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(request):
    av = request.session.get('av', 0)
    request.session['av'] = av + 1
    return render(request, "about.html",
                  {'nv': request.session['av'] })
def contact(request):
    cv = request.session.get('cv', 0)
    request.session['cv'] = cv + 1
    return render(request, "contact.html",
                  {'nv': request.session['cv'] })
def visitor(request):
    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_cookie('lv', value)
    return html
def simple_upload(request):
    if request.method == 'POST' and request.FILES['myfile']:
        myfile = request.FILES['myfile']
        fs = FileSystemStorage()
        filename = fs.save(myfile.name, myfile)
        uploaded_file_url = fs.url(filename)
        return render(request, 'supl.html',
                      {
                          'uploaded_file_url': uploaded_file_url
                      })
    return render(request, 'supl.html')
def model_form_upload(request):
```

```
if request.method == 'POST':
form = DocumentForm(request.POST, request.FILES)
if form.is_valid():
form.save()
return redirect('/up')
else:
form = DocumentForm()
return render(request, 'model_form_upload.html', {
'form': form
})
```

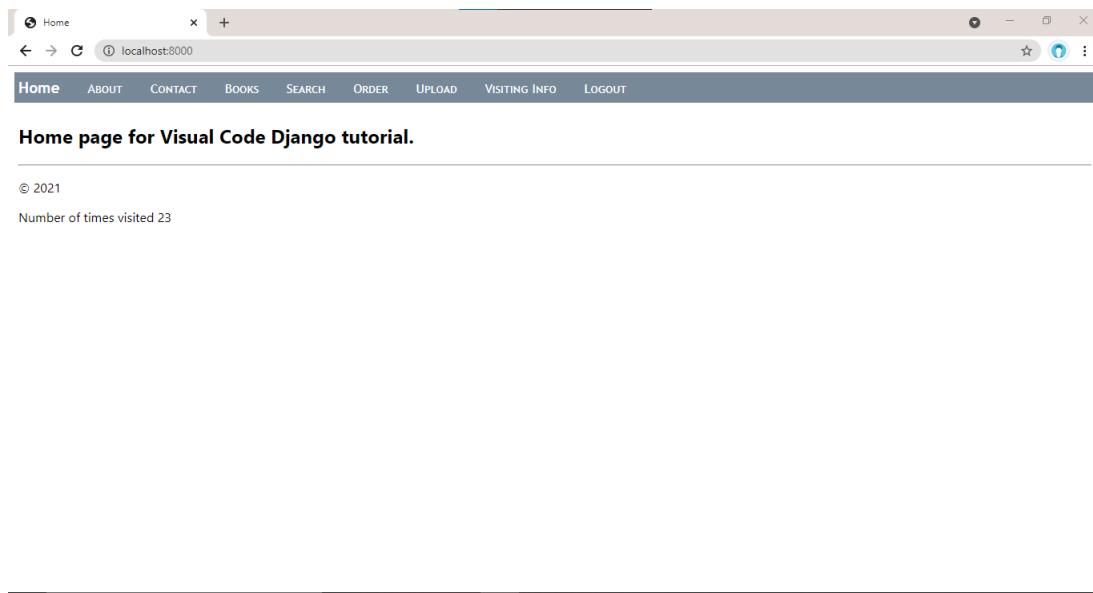


Figure 3.7: Snapshot of Assignment 4

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 documents:
> db . Karnataka . insertMany ([{ District :"Mandya" ,Active :120003 ,
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 ()
```

```
... db.Karnataka.find({{"District_Name":1,"Active_Cases":1,_id:0})
... { "District_Name" : "Mandya" , "Active_Cases" : 120003 }
... { "District_Name" : "Bagalkot" , "Active_Cases" : 20023 }
... { "District_Name" : "Udupi" , "Active_Cases" : 220032 }
... { "District_Name" : "Mysore" , "Active_Cases" : 20000 }
... { "District_Name" : "Tumkur" , "Active_Cases" : 325400 }
... { "District_Name" : "Chikballapur" , "Active_Cases" : 420000 }
... { "District_Name" : "Bidar" , "Active_Cases" : 24003 }
...
...
```

