KLE Technological University
Creating Value
Leveraging Knowledge

INTERNSHIP REPORT

Mr. SUDEEP.V. PATIL

01FE20BME154

AT

Springeveing private limited

-

Springeveing Private
Limited, Shri Vijaya
concrete works, Badami
Road, Bagalkote-587101

This report is submitted to,

SCHOOL OF MECHANICAL ENGINEERING

Experience Engineering @
School of Mechanical
EngineeringKLETECH
Engage | Explore | Excel | Evolve

KLE TECHNOLOGICAL UNIVERSITY

Hubballi 2023-24

1. INTRODUCTION

1.1 Company Introduction

The springeveing private limited software startup prides itself on its unwavering dedication to serving the tier 3 market, startups, and local businesses, recognizing the myriad of unique needs and challenges that enterprises within these segments encounter. Central to their mission is the provision of meticulously crafted, high-quality software solutions specifically tailored to alleviate the myriad challenges faced by startups and local businesses alike. What truly sets apart is their steadfast commitment to going above and beyond the conventional one-size-fits-all approach. Instead, we prioritize customization, ensuring that each and every one of their clients receives solutions meticulously tailored to their individual circumstances and requirements. This dedication to personalized service underscores their commitment to paving the way for the success of their clients, empowering them to thrive in their respective markets and endeavour's.

1.2 Training Objectives:

The purpose of Industrial Training is to expose students to real work environment experience and at the same time to gain the knowledge through hands-on experience, learning and execution. From industrial training, students will also develop skills in work ethics, communication, management and others. Moreover, this practical training program allows students to relate theoretical knowledge with its application in the manufacturing industry.

- With the help of industrial training, students can enhance the ability, skills and real work exposure.
- Students can increase their knowledge, experience in the individual technical fields.
- Students can get a personality development skill to find a better opportunity in the companies.
- In the Training, students can increase the self-confidence to deal with real world problems.
- Students can understand the work responsibility as per company standard level.
- To cultivate the student's leadership ability and responsibility to perform or execute the given task.

2. Company's background/ Profile

2.1 Mission

Create an environment where anyone can learn, innovate and excel.

2.2 SEO

The improvement of organic visibility in search engine results for our clients' websites is aimed by our SEO services. By optimizing on-page elements, conducting keyword research, and implementing effective link-building strategies, higher rankings are facilitated and relevant organic traffic is attracted by businesses.

2.3 Website development

Stunning, user-friendly websites that align with our clients' goals and capture their brand essence are specialized in by us. The latest technologies are leveraged by our development team to deliver high-quality websites tailored to each client's unique requirements.

2.4 Digital marketing

Comprehensive digital marketing strategies aimed at increasing brand visibility and driving targeted traffic to our clients' websites are provided by us. Creating impactful campaigns across various channels, including social media, search engines, email marketing, and more, is where our team excels.

2.5 Training/Internship

At our company, the strong belief that with proper training, anyone can code is held. Training programs and internship opportunities are offered to aspiring developers, equipping them with the required soft skills and knowledge needed to succeed in the software industry.

2.6 Guiding Principles

Thinking customer, company and individual.

Be number one in customer trust.

Attract, retain and develop the best talent.

Deliver best-in-class solutions.

Achieve financial goals to deliver shareholder value.

Act with purpose for a better world.

2.7 Core Values

Achievement

Agility

Honesty & Integrity

Inclusion & Diversity

Innovation & continuous improvement

Mutual trust & respect

Open communication

Ownership & accountability

Teamwork

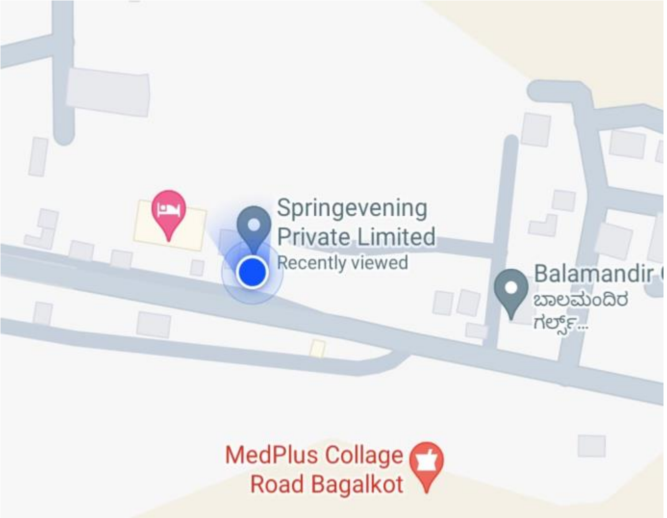
2.8 Company (Springevening Private Limited) CEO:

Shyamsunder Sedamkar,

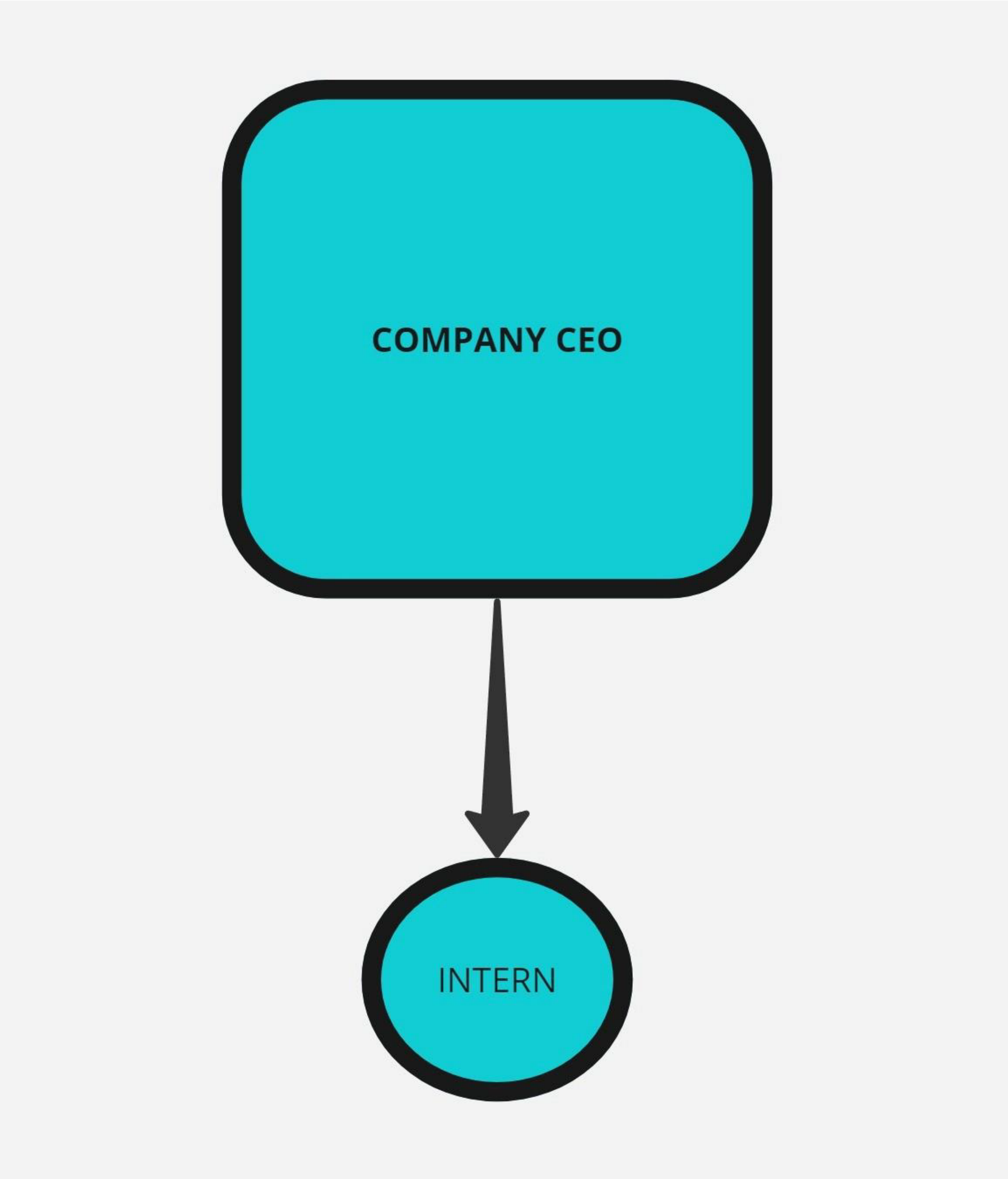
Shyamsunder Sedamkar is a technology startup founder with a focus on creating hyper-local solutions that have a tangible impact on communities. Shyamsunder holds a bachelor degree from Karnataka University and M.S from Illinois Institute of Technology, Chicago. With a background in software development and a particular emphasis on the media industry, Shyamsunder has a deep understanding of the latest trends and opportunities in the field. Demonstrating a proven track record, Shyamsunder has successfully led teams and brings a wealth of experience in product development, marketing, and advertising.

Above all, Shyamsunder is an Engineer who possesses a strong passion for problem-solving. As a technology enthusiast, Shyamsunder is always eager to expand his knowledge by learning from others and embracing new opportunities.

2.9 Company location and picture



2.10 Organization chart



2.11 Company History

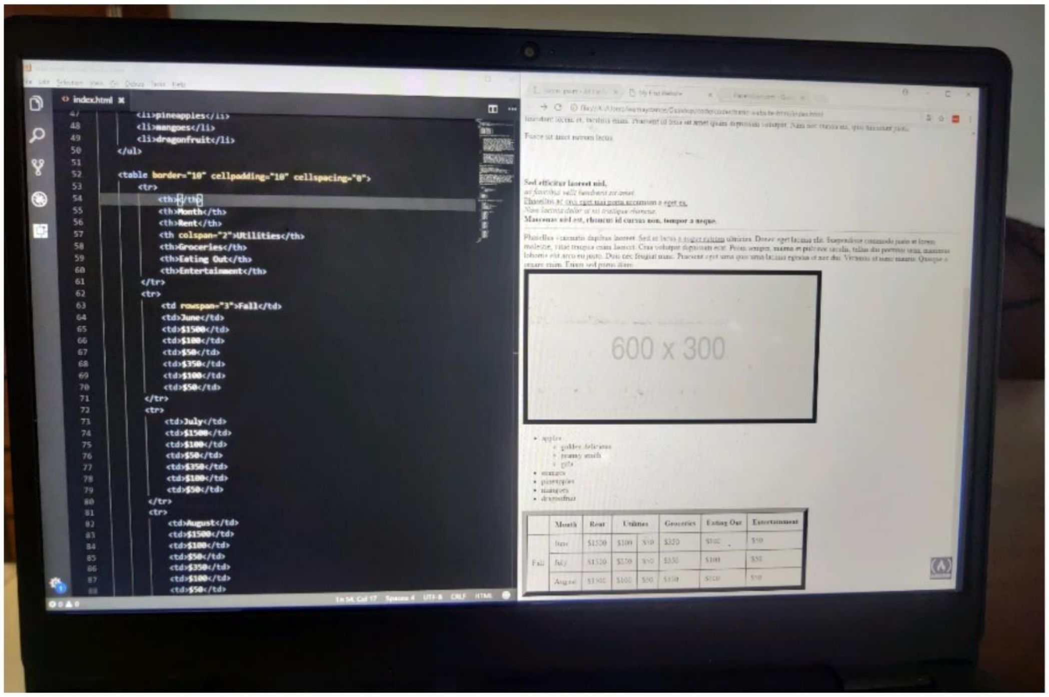
2022 - Company founded by Shyamsunder Sedamka in Bagalkot, Karnataka

2023 – started taking projects from market

3.WEEKLY REPORT:

