Data Foundation Cloud (DCloud) Environment

AI Model Training/ Execution

Team Name: SVM

Team Members:

Shlok Bansal : 2021201046

Vishal Pandey : 2021201070

Mayank Mukundam : 2021201057

PROJECT OVERVIEW

1. Docker nodes with defined dataset and constrained resource access (GPU/CPU, storage and memory)
2. Python and NVIDIA based AI/ ML setup for TensorFlow and PyTorch

SOFTWARE REQUIREMENTS

INTRODUCTION

The system provides an efficient solution for building a containerised ecosystem. Users can easily configure these ecosystems according to their requirements but within constraints of limited CPUs, GPUs, memory. User will be provided with a friendly interface to manage all sorts of configurations. This streamlined process reduces the time and effort required to manage the ecosystem.

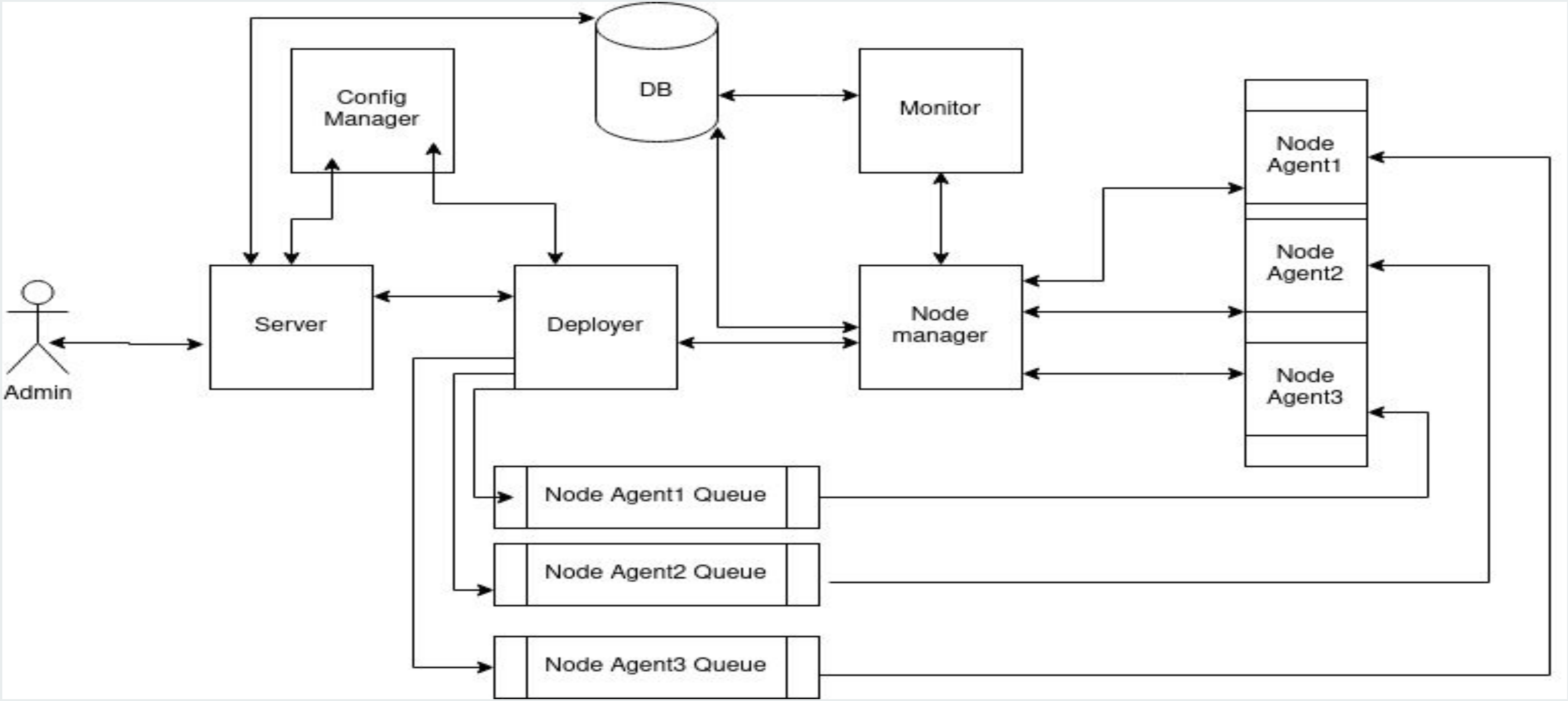
SYSTEM FEATURES

- Configurations management eg no of gpus, ram, automatic software installation support
- Docker container management according to requirements
- Container monitoring
- Real Time Status Updates
- handle concurrent requests
- Support for configuration templates