Figure 3.8: Snapshots of Assignment 5

6) Bulk Emailing in Flask

```
from flask import *
from flask_mail import *
import datetime
app = Flask(__name__)
```

```
app.config[ 'MAIL_SERVER' ]= 'smtp.gmail.com'
app.config[ 'MAIL_PORT' ]=465
app.config[ 'MAIL_USERNAME' ] = 'mbm19022000@gmail.com'
app.config[ 'MAIL_PASSWORD' ] = '*****'
app.config[ 'MAIL_USE_TLS' ] = False app.config[ 'MAIL_USE_SSL' ] = True

users = [{ 'name' : 'User 1' , 'email' : 'madhusudhanabm7@gmail.com' } ,
{ 'name' : 'User 2' , 'email' : 'madhubm007@gmail.com' } ,
{ 'name' : 'User 3' , 'email' : 'madhusudhanabm07@gmail.com' }]

mail = Mail(app)
@app.route("/")
def index():
    with mail.connect() as con:
        for user in users:
            message =
                "Hello %s now while sending this mail its currently %s
                Thank You."
                %(user[ 'name' ] , datetime.datetime.now())
            msgs =
            Message( recipients=
            [user[ 'email' ]] , body = message , subject = 'Date and Time' ,
            sender = 'mbm19022000@gmail.com')
            con.send(msgs)
    return "Sent"

if __name__ == "__main__":
    app.run(debug = True)
```

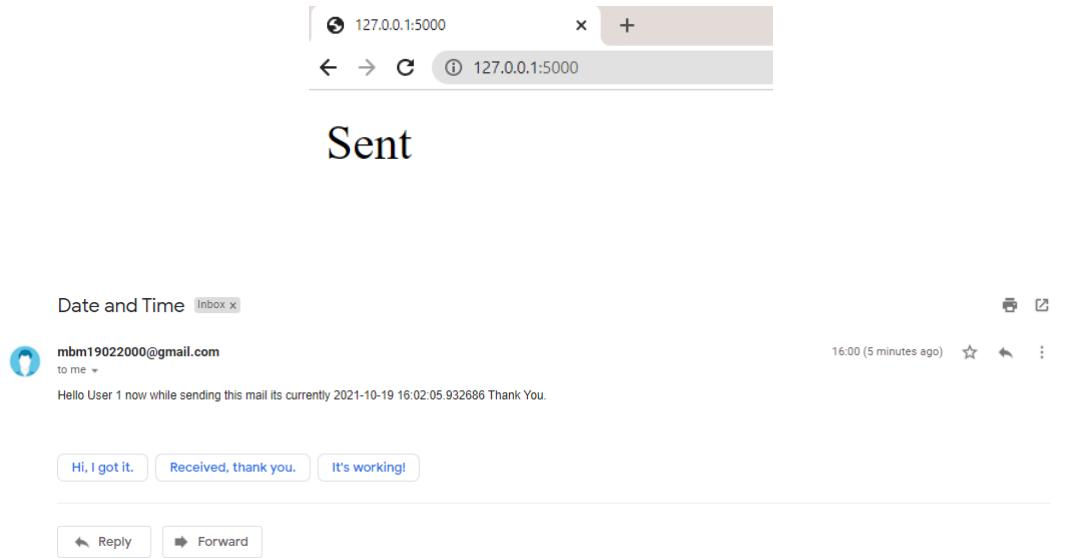


Figure 3.9: Snapshots of Assignment 6

7) Difference between RDD, Dataframes and Datasets

Criteria	RDD	DataFrames	DataSets
Data Representation	Distributed collection of data elements spread across many machines in the cluster.	Distributed collection of data organized into named columns conceptually equal to a table in a relational database.	It is an extension of DataFrame 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 DataFrame API.
Data Formats	It can easily and efficiently process data which is structured as well as unstructured. But like Dataframe and DataSets, RDD does not infer the schema of the ingested data and requires the user to specify it.	It works only on structured and semi-structured data. It organizes the data in the named column. DataFrames allow the Spark to manage schema.	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. Which is represented in tabular forms through encoders.
Data Sources API	Data source API allows that an RDD could come from any data source and easily handle data with no predefined structure.	Data source API allows Data processing in different formats. It can read and write from various data sources like AVRO, CSV, JSON etc.	Dataset API of spark also support data from different sources.
Compile-time type safety	Provides a familiar object-oriented programming style with compile-time type safety.	While access the column which does not exist in the table in such case Dataframe APIs does not support compile-time error. It detects attribute error only at runtime.	It provides compile-time type safety.
Optimization	No inbuilt optimization engine is available in RDD. When working with structured data, RDDs cannot take advantages of sparks advance optimizers. Developers optimise each RDD on the basis of its attributes.	Optimization takes place using catalyst optimizer. Dataframes use catalyst tree transformation framework in four phases: a) Analyzing a logical plan to resolve references. b) Logical plan optimization. c) Physical planning. d) Code generation to compile parts of the query to Java bytecode.	It includes the concept of Dataframe Catalyst optimizer for optimizing query plan.