3.1 Week 1

DATE	**WORK DONE on HTML (Hyper Text Markup Language)**
JAN 8	Orientation and introduction
JAN 9	heading elements
JAN 10	paragraph elements - self closing tags
JAN 11	nesting and indentation - anchor element
JAN 12	image elements



```
        <li>pineapples</li>
        <li>mangoes</li>
        <li>dragonfruit</li>
    </ul>

    <table border="10" cellpadding="10" cellspacing="0">
        <tr>
            <th></th>
            <th>Month</th>
            <th>Rent</th>
            <th colspan="2">Utilities</th>
            <th>Groceries</th>
            <th>Eating Out</th>
            <th>Entertainment</th>
        </tr>
        <tr>
            <td rowspan="3">Fall</td>
            <td>June</td>
            <td>$1500</td>
            <td>$100</td>
            <td>$50</td>
            <td>$350</td>
            <td>$100</td>
            <td>$50</td>
        </tr>
        <tr>
            <td>July</td>
            <td>$1500</td>
            <td>$100</td>
            <td>$50</td>
            <td>$350</td>
            <td>$100</td>
            <td>$50</td>
        </tr>
        <tr>
            <td>August</td>
            <td>$1500</td>
            <td>$100</td>
            <td>$50</td>
            <td>$350</td>
            <td>$100</td>
            <td>$50</td>
```

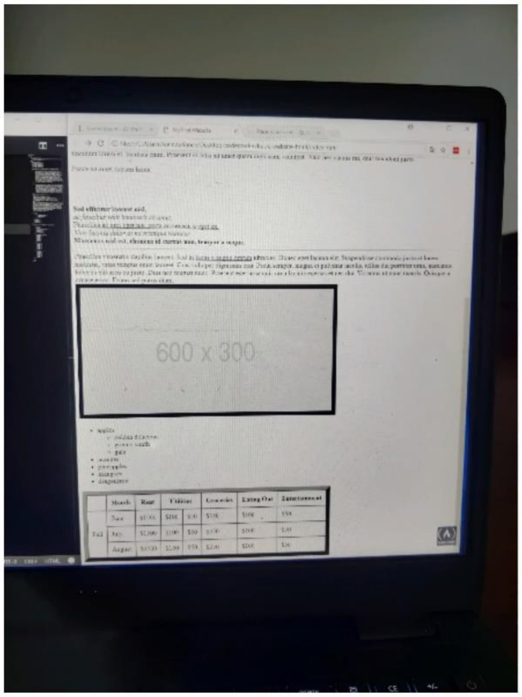
Sed efficitur laoreet nisl,

Maecenas nisl est, rhoncus id cursus non, tempor a neque.

600 x 300

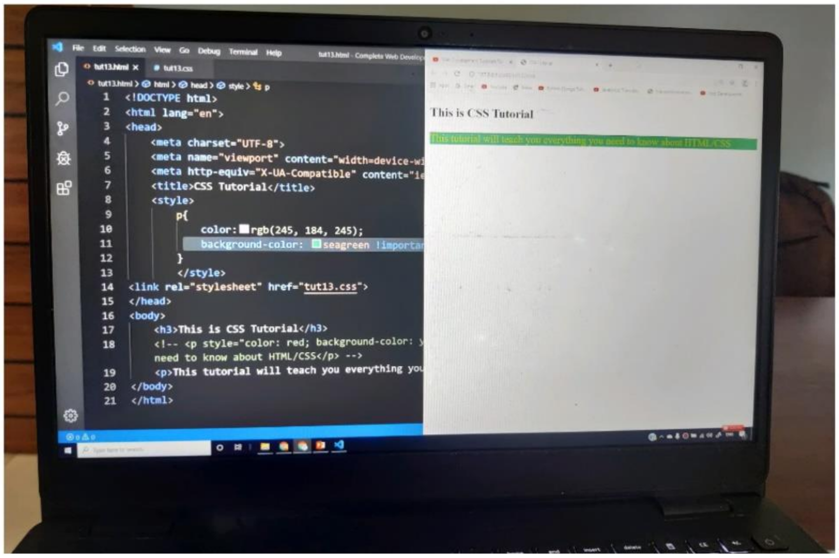
- apples
  - golden delicious
  - granny smith
  - gala
- oranges
- pineapples
- mangoes
- dragonfruit

	Month	Rent	Utilities		Groceries	Eating Out	Entertainment
Fall	June	$1500	$100	$50	$350	$100	$50
	July	$1500	$100	$50	$350	$100	$50
	August	$1500	$100	$50	$350	$100	$50



3.2 Week 2

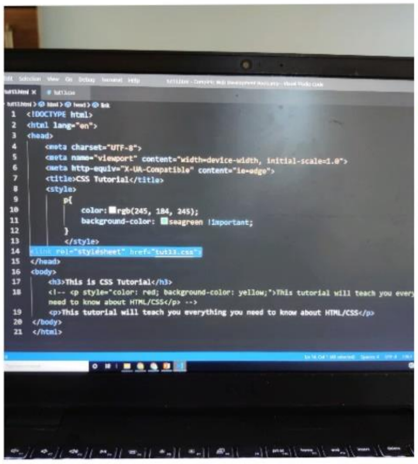
DATE	**WORK DONE on CSS (cascading style sheets)**
JAN 15	CSS colour - font properties - inspection css
JAN 16	CSS box model - margin, padding and border
JAN 17	CSS positioning - CSS display and float
JAN 18	media queries - flex (display, direction, layout and sizing)
JAN 19	Grid (display, direction, layout and sizing)



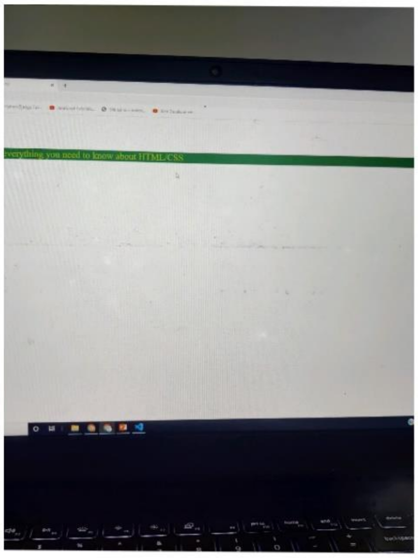
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-wi
    <meta http-equiv="X-UA-Compatible" content="ie
    <title>CSS Tutorial</title>
    <style>
        p{
            color: rgb(245, 184, 245);
            background-color: seagreen !importa
        }
        </style>
 <link rel="stylesheet" href="tut13.css">
 </head>
 <body>
     <h3>This is CSS Tutorial</h3>
     <!-- <p style="color: red; background-color: y
     need to know about HTML/CSS</p> -->
     <p>This tutorial will teach you everything you
 </body>
 </html>
```

This is CSS Tutorial

This tutorial will teach you everything you need to know about HTML/CSS



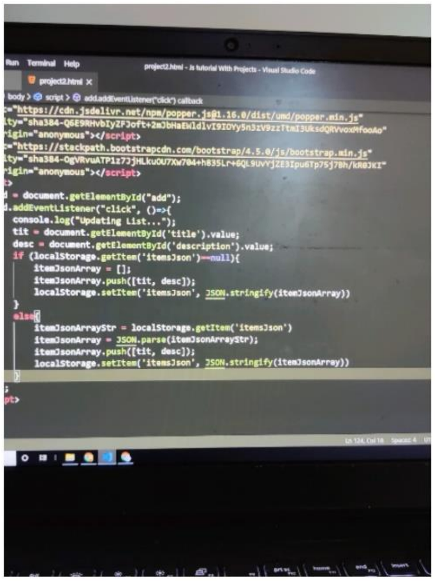
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>CSS Tutorial</title>
    <style>
        p{
            color: rgb(245, 184, 245);
            background-color: seagreen !important;
        }
        </style>
 <link rel="stylesheet" href="tut13.css">
 </head>
 <body>
     <h3>This is CSS Tutorial</h3>
     <!-- <p style="color: red; background-color: yellow;">This tutorial will teach you every
     need to know about HTML/CSS</p> -->
     <p>This tutorial will teach you everything you need to know about HTML/CSS</p>
 </body>
 </html>
```



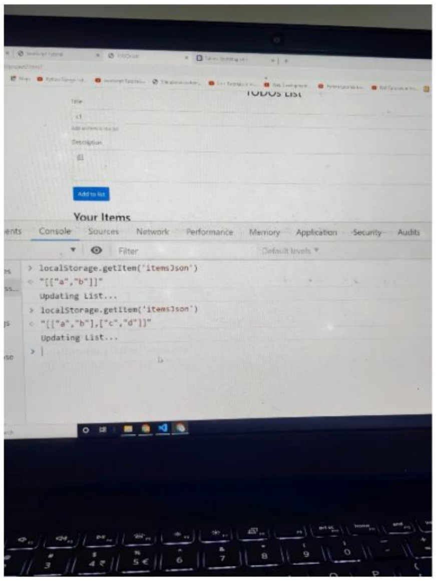
everything you need to know about HTML/CSS

3.3 Week 3

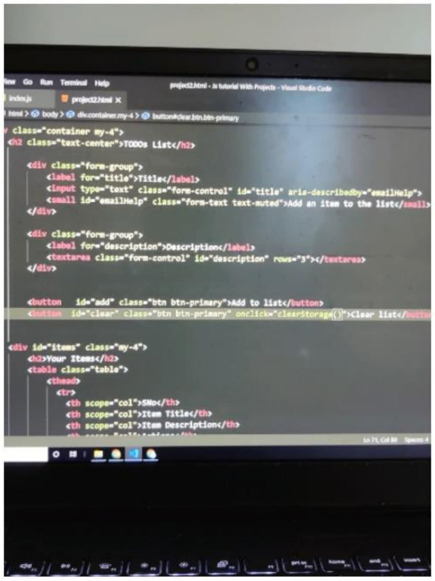
DATE	**WORK DONE on JavaScript**
JAN 22	variables
JAN 23	strings
JAN 24	functions
JAN 25	Arrays - document object model
JAN 26	Event handing


Run Terminal Help
project2.html - Js tutorial With Projects - Visual Studio Code
project2.html ×
body > script > add.addEventListener("click") callback
Ln 124, Col 18 Spaces: 4

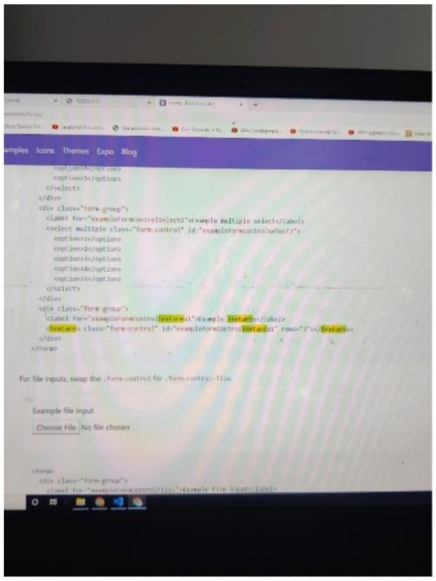
```
c="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
ity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
rigin="anonymous"></script>
c="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
ity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
rigin="anonymous"></script>
t>
d = document.getElementById("add");
d.addEventListener("click", ()=>{
  console.log("Updating List...");
  tit = document.getElementById('title').value;
  desc = document.getElementById('description').value;
  if (localStorage.getItem('itemsJson')==null){
      itemJsonArray = [];
      itemJsonArray.push([tit, desc]);
      localStorage.setItem('itemsJson', JSON.stringify(itemJsonArray))
  }
  else{
      itemJsonArrayStr = localStorage.getItem('itemsJson')
      itemJsonArray = JSON.parse(itemJsonArrayStr);
      itemJsonArray.push([tit, desc]);
      localStorage.setItem('itemsJson', JSON.stringify(itemJsonArray))
  }
;
pt>
```



```
> localStorage.getItem('itemsJson')
< "[["a","b"]]"
  Updating List...
> localStorage.getItem('itemsJson')
< "[["a","b"],["c","d"]]"
  Updating List...
> |
```