NON-FUNCTIONAL REQUIREMENTS

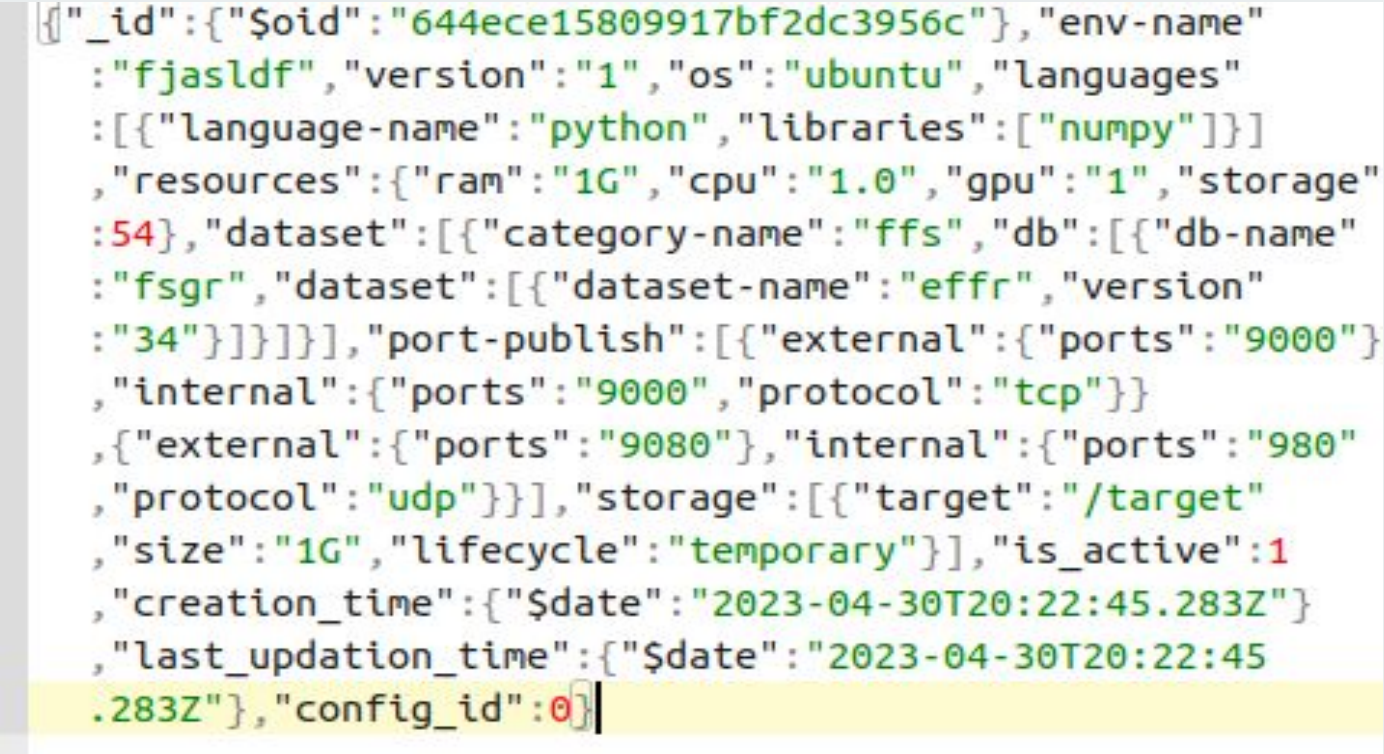
- Performance: The system should be able to handle high and perform efficiently, with quick response time.
- High Scalability, High Availability
- Interoperability: The system should be able to integrate with DFS platform
- User-friendly: The system should be easy to use, with a user-friendly interface and intuitive functionality.
- Reliability: The system should be reliable, with minimal downtime and a high degree of stability.
- Compatibility: The system should be compatible with various data formats, allowing users to upload their own ecosystem config files.
- Maintainability: The system should be easy to maintain and upgrade, with clear documentation and a robust architecture