Figure 3.10: Snapshot of Assignment 7

8) Working in DataBricks using practical examples

```
import urllib.request
f= urllib.request.urlretrieve
("http://kdd.ics.uci.edu/databases/kddcup99/
kddcup.data_10_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_data.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_count))
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 {}
seconds".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_interactions =
key_csv_data.filter(lambda x: x[0] == "normal.")
t0 = time()
all_normal = normal_key_interactions.collect()
tt = time() - t0
normal_count = len(all_normal)
print("Data collected in {} seconds".format(round(tt,3)))
print("There are {} 'normal' interactions".format(normal_count))

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.collect()

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)))
```

```

rdd.take(2)
Out[82]:
[[-122.230000, 37.880000, 41.000000, 880.000000, 129.000000, 322.000000, 126.000000, 8.325200,
452600.000000',
'-122.220000, 37.860000, 21.000000, 7099.000000, 1106.000000, 2401.000000, 1138.000000, 8.301400,
00, 358500.000000']

rdd = rdd.map(lambda line: line.split(","))
rdd.take(2)
Out[83]: [['-122.230000', '37.880000', '41.000000', '880.000000', '129.000000',
'322.000000', '126.000000', '8.325200', '452600.000000'],
['-122.220000', '37.860000', '21.000000', '7099.000000', '1106.000000', '2401.000000', '1138.000000', '8.301400',
'358500.000000']]

rdd.first()
Out[85]: ['-122.230000', '37.880000', '41.000000', '880.000000', '129.000000',
'322.000000', '126.000000', '8.325200', '452600.000000']

rdd.top(2)
Out[86]: [['-124.350000', '40.540000', '52.000000', '1820.000000', '300.000000',
'806.000000', '270.000000', '3.014700', '94600.000000'], ['-124.300000', '41.840000',
'17.000000', '2677.000000', '531.000000', '1244.000000', '456.000000', '3.031300',
'103600.000000']]]

from pyspark.sql import Row
df = rdd.map(lambda line: Row(longitude=line[0],
                               latitude=line[1],
                               housingMedianAge=line[2],
                               totalRooms=line[3],
                               totalBedRooms=line[4],
                               population=line[5],
                               households=line[6],
                               medianIncome=line[7],
                               medianHouseValue=line[8])).toDF()
df.show()
+-----+
| longitude| latitude|housingMedianAge| totalRooms|totalBedRooms| population|households|medianIncome|medianHouseValue|
+-----+
| -122.230000|37.880000| 41.000000| 880.000000| 129.000000| 322.000000| 126.000000| 8.325200| 452600.000000|
| -122.220000|37.860000| 21.000000| 7099.000000| 1106.000000| 2401.000000| 1138.000000| 8.301400| 358500.000000|
| -122.240000|37.850000| 52.000000| 1467.850000| 198.000000| 496.000000| 177.000000| 7.257400| 352180.000000|
| -122.250000|37.850000| 52.000000| 1274.800000| 235.000000| 558.000000| 219.000000| 5.643100| 341380.000000|
| -122.250000|37.850000| 52.000000| 1627.000000| 280.000000| 565.000000| 259.000000| 3.846200| 342200.000000|