View Go Run Terminal Help
project2.html - Js tutorial With Projects - Visual Studio Code
index.js
project2.html ×
html > body > div.container.my-4 > button#clear.btn.btn-primary
Ln 71, Col 80 Spaces: 4

```
v class="container my-4">
 <h2 class="text-center">TODOs List</h2>

    <div class="form-group">
        <label for="title">Title</label>
        <input type="text" class="form-control" id="title" aria-describedby="emailHelp">
        <small id="emailHelp" class="form-text text-muted">Add an item to the list</small>
    </div>

    <div class="form-group">
        <label for="description">Description</label>
        <textarea class="form-control" id="description" rows="3"></textarea>
    </div>


    <button  id="add" class="btn btn-primary">Add to list</button>
    <button  id="clear" class="btn btn-primary" onclick="clearStorage()">Clear list</butto

 <div id="items" class="my-4">
    <h2>Your Items</h2>
    <table class="table">
        <thead>
          <tr>
            <th scope="col">SNo</th>
            <th scope="col">Item Title</th>
            <th scope="col">Item Description</th>
```



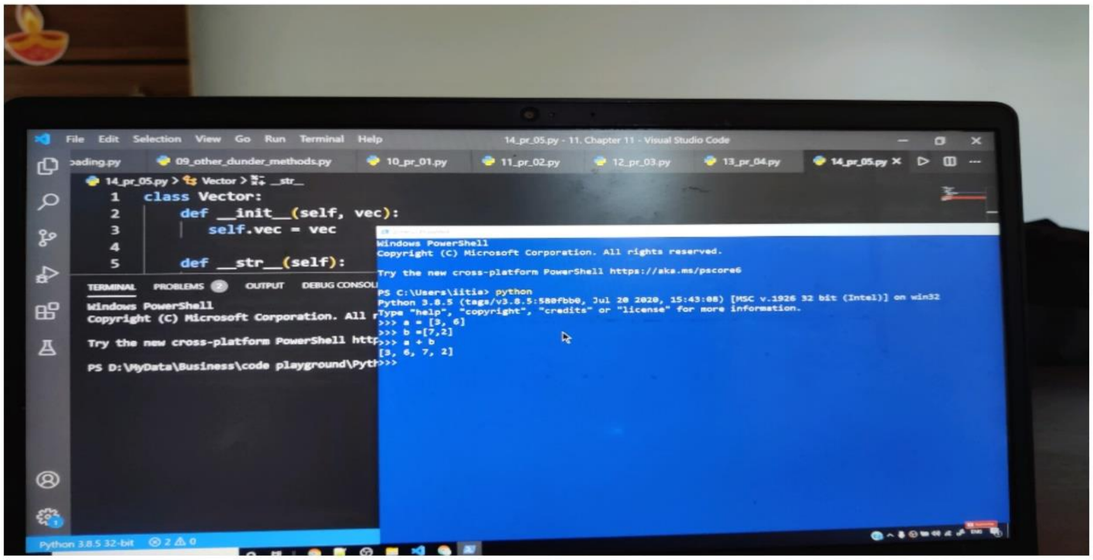
```
      <option>5</option>
    </select>
  </div>
  <div class="form-group">
    <label for="exampleFormControlSelect2">Example multiple select</label>
    <select multiple class="form-control" id="exampleFormControlSelect2">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      <option>4</option>
      <option>5</option>
    </select>
  </div>
  <div class="form-group">
    <label for="exampleFormControlTextarea1">Example textarea</label>
    <textarea class="form-control" id="exampleFormControlTextarea1" rows="3"></textarea>
  </div>
</form>
```

For file inputs, swap the .form-control for .form-control-file.

```
<form>
  <div class="form-group">
    <label for="exampleFormControlFile1">Example file input</label>
```

3.4 Week 4

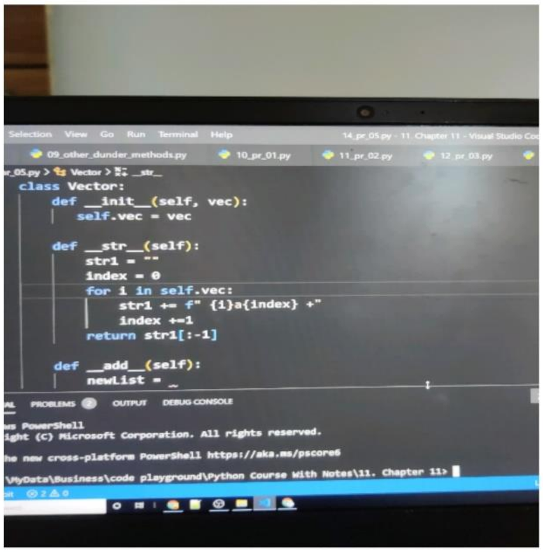
DATE	**WORK DONE on basic python**
JAN 29	Python - Variables
JAN 30	Control structures (if statements, loops) - Functions
JAN 31	Modules - File handling
FEB 1	Integers and floating-point numbers
FEB 2	Strings - Boolean values


File Edit Selection View Go Run Terminal Help
14_pr_05.py - 11. Chapter 11 - Visual Studio Code
oading.py
09_other_dunder_methods.py
10_pr_01.py
11_pr_02.py
12_pr_03.py
13_pr_04.py
14_pr_05.py
14_pr_05.py > Vector > __str__
TERMINAL
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
Python 3.8.5 32-bit
⊗ 2 ⚠ 0

```
class Vector:
    def __init__(self, vec):
        self.vec = vec

    def __str__(self):
```

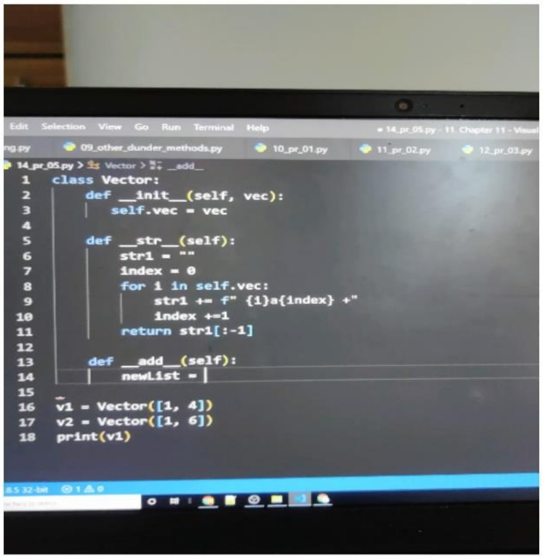
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All r

Try the new cross-platform PowerShell http

PS D:\MyData\Business\code playground\Pyth
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\iitia> python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = [3, 6]
>>> b =[7,2]
>>> a + b
[3, 6, 7, 2]
>>>
```


Selection View Go Run Terminal Help
14_pr_05.py - 11. Chapter 11 - Visual Studio Coc
09_other_dunder_methods.py
10_pr_01.py
11_pr_02.py
12_pr_03.py
r_05.py > Vector > __str__
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
⊗ 2 ⚠ 0

```
class Vector:
    def __init__(self, vec):
        self.vec = vec

    def __str__(self):
        str1 = ""
        index = 0
        for i in self.vec:
            str1 += f" {i}a{index} +"
            index +=1
        return str1[:-1]

    def __add__(self):
        newList = _
```

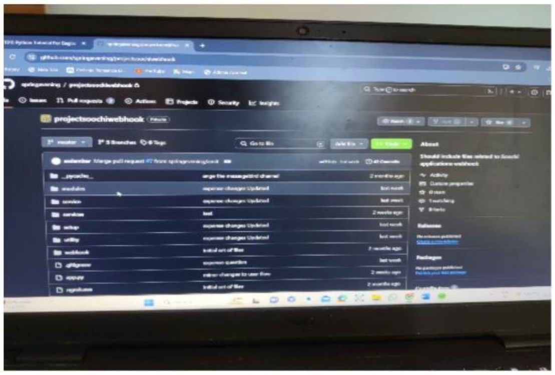
```
ws PowerShell
ight (C) Microsoft Corporation. All rights reserved.

he new cross-platform PowerShell https://aka.ms/pscore6

\MyData\Business\code playground\Python Course With Notes\11. Chapter 11>
```

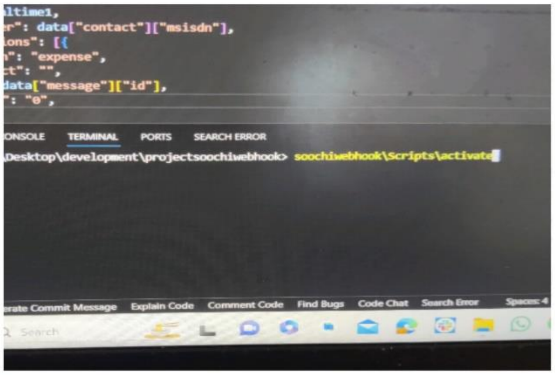

Edit Selection View Go Run Terminal Help
● 14_pr_05.py - 11. Chapter 11 - Visual
ng.py
09_other_dunder_methods.py
10_pr_01.py
11_pr_02.py
12_pr_03.py
14_pr_05.py > Vector > __add__
8.5 32-bit
⊗ 1 ⚠ 0

```
class Vector:
    def __init__(self, vec):
        self.vec = vec

    def __str__(self):
        str1 = ""
        index = 0
        for i in self.vec:
            str1 += f" {i}a{index} +"
            index +=1
        return str1[:-1]

    def __add__(self):
        newList = 

v1 = Vector([1, 4])
v2 = Vector([1, 6])
print(v1)
```

3.5 Week 5

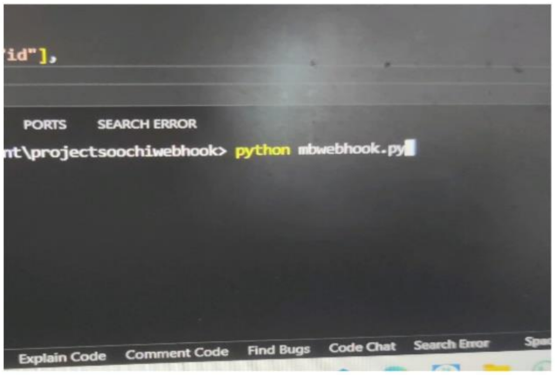
DATE	**WORK DONE on networking**
FEB 5	Git hub – push, pull request, pull, commit, merge and clone
FEB 6	Flask - TRELLO
FEB 7	NG ROCK
FEB 8	Webhook
FEB 9	Message bird





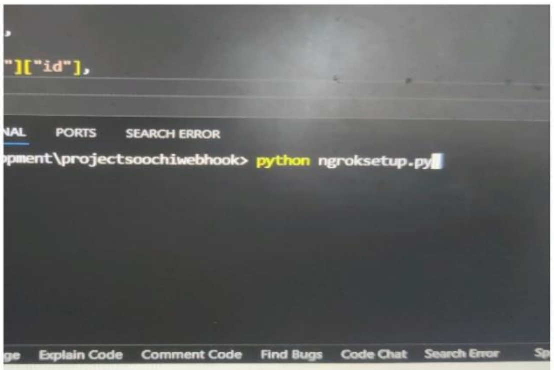
```
ltime1,
r": data["contact"]["msisdn"],
ons": [{
": "expense",
t": "",
data["message"]["id"],
": "0",
```

```
Desktop\development\projectsoochiwebhook> soochiwebhook\Scripts\activate
```



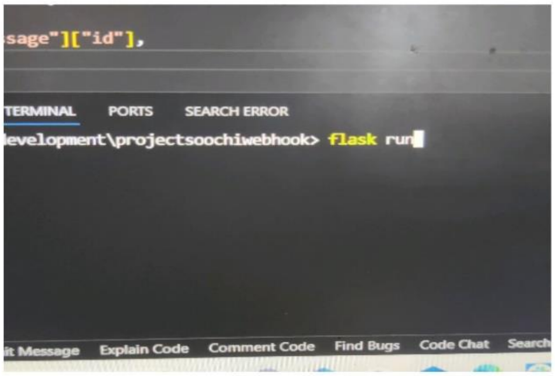
```
id"],
```