| -122.250000|37.850000| 52.000000| 919.000000| 213.000000| 413.000000| 193.000000| 4.036800| 260788.000000|
| -122.250000|37.840000| 52.000000| 2555.000000| 489.000000| 1094.000000| 514.000000| 3.659100| 299280.000000|
| -122.250000|37.840000| 52.000000| 3104.000000| 687.000000| 1157.000000| 647.000000| 3.126000| 241400.000000|
| -122.250000|37.840000| 42.000000| 2555.000000| 665.000000| 1296.000000| 595.000000| 2.088400| 226788.000000|
| -122.250000|37.840000| 52.000000| 3549.000000| 707.000000| 1551.000000| 714.000000| 3.691200| 261180.000000|
| -122.250000|37.840000| 52.000000| 2292.000000| 434.000000| 910.000000| 462.000000| 3.283100| 281580.000000|
| -122.250000|37.840000| 52.000000| 3593.000000| 752.000000| 1594.000000| 734.000000| 3.270500| 241800.000000|
| -122.250000|37.840000| 52.000000| 2491.000000| 474.000000| 1098.000000| 468.000000| 3.075000| 213500.000000|
| -122.250000|37.840000| 52.000000| 696.000000| 191.000000| 345.000000| 174.000000| 2.673600| 191380.000000|
| -122.250000|37.850000| 52.000000| 2643.000000| 626.000000| 1212.000000| 620.000000| 1.916700| 159280.000000|
| -122.250000|37.850000| 50.000000| 1120.000000| 283.000000| 697.000000| 264.000000| 2.125000| 140900.000000|
| -122.270000|37.850000| 52.000000| 1966.000000| 347.000000| 793.000000| 331.000000| 2.775000| 152580.000000|
| -122.270000|37.850000| 52.000000| 1228.000000| 293.000000| 648.000000| 363.000000| 2.128200| 155580.000000|
| -122.260000|37.840000| 50.000000| 2239.000000| 455.000000| 990.000000| 419.000000| 1.991100| 158780.000000|
| -122.270000|37.840000| 52.000000| 1503.000000| 298.000000| 690.000000| 275.000000| 2.603300| 162980.000000|
+-----+
only showing top 28 rows

df.printSchema()
+-----+
| from pyspark.ml.feature import StandardScaler
standardScaler = StandardScaler(inputCol="features", outputCol="features_scaled")
scaler = standardScaler.fit(df)
scaled_df = scaler.transform(df)
scaled_df.take(2)
Out[100]: [Row(label=4.526, features=DenseVector([129.0, 322.0, 126.0, 8.3252, 6.9841,
2.5556, 0.1466]), features_scaled=DenseVector([0.3062, 0.2843, 0.3296, 4.3821, 2.8228,
0.2461, 2.5264])), Row(label=3.585, features=DenseVector([1106.0, 2401.0, 1138.0,
8.3014, 6.2381, 2.1098, 0.1558]), features_scaled=DenseVector([2.6255, 2.1202, 2.9765,
4.3696, 2.5213, 0.2031, 2.6851]))]
train_data, test_data = scaled_df.randomSplit([.8,.2],seed=1234)

from pyspark.ml.regression import LinearRegression
lr = LinearRegression(labelCol="label", maxIter=10, regParam=0.3,
elasticNetParam=0.8)
linearModel = lr.fit(train_data)

predicted = linearModel.transform(test_data)
predictions = predicted.select("prediction").rdd.map(lambda x: x[0])
labels = predicted.select("label").rdd.map(lambda x: x[0])
predictionAndLabel = predictions.zip(labels).collect()
predictionAndLabel[:5]
Out[103]: [(1.5755352395152893, 0.14999), (1.734123382776406, 0.225),
(1.531875281086717, 0.388), (1.362861789920583, 0.394), (1.286298256596895, 0.396)]
linearModel.coefficients
linearModel.coefficients
linearModel.summary.rootMeanSquaredError
Out[107]: 0.875727593215161
linearModel.summary.r2
Out[108]: 0.42060513228120754

```

Figure 3.11: Snapshots of Assignment 8

3.3 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 3.1: Meeting Details

Sl. No.	Date	Name with USN	Meeting contents	Members present	Outcome
1	02/09/2021	Madhusudhana B M 4JN18CS043	Django	16	Full stack python with Django working with relational, MongoDB and REST API using Django
2	09/09/2021	Madhusudhana B M 4JN18CS043	Flask	16	Micro web framework Flex with interfaces to open-source cloud databases
3	14/09/2021	Madhusudhana B M 4JN18CS043	Cassandra	16	Interfaces to open-source cloud databases- Cassandra, MongoDB and Elastic search
4	23/09/2021	Madhusudhana B M 4JN18CS043	Pyspark	12	Exploring Big data and Machine learning on big data using PySpark

3.4 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: Organisation, Prioritisation, 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 ourselves right from within, thus making us thrive in whatever situation we find ourselves.

Chapter 4

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.

4.1 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
- Django – 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 CassandraPySpark 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 Sports Equipments management online using Python Flask Microweb framework.

At first, a new flask project was created and a connection is made to MongoDB Atlas Connection.

```
DB_URI = "mongodb+srv://madhubm:8861122380@cluster0.qxgty.mongodb.net/myFirstDatabase?retryWrites=true&w=majority"
app.config["MONGODB_HOST"] = DB_URI
db = MongoEngine()
db.init_app(app)
```

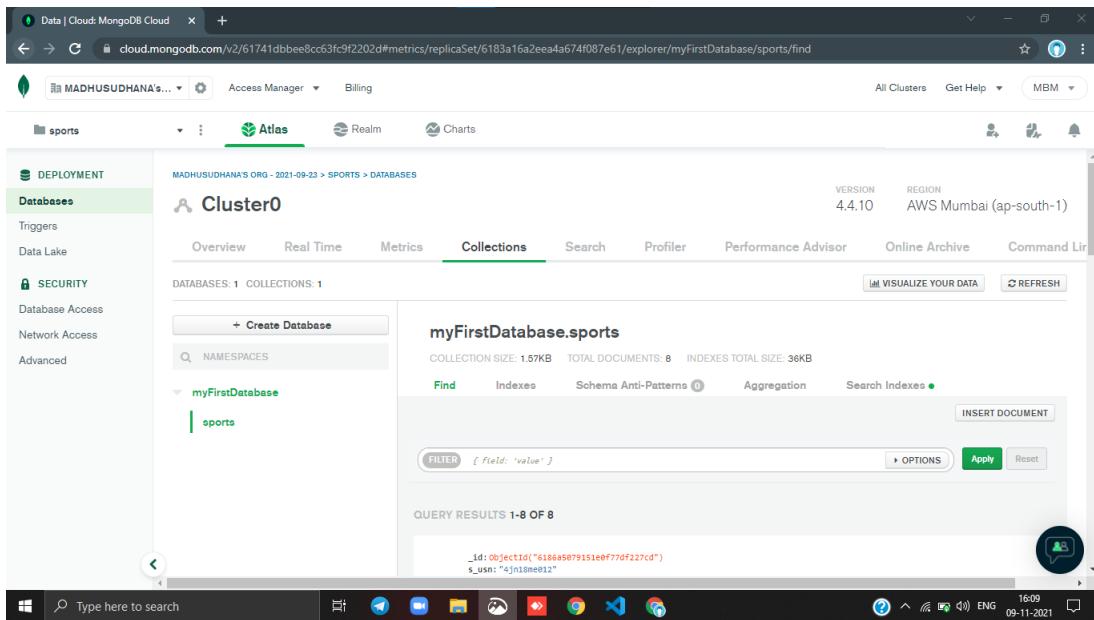


Figure 4.1: Snapshot of MongoDB Atlas Connection

A below picture is front Home page of College Sport Equipments Management System with Main Menu option to select different fields.