```
nt\projectsoochiwebhook> python mbwebhook.py
```



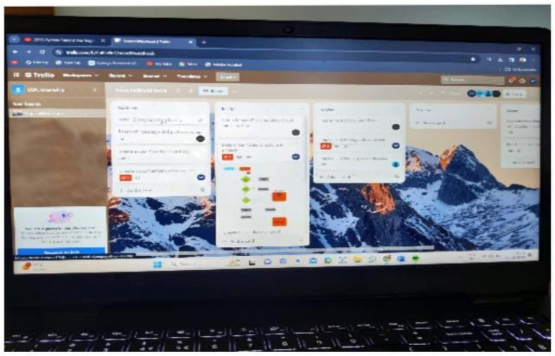
```
"]["id"],
```

```
pment\projectsoochiwebhook> python ngroksetup.py
```



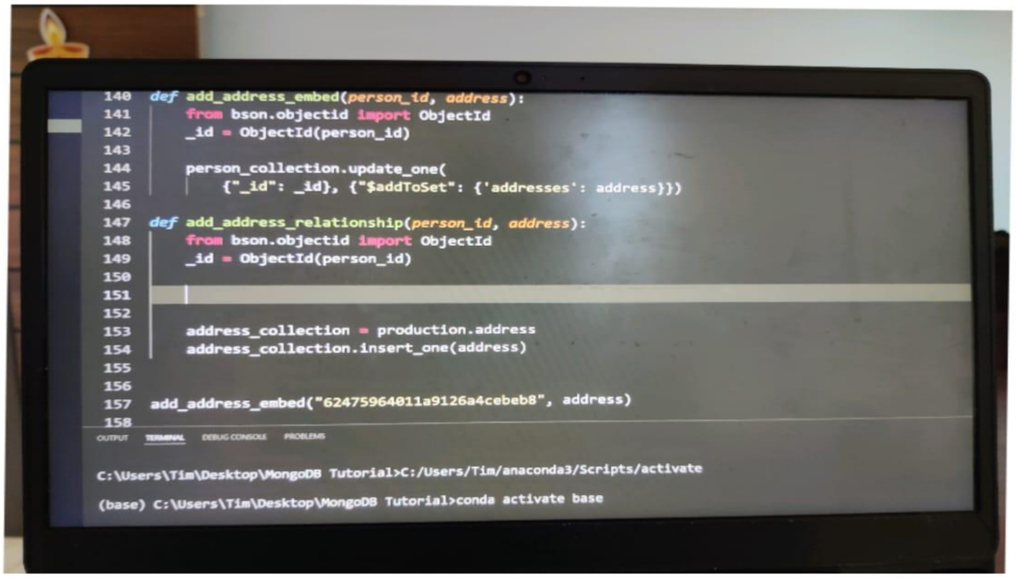
```
sage"]["id"],
```

```
evelopment\projectsoochiwebhook> flask run
```



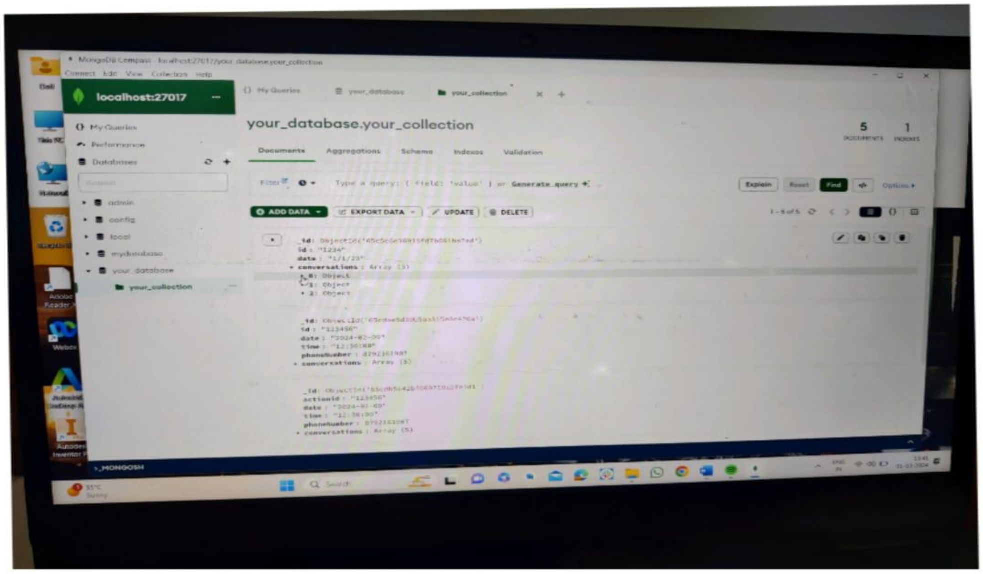
3.6 Week

DATE	**WORK DONE on mongo dB (data base)**
FEB 12	MongoDB Atlas Setup
FEB 13	PyMongo Setup - MongoDB Compass
FEB 14	MongoDB Document Model - Inserting Documents
FEB 15	Reading Documents - Updating Documents
FEB 16	Deleting Documents - Relationships



```
def add_address_embed(person_id, address):
    from bson.objectid import ObjectId
    _id = ObjectId(person_id)

    person_collection.update_one(
        {"_id": _id}, {"$addToSet": {'addresses': address}})

def add_address_relationship(person_id, address):
    from bson.objectid import ObjectId
    _id = ObjectId(person_id)


    address_collection = production.address
    address_collection.insert_one(address)


add_address_embed("62475964011a9126a4cebeb8", address)
```

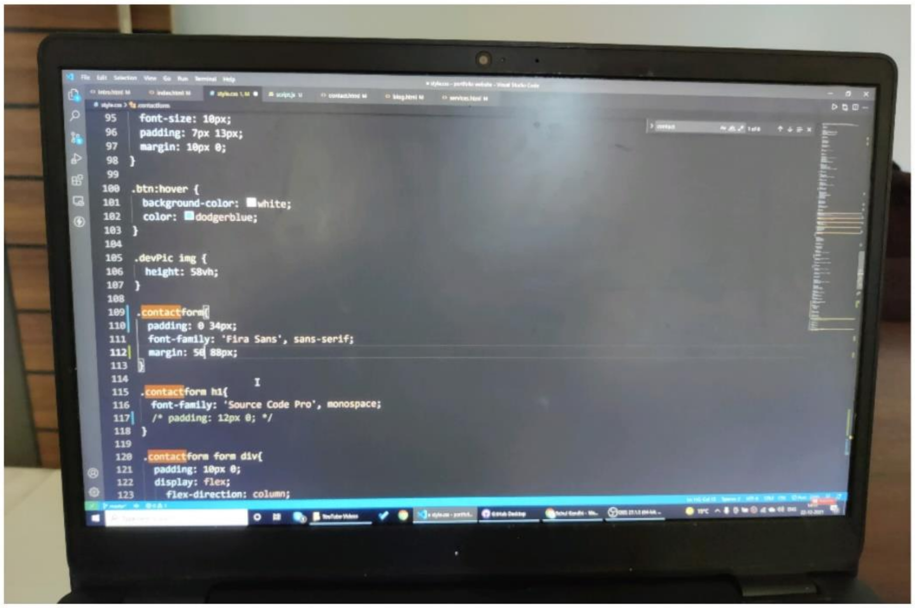
```
C:\Users\Tim\Desktop\MongoDB Tutorial>C:/Users/Tim/anaconda3/Scripts/activate

(base) C:\Users\Tim\Desktop\MongoDB Tutorial>conda activate base
```



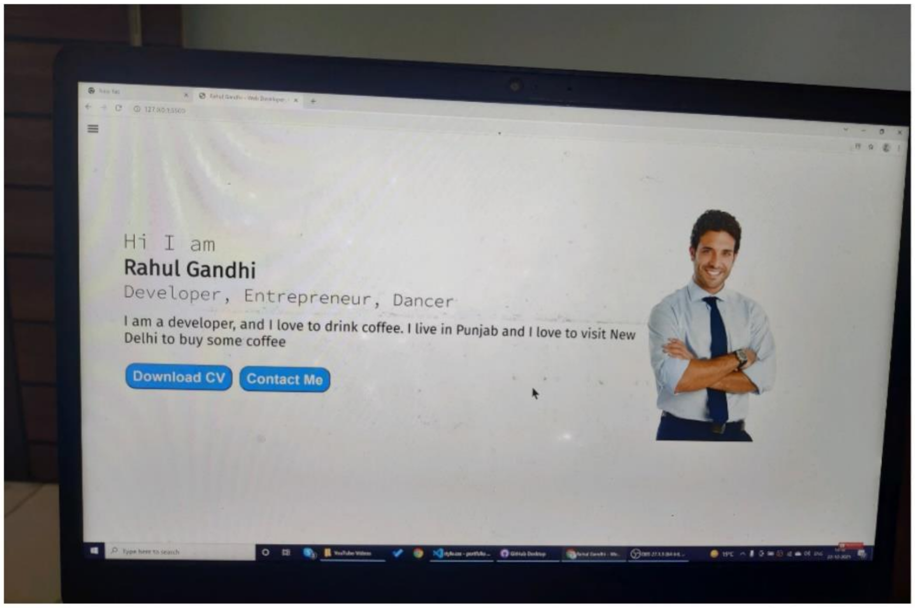
```
_id: ObjectId('[illegible]')
id : "1234"
date : "1/1/23"
conversations : Array (3)
  0: Object
  1: Object
  2: Object

_id: ObjectId('[illegible]')
id : "123456"
date : "2024-02-09"
time : "12:30:00"
phoneNumber : 8792161987
conversations : Array (5)

_id: ObjectId('[illegible]')
actionId : "123456"
date : "2024-02-09"
time : "12:30:00"
phoneNumber : 8792161987
conversations : Array (5)
```

3.7 Week 7

DATE	**WORK DONE on example project**
FEB 19	portfolio website
FEB 20- 22	Creating and example conversation
FEB 23	Storing the example chat in mongo db



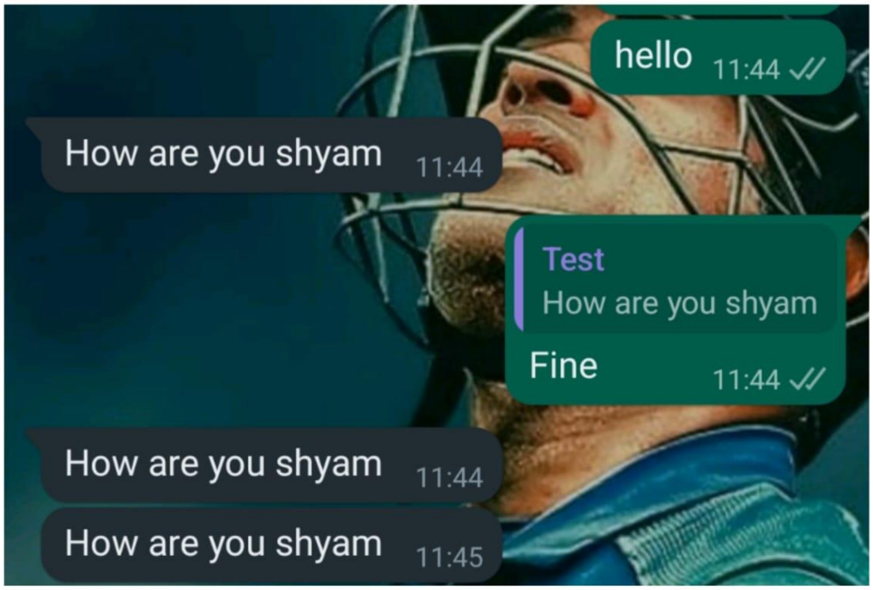
```
  font-size: 10px;
  padding: 7px 13px;
  margin: 10px 0;
}

.btn:hover {
  background-color: white;
  color: dodgerblue;
}

.devPic img {
  height: 58vh;
}

.contactform{
  padding: 0 34px;
  font-family: 'Fira Sans', sans-serif;
  margin: 50 88px;
}

.contactform h1{
  font-family: 'Source Code Pro', monospace;
  /* padding: 12px 0; */
}

.contactform form div{
  padding: 10px 0;
  display: flex;
    flex-direction: column;
```



Hi I am
Rahul Gandhi
Developer, Entrepreneur, Dancer

I am a developer, and I love to drink coffee. I live in Punjab and I love to visit New Delhi to buy some coffee

3.7.1Creating and example conversation



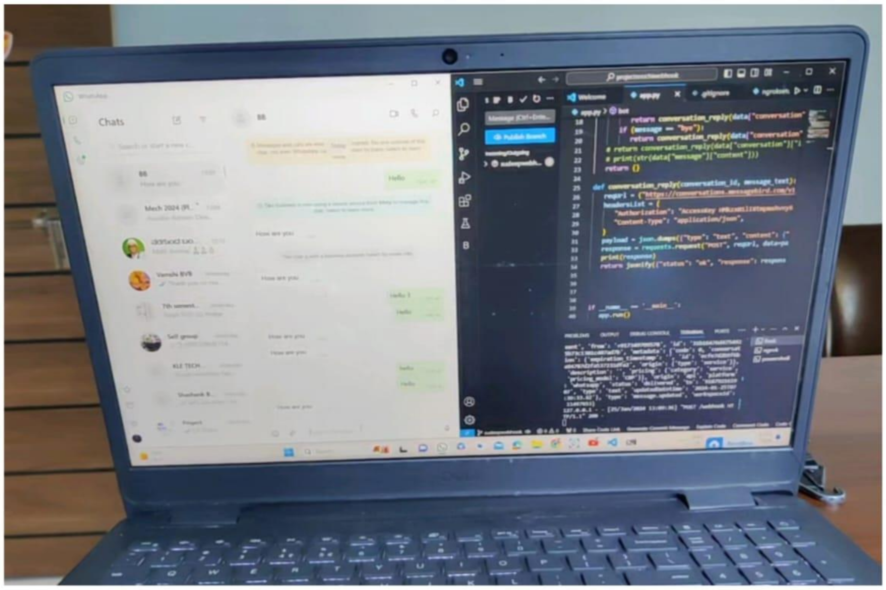
hello 11:44

How are you shyam 11:44

Test
How are you shyam
Fine 11:44

How are you shyam 11:44

How are you shyam 11:45



3.8 Week 8

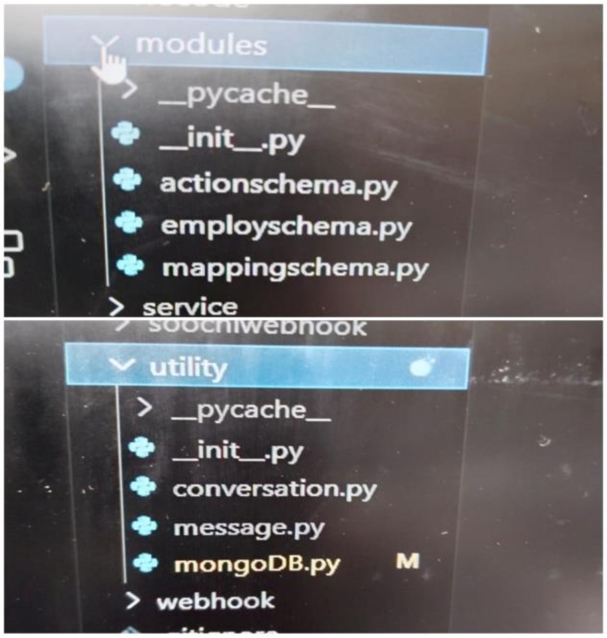
DATE	**WORK DONE**
FEB 26	Working on the previous week example
FEB 27 - 29	Internship training report
FEB 29 - march 1	Discussion and planning of project

3.9 Week 9

DATE	WORK DONE
MAR 04	detailed information about our project
MAR 05	methods, technique and tools that we are going to use in our project
MAR 07	create and setup and files and folders in the vs code workbench

3.10 Week 10

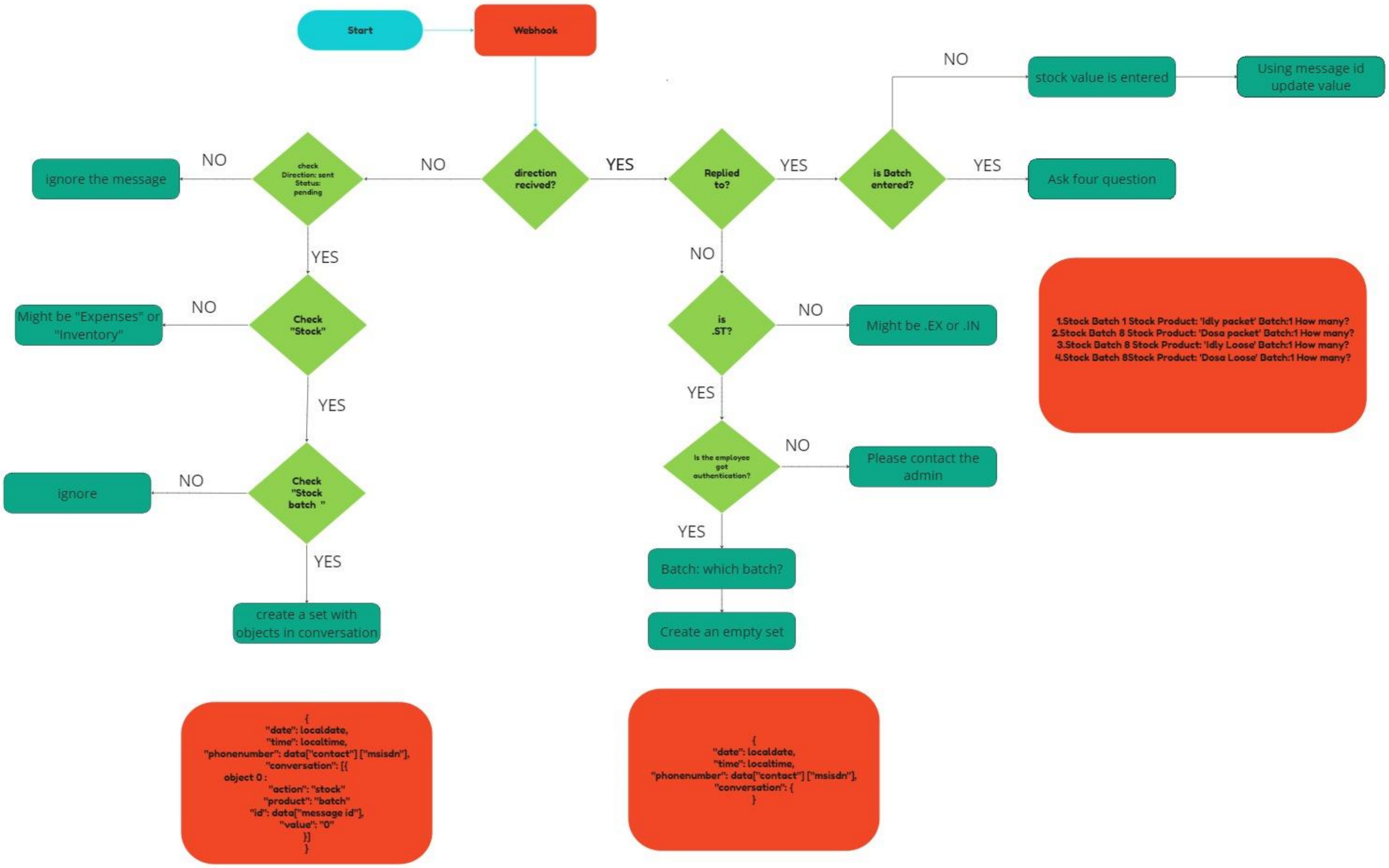
DATE	WORK DONE
MAR 11	discussing about the work flow
MAR 12	creating user flow in utility
MAR 13- 15	designing of workflow in MongoDB and modules