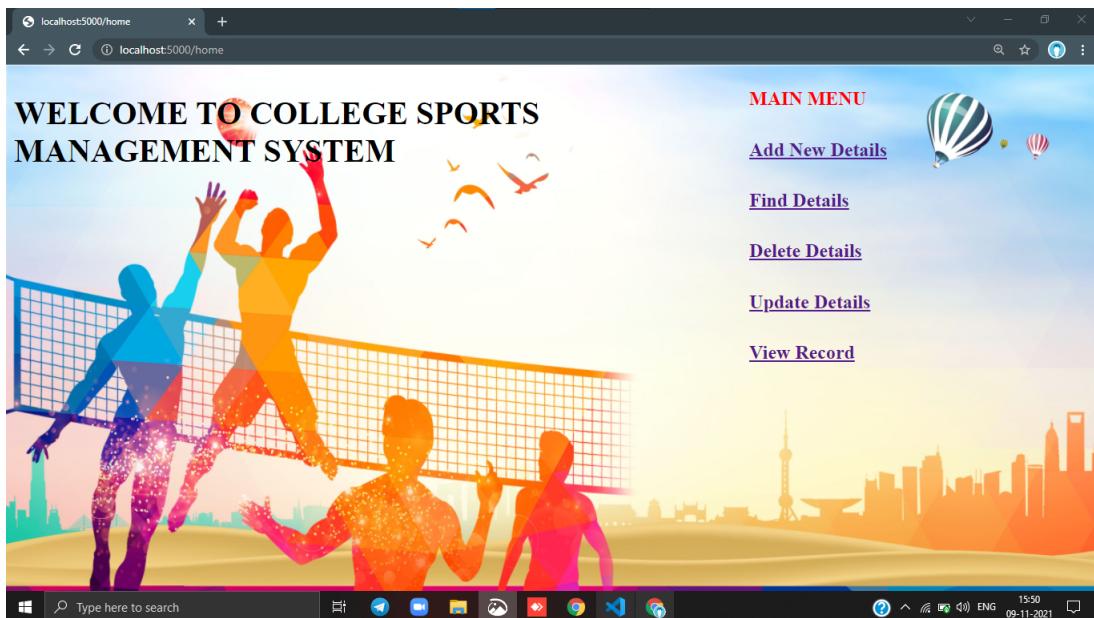


Figure 4.2: Snapshot of Home Page

Then some database were created in database inorder to store the user, details ,equipments data using Python Flask.

```
@app.route('/add', methods=['GET', 'POST'])
def add():
    if request.method=="GET":
        return render_template("add.html")
    else:
        x={
            "s_usn":request.form['s_usn'],
            "s_name":request.form['s_name'],
            "s_equipment":request.form['s_equipment'],
            "s_quantity":request.form['s_quantity'],
            "s_department":request.form['s_department'],
            "s_taken":request.form['s_taken'],
            "s_return":request.form['s_return'],
            "s_costrent":request.form['s_costrent'],
        }
        x=json.dumps(x)
        response =
        requests.post(url="http://127.0.0.1:5000/", data=x)
        return response.text
```



Figure 4.3: Snapshot of Data Entry Page

And find page was created inorder to search weather the user record is present in

database system or not through user ID:

```
@app.route('/find', methods=['GET', 'POST'])
def find():
    if request.method=="GET":
        return render_template("find.html", r=0)
    else:
        s_usn=request.form['s_usn']
        response =
        requests.get(url="http://127.0.0.1:5000/",
                      params={"s_usn":s_usn})
        try:
            loaded_json = json.loads(response.json())
            return render_template("find.html", r=100, result=loaded_json)

        except:
            return
    render_template
("find.html",r=200,error="No such records related to given ID")
```

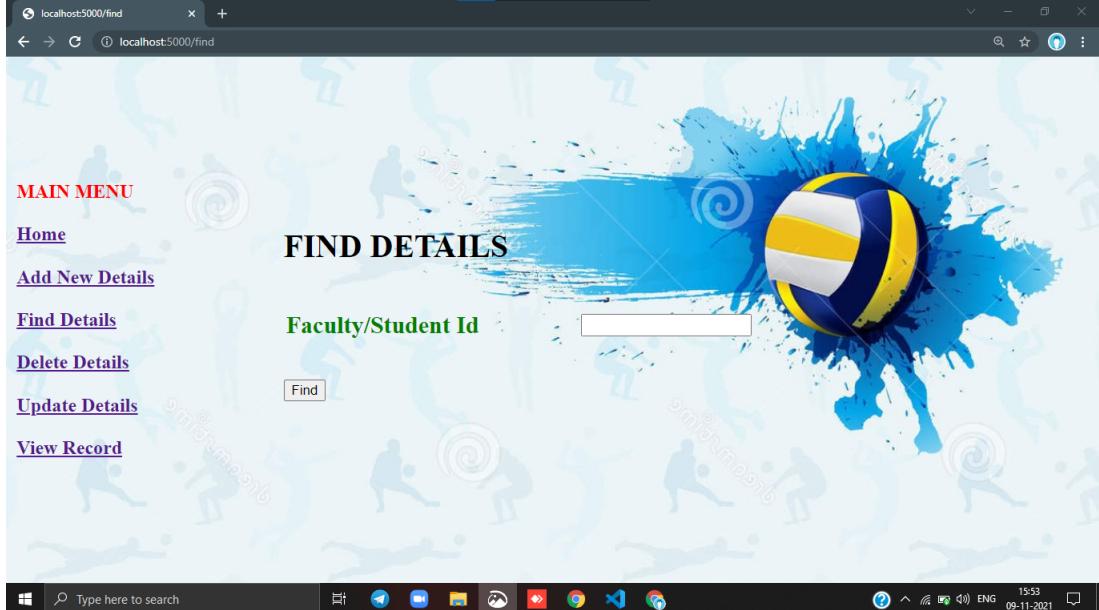


Figure 4.4: Snapshot of Find Page

Further if the particular user record present in the database need to be deleted then user ID will entered and that user record will be deleted from database :

```
@app.route('/delete', methods=['GET', 'POST'])
def delete():
```

```

if request.method=="GET":
    return render_template("del.html", show=100)
else:
    x={
        "s_usn":request.form['s_usn'],
    }
x=json.dumps(x)
response =
requests.delete(url="http://127.0.0.1:5000/", data=x)
if(response.text=="None"):
    return render_template("del.html", show=200)
else:
    return render_template("del.html", show=300)

```

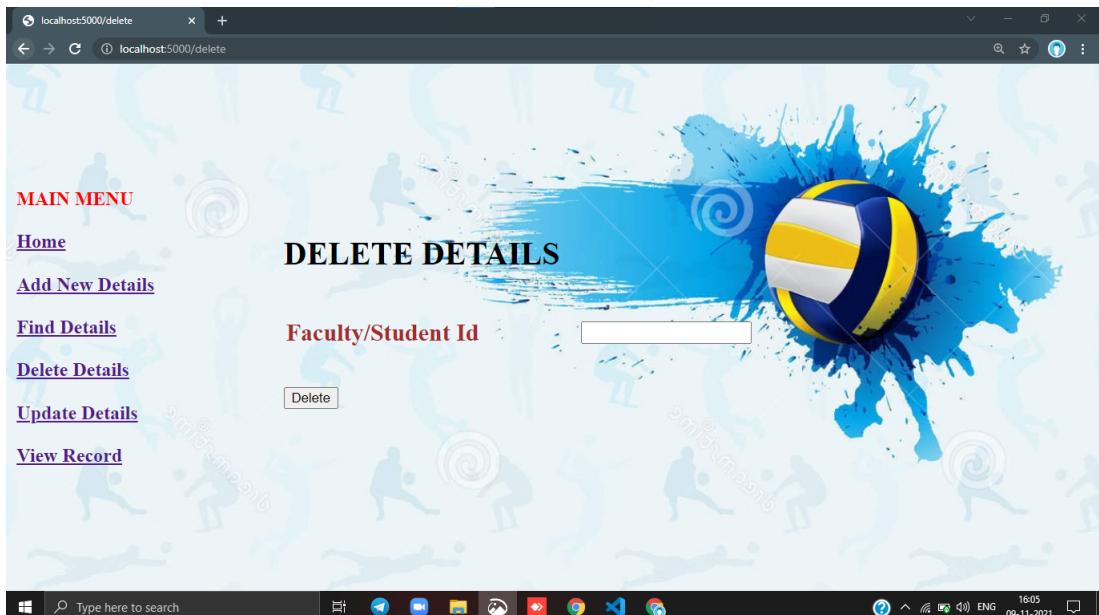


Figure 4.5: Snapshot of Delete Page

A update page is helped to re-correct user record in database system through user ID and we can replace fault data correctly in the database system:

```

@app.route('/update', methods=['GET', 'POST'])
def update():
    if request.method=="GET":
        return render_template("update.html")
    else:
        x={
            "s_usn":request.form['s_usn'],

```

```

    "s_name": request.form['s_name'],
    "s_equipment": request.form['s_equipment'],
    "s_quantity": request.form['s_quantity'],
    "s_department": request.form['s_department'],
    "s_taken": request.form['s_taken'],
    "s_return": request.form['s_return'],
    "s_costrent": request.form['s_costrent'],
}
x=json.dumps(x)
response =
requests.put(url="http://127.0.0.1:5000/",
data=x)
return response.text