```
∨ modules
  > __pycache__
    __init__.py
    actionschema.py
    employschema.py
    mappingschema.py
> service
> soochiwebhook
∨ utility
  > __pycache__
    __init__.py
    conversation.py
    message.py
    mongoDB.py        M
> webhook
```

3.11 Week

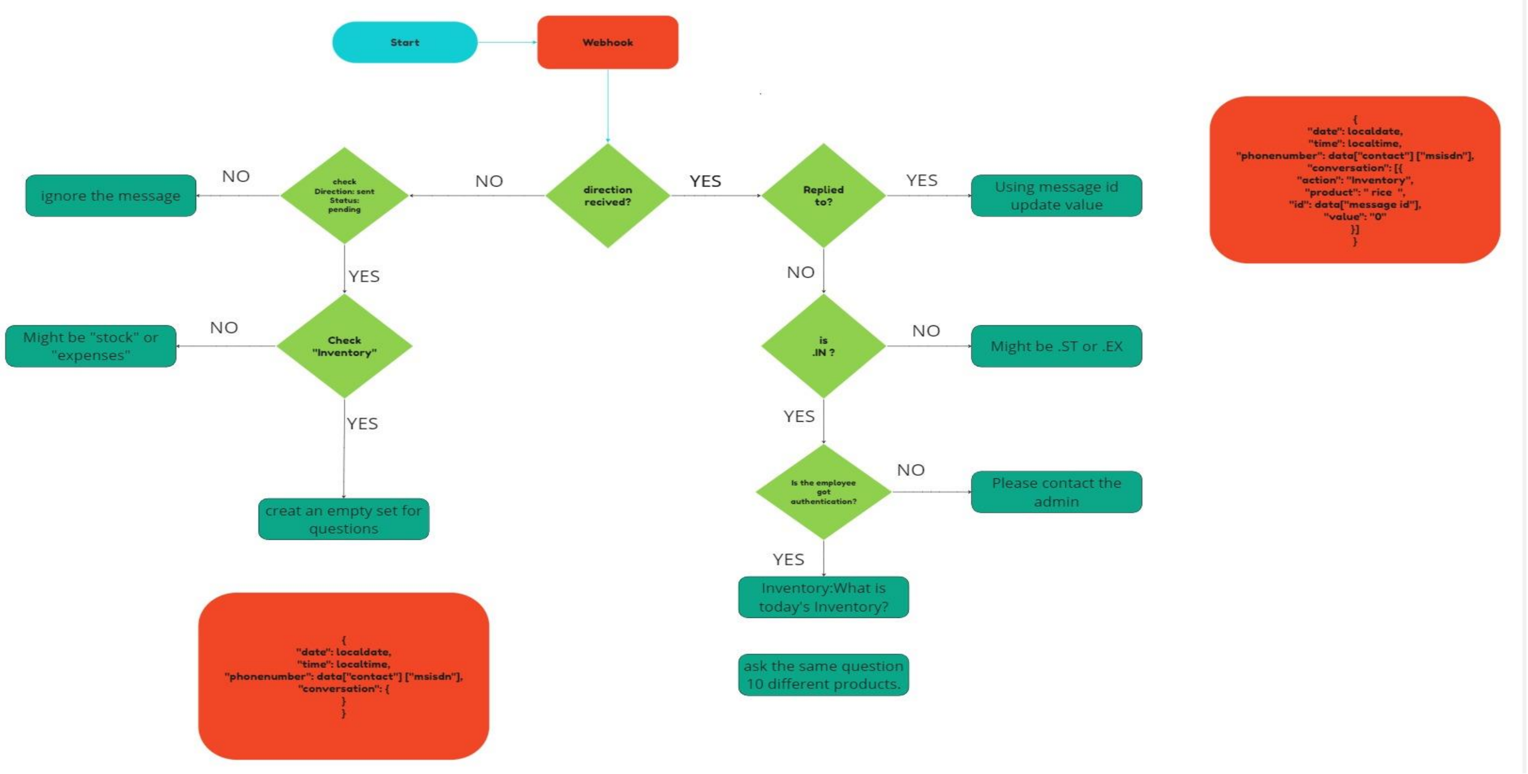
DATE	**WORK DONE on FLOW CHART**
MAR 18-19	Discussion for layouts for flowcharts
MAR 20	Stock flowchart
MAR21	Inventory flowchart
MAR 22	Expenses flowchart



1.Stock Batch 1 Stock Product: 'Idly packet' Batch:1 How many?
2.Stock Batch 8 Stock Product: 'Dosa packet' Batch:1 How many?
3.Stock Batch 8 Stock Product: 'Idly Loose' Batch:1 How many?
4.Stock Batch 8Stock Product: 'Dosa Loose' Batch:1 How many?

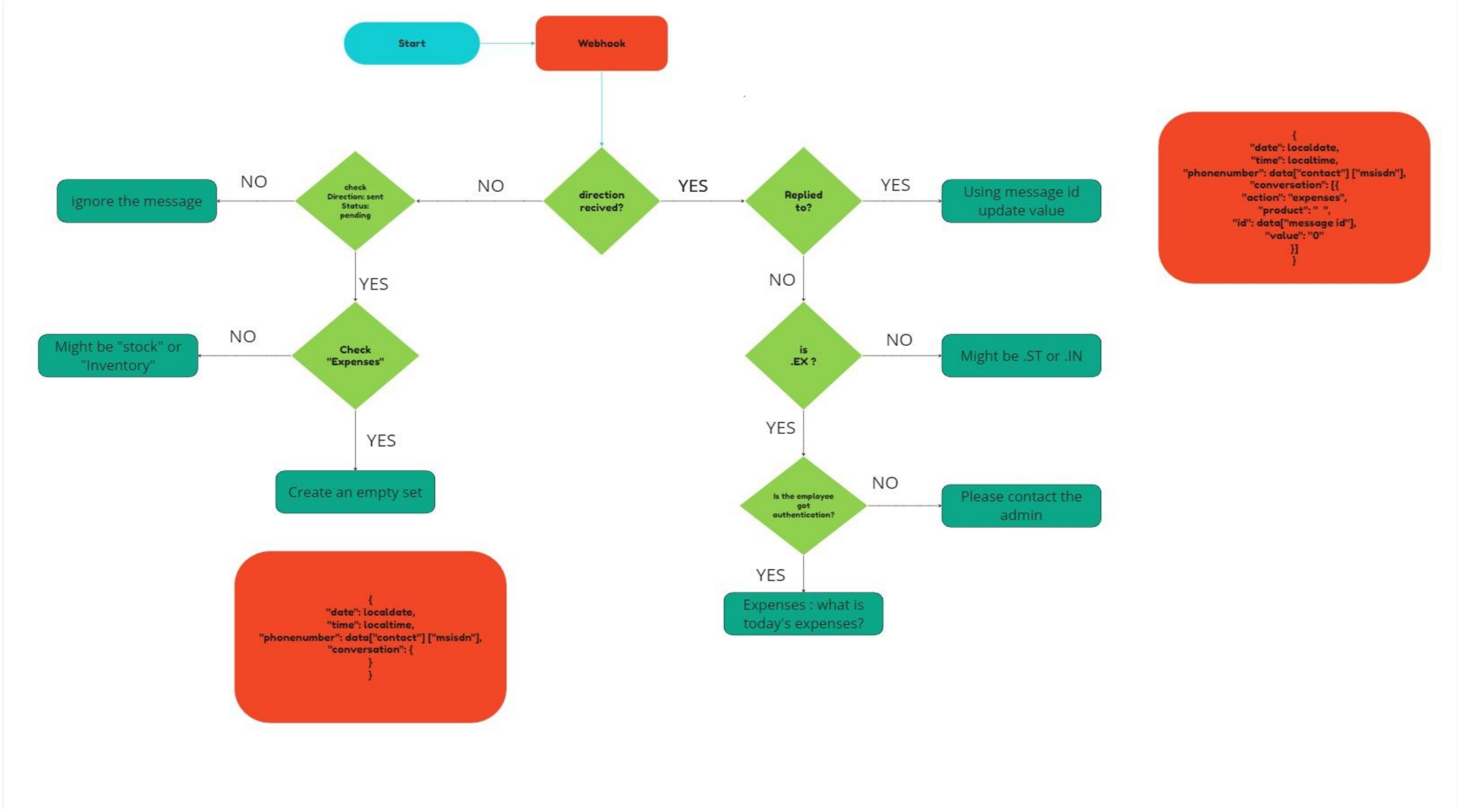
```
{
"date": localdate,
"time": localtime,
"phonenumber": data["contact"] ["msisdn"],
"conversation": [{
object 0 :
"action": "stock"
"product": "batch"
"id": data["message id"],
"value": "0"
}]
}
```

```
{
"date": localdate,
"time": localtime,
"phonenumber": data["contact"] ["msisdn"],
"conversation": {
}
```


Start
Webhook
direction recived?
NO
YES
check Direction: sent Status: pending
NO
ignore the message
YES
Check "Inventory"
NO
Might be "stock" or "expenses"
YES
creat an empty set for questions
Replied to?
YES
Using message id update value
NO
is .IN ?
NO
Might be .ST or .EX
YES
Is the employee got authentication?
NO
Please contact the admin
YES
Inventory:What is today's Inventory?
ask the same question 10 different products.

```
{
"date": localdate,
"time": localtime,
"phonenumber": data["contact"] ["msisdn"],
"conversation": [{
"action": "Inventory",
"product": " rice  ",
"id": data["message id"],
"value": "0"
}]
}
```

```
{
"date": localdate,
"time": localtime,
"phonenumber": data["contact"] ["msisdn"],
"conversation": {
}
}
```