```

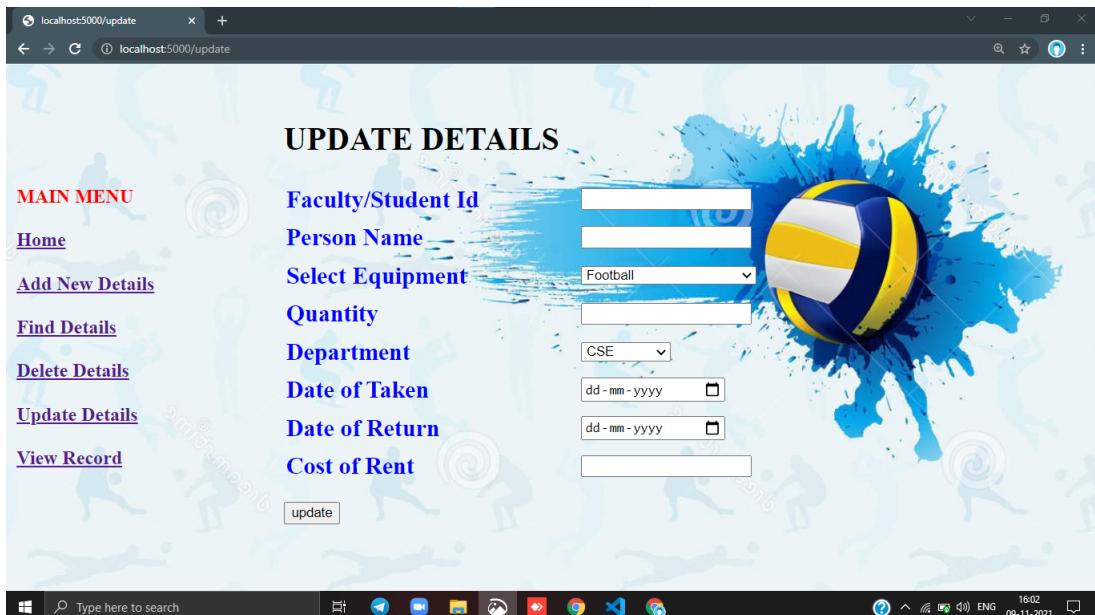


Figure 4.6: Snapshot of Update Page

Atlast record page help to find number of person who taken equipments on that particular day through using date of taken and find all option will help to display overall data present in the database system :

```

@app.route('/record', methods=['GET', 'POST'])
def record():
    if request.method=="GET":
        return render_template("record.html", r=0)
    else:
        s_taken=request.form['s_taken']

```

```
response =
    requests.get(url="http://127.0.0.1:5000/" ,
    params={"s_taken":s_taken})
try:
    loaded_json = json.loads(response.json())
    return render_template("record.html",r=400,result=loaded_json)

except:
    return
render_template
("record.html",r=300,error="No such records related to given ID")
```



Figure 4.7: Snapshot of Record Page

The below picture show the overall data present in database system and we can generate a softcopy of overall data by clicking on download option.

Faculty/Student ID	Person Name	Department	Equipment	Quantity	Date of Taken	Date of Return	Cost of Rent
4jn18me012	kushal	MECH	Basekat ball	2	2021-11-06	2021-11-26	456
4jn18me13	gozeen	MECH	Chess Borad Equipment	1	2021-11-06	2021-11-20	263
4jn18cs024	Darashan cs	CSE	Basekat ball	2	2021-11-07	2021-11-19	542
4jn18cs025	Devesh	CSE	Football	1	2021-11-07	2021-11-20	544
4jn18cs048	moin m	CSE	Football	2	2021-11-07	2021-11-20	412
4jn18cs049	Nagaraj t	CSE	Carrom Board Equipment	1	2021-11-07	2021-11-18	321
4jn18cs046	zama m	CSE	Table Tennis Equipment	2	2021-11-07	2021-11-12	200
5231523	james	DEPLOMA	Carrom Board Equipment	1	2021-11-08	2021-11-13	854

Figure 4.8: Snapshot of Overall Data Page

Then downloaded softcopy will be directly saved in PC in csv file format and user can take hardcopy print out for reference:

Faculty/Student ID	Person Name	Department	Equipment	Quantity	Date of Taken	Date of Return	Cost of Rent
4jn18me012	kushal	MECH	Basekat ball	2	06-11-2021	26-11-2021	456
4jn18me13	gozeen	MECH	Chess Borad Equipment	1	06-11-2021	20-11-2021	263
4jn18cs024	Darashan cs	CSE	Basekat ball	2	07-11-2021	19-11-2021	542
4jn18cs025	Devesh	CSE	Football	1	07-11-2021	20-11-2021	544
4jn18cs048	moin m	CSE	Football	2	07-11-2021	20-11-2021	412
4jn18cs049	Nagaraj t	CSE	Carrom Board Equipment	1	07-11-2021	18-11-2021	321
4jn18cs046	zama m	CSE	Table Ten	2	07-11-2021	12-11-2021	200
5231523	james	DEPLOMA	Carrom Board Equipment	1	08-11-2021	13-11-2021	854

Figure 4.9: Snapshot of Downloaded CSV File

4.2 Experiences

My internship as a data analytics intern for 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 learnt 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.

Chapter 5

Conclusion

In this project, I have used Flask micro web development framework for creating Sports Equipments Management System. I have used MongoDB Atlas which is a cloud-based Database System for my project which is useful in storing data in cloud. I have implemented different database operations like adding users , finding users , deleting users , updating users and retrieving overall data present in Database System. I have created file download system to generate the softcopy of data present in Database System if admin needed. I believe that this project is useful, light weight due to the use of Flask and helping in storing the data in structured way to the Database.

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>