```
{
"date": localdate,
"time": localtime,
"phonenumber": data["contact"] ["msisdn"],
"conversation": [{
"action": "expenses",
"product": "  ",
"id": data["message id"],
"value": "0"
}]
}
```

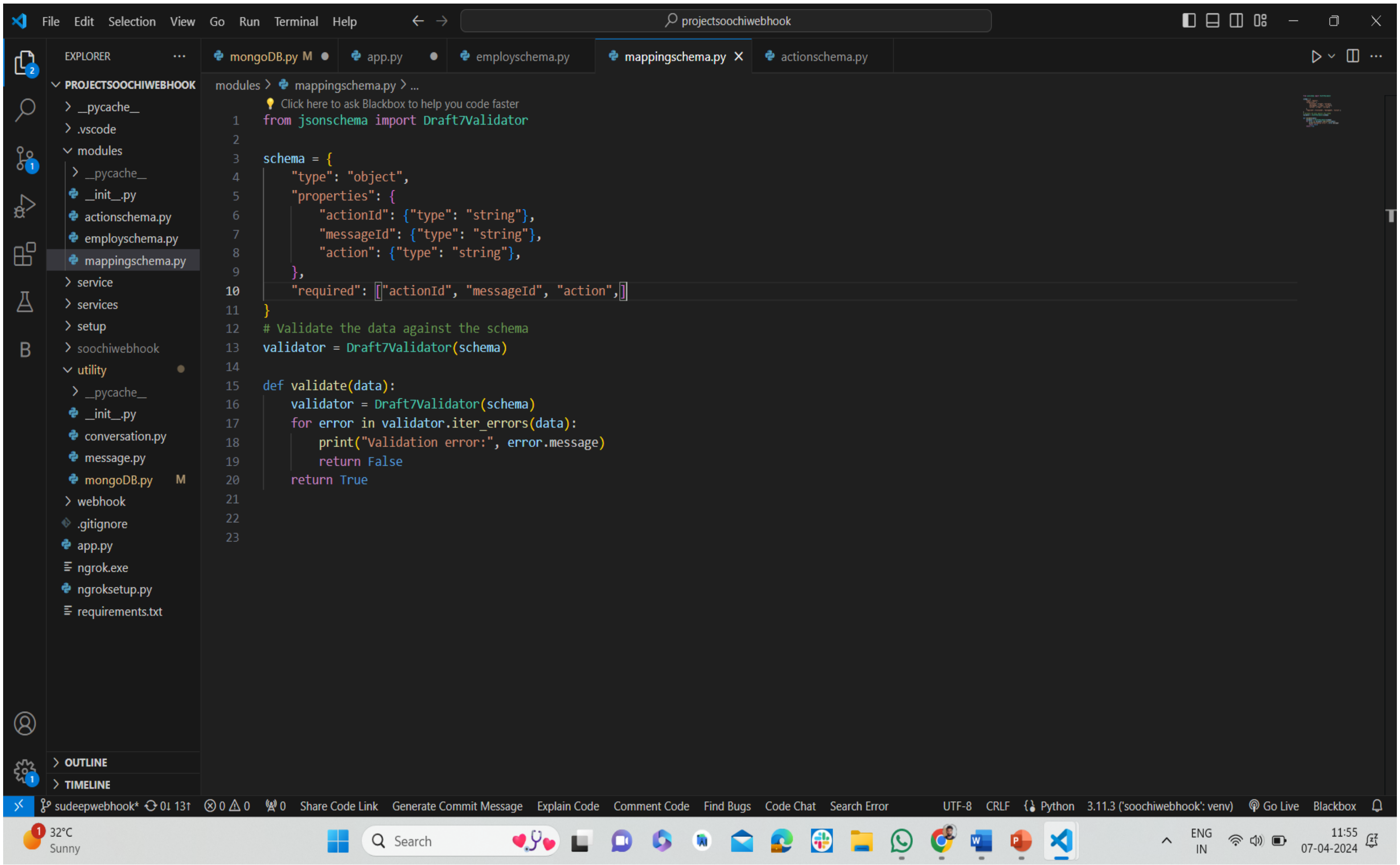
```
{
"date": localdate,
"time": localtime,
"phonenumber": data["contact"] ["msisdn"],
"conversation": {
}
}
```

3.12 Week

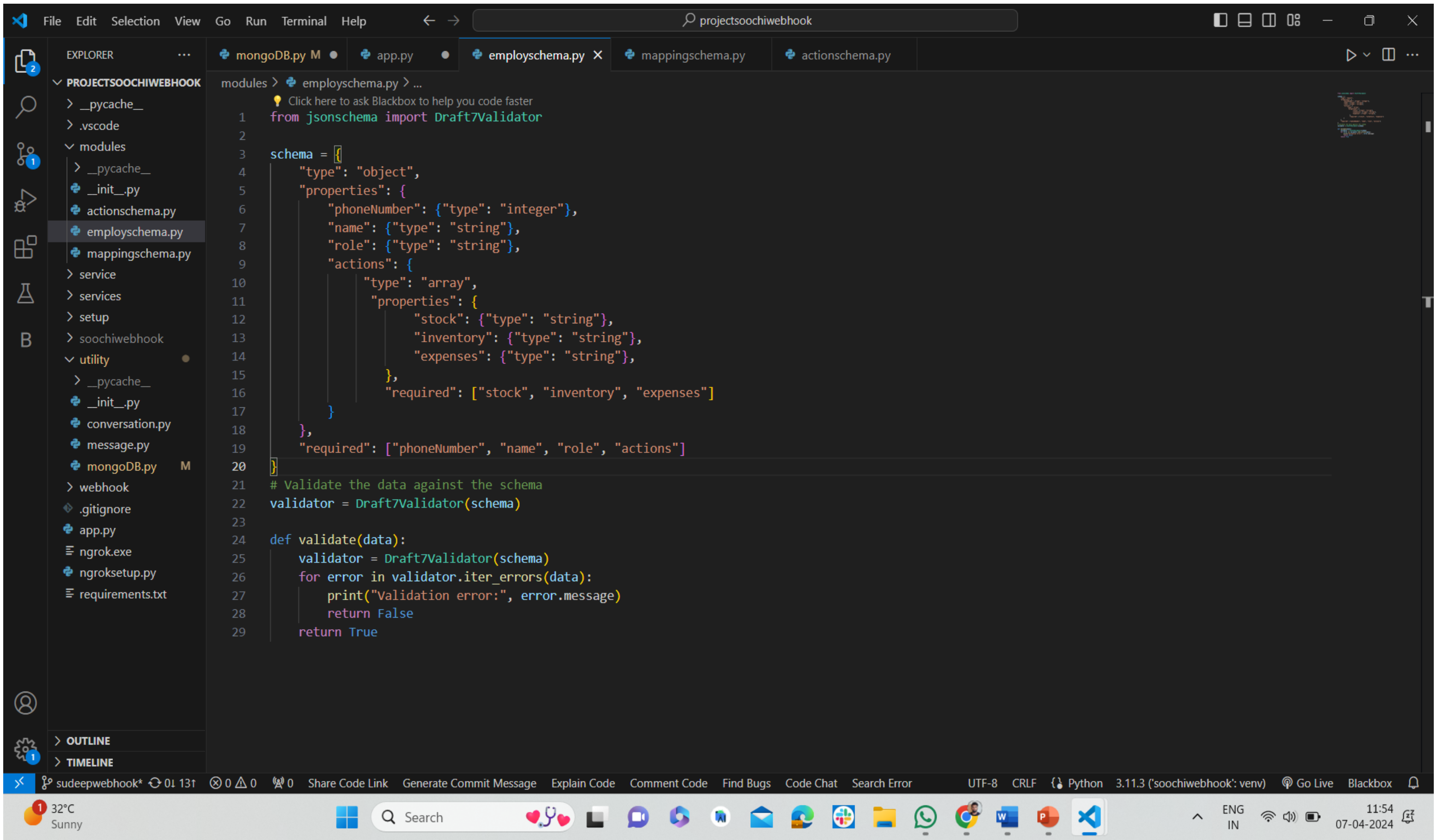
DATE	**WORK DONE on coding**
MAR 25-29	Holiday for Holi festival – work from home

3.12 Week

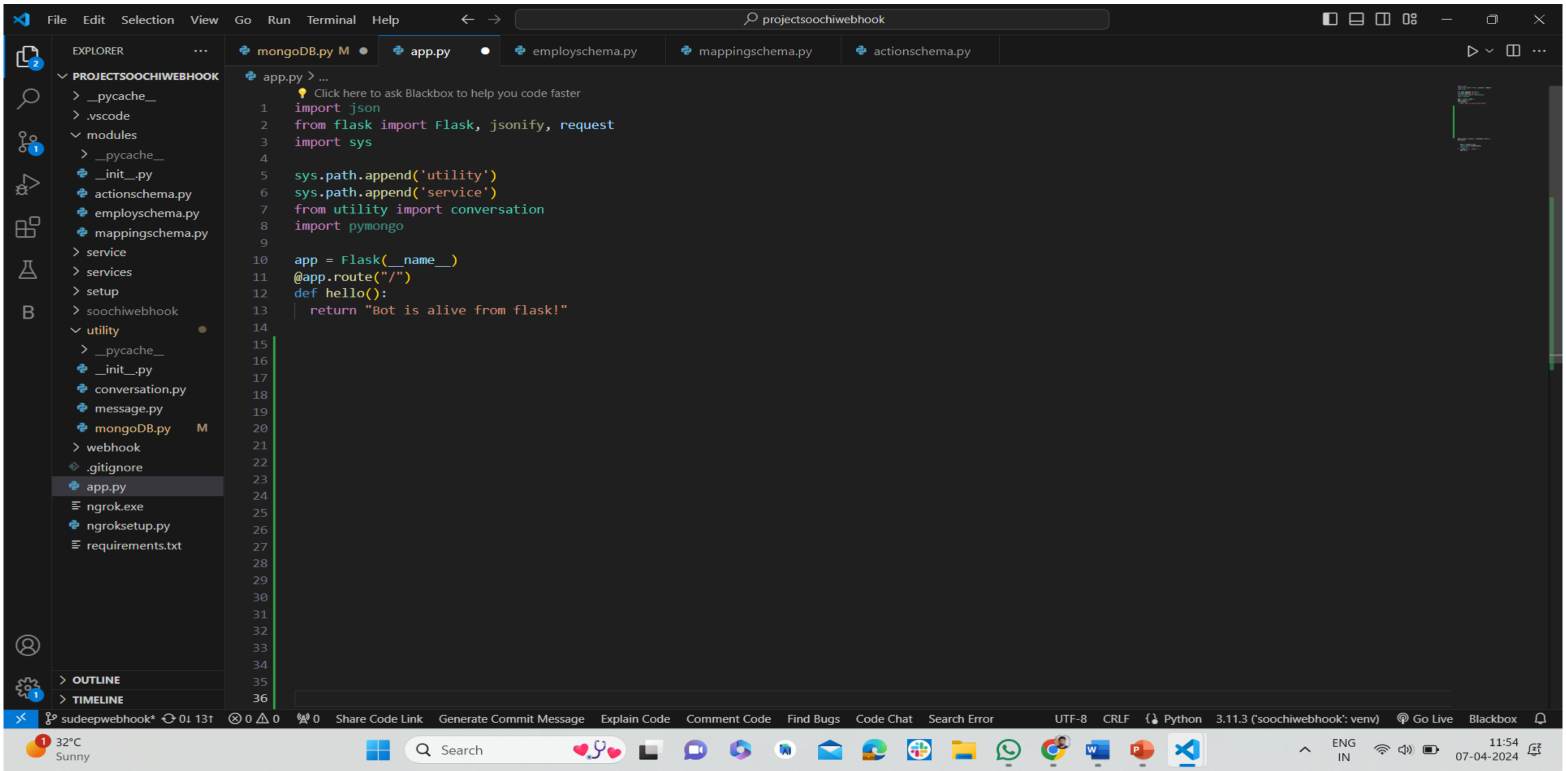
DATE	**WORK DONE on coding**
APRIL 1	Working on app.py to connect all other folder
APRIL 2-3	Modules: actionschema.py and mappingschema.py To structure the data
APRIL 4-5	Utility: MongoDB.py Import files from folder, connect the application and create a collection



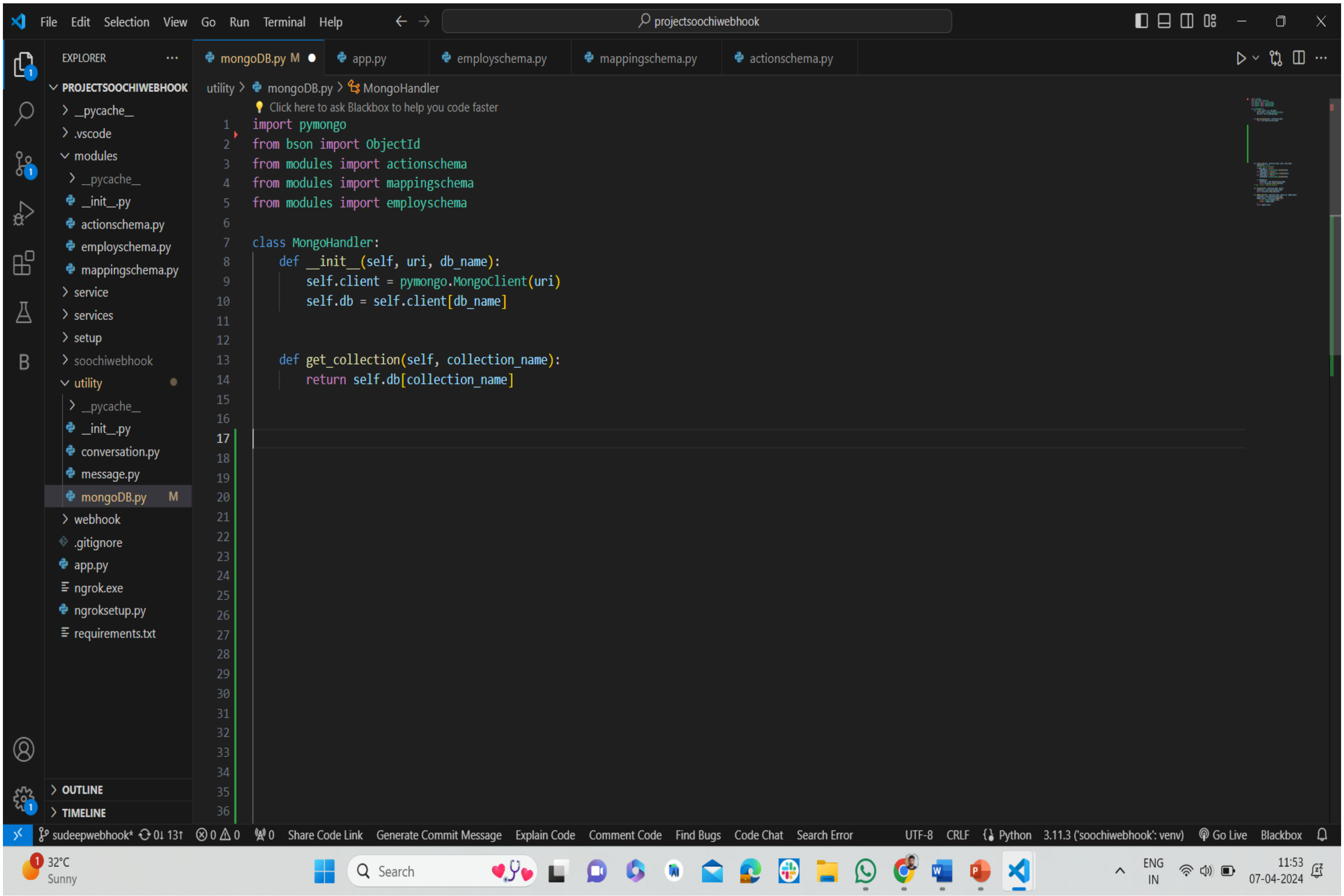
```python
from jsonschema import Draft7Validator

schema = {
    "type": "object",
    "properties": {
        "actionId": {"type": "string"},
        "messageId": {"type": "string"},
        "action": {"type": "string"},
    },
    "required": ["actionId", "messageId", "action",]
}
# Validate the data against the schema
validator = Draft7Validator(schema)

def validate(data):
    validator = Draft7Validator(schema)
    for error in validator.iter_errors(data):
        print("Validation error:", error.message)
        return False
    return True
```



```python
from jsonschema import Draft7Validator

schema = {
    "type": "object",
    "properties": {
        "phoneNumber": {"type": "integer"},
        "name": {"type": "string"},
        "role": {"type": "string"},
        "actions": {
             "type": "array",
              "properties": {
                    "stock": {"type": "string"},
                    "inventory": {"type": "string"},
                    "expenses": {"type": "string"},
                },
                "required": ["stock", "inventory", "expenses"]
        }
    },
    "required": ["phoneNumber", "name", "role", "actions"]
}
# Validate the data against the schema
validator = Draft7Validator(schema)

def validate(data):
    validator = Draft7Validator(schema)
    for error in validator.iter_errors(data):
        print("Validation error:", error.message)
        return False
    return True
```



```python
import json
from flask import Flask, jsonify, request
import sys

sys.path.append('utility')
sys.path.append('service')
from utility import conversation
import pymongo

app = Flask(__name__)
@app.route("/")
def hello():
  return "Bot is alive from flask!"
```



```python
import pymongo
from bson import ObjectId
from modules import actionschema
from modules import mappingschema
from modules import employschema

class MongoHandler:
    def __init__(self, uri, db_name):
        self.client = pymongo.MongoClient(uri)
        self.db = self.client[db_name]


    def get_collection(self, collection_name):
        return self.db[collection_name]
```

4. TECHNICAL CONTENT:

4.1 Trainings

4.1.1 Pre-requisite trainings

These trainings are given to understand company standards and employee's safety.

4.1.2 These trainings include:

1. HTML - Hyper Text Markup Language
2. CSS - cascading style sheets
3. JS – JavaScript
4. GITHUB
5. PYTHON
6. MONGO DB
7. TRELLO
8. FLASK
9. NG ROCK
10. Webhook
11. Message bird

In the web development industry, interns are typically trained in a variety of essential skills and technologies to equip them for success in the field. This training often encompasses fundamental programming languages such as HTML, CSS, and JavaScript, which form the backbone of web development. Interns also learn about frameworks and libraries like React, Angular, or Vue.js, which streamline the development process and enhance functionality. Additionally, they receive instruction in version control systems such as Git, enabling them to collaborate effectively with team members and manage project code efficiently. Moreover, interns are introduced to concepts like responsive design, accessibility, and user experience principles to ensure they can create websites and applications that are both visually appealing and user-friendly. Throughout their training, emphasis is placed on problem-solving, critical thinking, and the ability to adapt to evolving technologies, preparing interns to contribute meaningfully to web development projects in a professional setting.

4.2Technical Trainings

4.2.1 Web Development: Web development involves creating websites and web applications for the internet or an intranet. It encompasses various aspects such as web design, web content development, client-side/server-side scripting, and network security configuration. Web developers use programming languages, frameworks, and databases to build interactive and functional websites tailored to user needs.

4.2.2 HTML (Hypertext Markup Language): HTML is the standard markup language used to create web pages. It provides the structure and content of a webpage by using tags to define elements such as headings, paragraphs, links, images, and multimedia.

4.2.3 CSS (Cascading Style Sheets): CSS is a stylesheet language used to control the presentation and layout of HTML documents. It enables web developers to customize the appearance of web pages by defining styles for fonts, colors, spacing, and positioning of elements.

4.2.4 JavaScript (JS): JavaScript is a high-level programming language commonly used for client-side scripting in web development. It adds interactivity and dynamic behavior to web pages, allowing developers to create features like animations, form validation, and interactive maps.

4.2.5 Python: Python is a versatile and beginner-friendly programming language often used in web development for both backend and frontend tasks. It offers simplicity, readability, and a vast ecosystem of libraries and frameworks. Python frameworks like Django and Flask are popular choices for building web applications.

4.2.6 MongoDB: MongoDB is a NoSQL database used for storing and managing data in web development. It offers a flexible document-based data model, scalability, and high performance, making it suitable for handling large volumes of data in modern web applications.

4.2.7 Flask: Flask is a lightweight and flexible web framework for Python. It provides tools and libraries for building web applications quickly and efficiently. Flask is known for its simplicity and extensibility, making it a popular choice for developing prototypes, APIs, and small to medium-sized web applications.

4.2.8 NGINX (pronounced "engine-x"): NGINX is a high-performance web server and reverse proxy server used to serve static and dynamic content on the web. It is known for its speed, scalability, and reliability, making it a popular choice for powering modern web applications.

4.2.9 Webhook: Webhooks are a way for web applications to communicate with each other in real-time. They enable automatic notifications and data exchange between different systems or services, allowing for seamless integration and event-driven workflows in web development.

4.2.10 Message Bird: Message Bird is a communication platform that provides APIs and tools for integrating SMS, voice, and chat functionality into web and mobile applications. It allows developers to send and receive messages, make phone calls, and manage communication channels programmatically, enabling businesses to engage with their customers effectively.

5. project details:

5.1 Introduction:

Welcome to our professional WhatsApp chatbot designed to streamline information gathering and management tasks for the business. Our chatbot is tailored to collect essential data such as product stock, inventory details, daily expenses, and employee information. Additionally, it intelligently captures dates from employee responses for analysis, routine surveys, and facilitating company growth.

5.2 Features:

5.2.1 Product Stock Management: Our chatbot allows you to effortlessly inquire about the stock availability of various products within your inventory. Receive real-time updates on product availability and streamline your restocking processes.

5.2.2 Inventory Tracking: Gain insights into your inventory levels through our chatbot. Easily monitor stock levels, track usage, and make informed decisions regarding inventory management.

5.2.3 Daily Expenses Monitoring: Keep track of your daily expenses seamlessly using our chatbot. Input and retrieve expense data efficiently, enabling you to maintain accurate financial records and make informed budgeting decisions.

5.2.4 Employee Information Management: Our chatbot facilitates the storage and retrieval of essential employee information. From contact details to roles and responsibilities, access employee data swiftly and securely.

5.2.5 Date Analysis and Survey Integration: Utilize the date information extracted from employee responses for analysis, scheduling, and conducting routine surveys. Gain valuable insights into employee engagement, productivity trends, and more.

5.3 Benefits:

5.3.1 Improved Efficiency: Streamline data collection and management tasks, saving time and resources for your business operations.

5.3.2 Enhanced Accuracy: Minimize human error in data entry and retrieval processes, ensuring the integrity of your business information.

5.3.3 Data-Driven Insights: Leverage collected data for analysis, enabling informed decision-making and strategic planning.

5.3.4 Seamless Integration: Integrate our chatbot seamlessly into your existing communication channels, enhancing accessibility for employees and stakeholders.

5.4 files and folder setup:

- **APP.PY:** This file serves as the central hub for establishing connections to other essential components such as services, utilities, and modules. Its purpose is to initialize the overall process flow within our project. By consolidating these connections in one file, we streamline the initiation process, ensuring efficient interaction between different functionalities. This architectural approach enhances code manageability, facilitates easier debugging, and promotes scalability as our project evolves.

- **SERVICES:** This folder houses distinct files for handling various actions, such as stock management, inventory tracking, and daily expenses. Each file is dedicated to a specific task, allowing us to customize responses based on the nature of the message received. This modular approach enables precise and tailored interactions with the user, enhancing the clarity and effectiveness of our chatbot's responses. By organizing functionalities into separate files, we promote code clarity, maintainability, and scalability, facilitating seamless integration of new features as needed.

- **UTILITY:** This folder comprises three distinct files: conversation.py, message.py, and mongoDB.py, each serving a unique role within our system.

- message.py is responsible for generating responses tailored to specific actions such as stock, inventory, and daily expenses, ensuring accurate and relevant communication with users.

- conversation.py facilitates the retrieval of essential IDs from WhatsApp and orchestrates the collection process, ensuring seamless interaction between our chatbot and users.

- mongoDB.py plays a crucial role in establishing connectivity with the MongoDB application, enabling the storage of entire conversations from WhatsApp. It manages the storage process by utilizing conversation IDs and other relevant identifiers.

- By dividing functionalities into separate files, we enhance code organization, maintainability, and scalability, fostering a robust and efficient system architecture.

- **MODULES:**

- This folder encompasses three essential files: actionschema.py, employschema.py, and mappingschema.py, each contributing to the construction of structured data within our MongoDB application.

- **actionschema.py:** This file is dedicated to defining the appropriate data structure for various actions such as stock, inventory, and daily expenses. It ensures consistency and organization in storing relevant information related to these actions.

- **employschema.py:** Here, we establish the structure for organizing employee information. This schema allows for systematic storage and retrieval of essential employee data, including contact details, roles, and responsibilities.

- **mappingschema.py:** This file facilitates the creation of a structured format for mapping conversations to dates using conversation IDs. It enables efficient retrieval and analysis of conversation data based on specific dates, enhancing the functionality of our chatbot.

- By employing these schema files, we maintain data integrity, enhance query efficiency, and promote scalability within our MongoDB application. This structured approach ensures clarity, consistency, and optimal performance in managing diverse data sets.

5.5 Flowchart:

Stock flowchart:

1. When the employee opens the WhatsApp chat, the webhook gets activated.

2. If the message direction is received, the system checks if it has been replied to.

- If the message is replied to:
- If the batch is entered, the system asks four questions:
    1. **"Stock Batch 1 Stock Product: 'Idly packet' Batch:1 How many?"**
    2. **"Stock Batch 1 Stock Product: 'Dosa packet' Batch:1 How many?"**
    3. **"Stock Batch 1 Stock Product: 'Idly Loose' Batch:1 How many?"**
    4. **"Stock Batch 1 Stock Product: 'Dosa Loose' Batch:1 How many?"**
- If the batch is not entered, the system retrieves the stock value and updates it using the message ID.
    - If the message is not replied to:
    - Check if it is **".st".** If not, it might be **".ex"** or **".in".**
    - If it is **".st"**, check if the employee is authorized. If not, send a message:

      "Please contact the admin."
- If authorized, ask for the batch: **"Batch: which batch?"** and create an empty set:

```
 “
     {
         "date": local date,
         "time": local time,
         " Phone number": data["contact"] ["msisdn"],
          "conversation": {}
     }
        ”
```

3. If the message direction is not received, check if the direction is sent and the status is pending.

- If not, ignore the message.
- If yes, check if the message contains the word **"stock"**.
- If not, it might be **"expenses"** or "**inventory"**.
- If yes, check if the message contains **"stock batch"**.
- If not, ignore it.
- If yes, create a set with objects in conversation:

“

```
        {
           "date": local date,
           "time": local time,
           "phone number": data["contact"]["msisdn"],
           "conversation": [{
                              "action": "stock",
                              "product": "batch",
                              "id": data ["message id"],
                              "value": "0"
                            }]
        }
           ”
```

Inventory flow chart:

1. When the employee opens the WhatsApp chat, the webhook gets activated.

2. If the message direction is received:

- Check if it is replied to.
- If replied to, update the value using the message ID:

```
“
    {
            "date": local date,
           "time": local time,
          "phone number": data["contact"]["msisdn"],
          "conversation": [{
                              "action": "Inventory",
                              "product": "rice",
                             "id": data["message id"],
                            "value": "0"
                        }]
  }
    “
```

- If not replied to, check if it is **".in".**
- If not, it might be **".ex"** or **".st".**
- If it is **".in"**, check if the employee is authorized.
- If not authorized, send a message: **"Please contact the admin."**
- If authorized, ask for the inventory: **"Inventory: What is today's inventory?"** and ask the same question for 10 different products.

3. If the message direction is not received:

- Check if the direction is sent and the status is pending.
- If not, ignore the message.

- If yes, check if the message contains the word **"inventory".**
- If not, it might be **"expenses"** or **"stock".**
- If yes, create an empty set for questions:

```
“
     {
          "date": local date,
         "time": local time,
         "phone number": data["contact"]["msisdn"],
        "conversation": {
                               }
    }
           “
```

Expenses flow chart:

1. When the employee opens the WhatsApp chat, the webhook gets activated.
2. If the message direction is received:
    - Check whether it is replied to.
    - If replied to, update the value using the message ID:

```
“
    {
        "date": local date,
        "time": local time,
       "phone number": data["contact"]["msisdn"],
      "conversation": [{
                          "action": "expenses",
                          "product": " ",
                          "id": data["message id"],
                          "value": "0"
                        }]
     }
        ”
```

    - If not replied to, check if it is **".ex".**
    - If not, it might be **".in"** or **".st".**
    - If it is **".ex",** check whether the employee is authorized.
    - If not authorized, send a message: **"Please contact the admin."**
    - If authorized, ask about the expenses: **"Expenses: What is today's expenses?"**

3. If the message direction is not received:
    - Check whether the direction is sent and the status is pending.
    - If not, ignore the message.

- If yes, check whether the message contains the word **"expenses".**
- If not, it might be **"inventory"** or **"stock".**
- If yes, create an empty set for questions:

```
“
  {
    "date": local date,
    "time": local time,
    "phone number": data["contact"]["msisdn"],
    "conversation": {
                     }
  }
      ”
```