ARCHITECTURE OVERVIEW



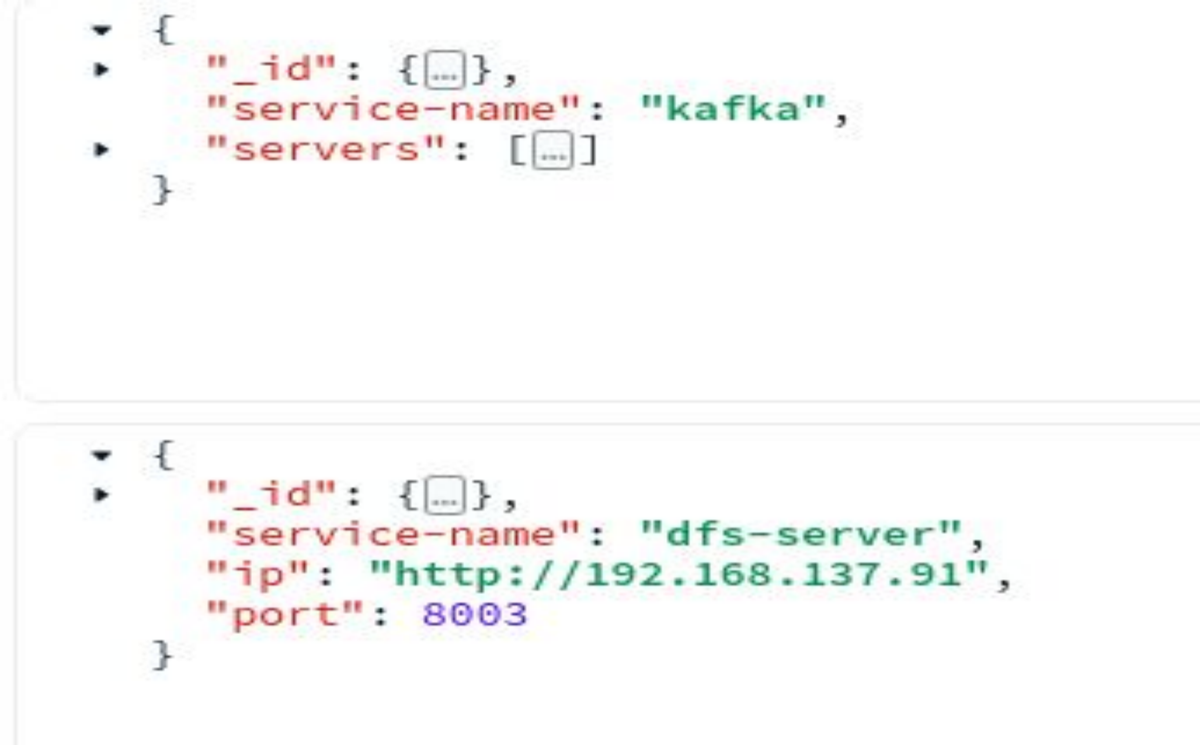
CONFIGURATION CONTRACTS

ENVIRONMENT CONFIGURATION



```
{"_id":{"$oid":"644ece15809917bf2dc3956c"},"env-name"
  :"fjasldf","version":"1","os":"ubuntu","languages"
  :[{"language-name":"python","libraries":["numpy"]}]
  ,"resources":{"ram":"1G","cpu":"1.0","gpu":"1","storage"
  :54},"dataset":[{"category-name":"ffs","db":[{"db-name"
  :"fsgr","dataset":[{"dataset-name":"effr","version"
  :"34"}]}]}],"port-publish":[{"external":{"ports":"9000"}
  ,"internal":{"ports":"9000","protocol":"tcp"}}
  ,{"external":{"ports":"9080"},"internal":{"ports":"980"
  ,"protocol":"udp"}}],"storage":[{"target":"/target"
  ,"size":"1G","lifecycle":"temporary"}],"is_active":1
  ,"creation_time":{"$date":"2023-04-30T20:22:45.283Z"}
  ,"last_updation_time":{"$date":"2023-04-30T20:22:45
  .283Z"},"config_id":0}
```

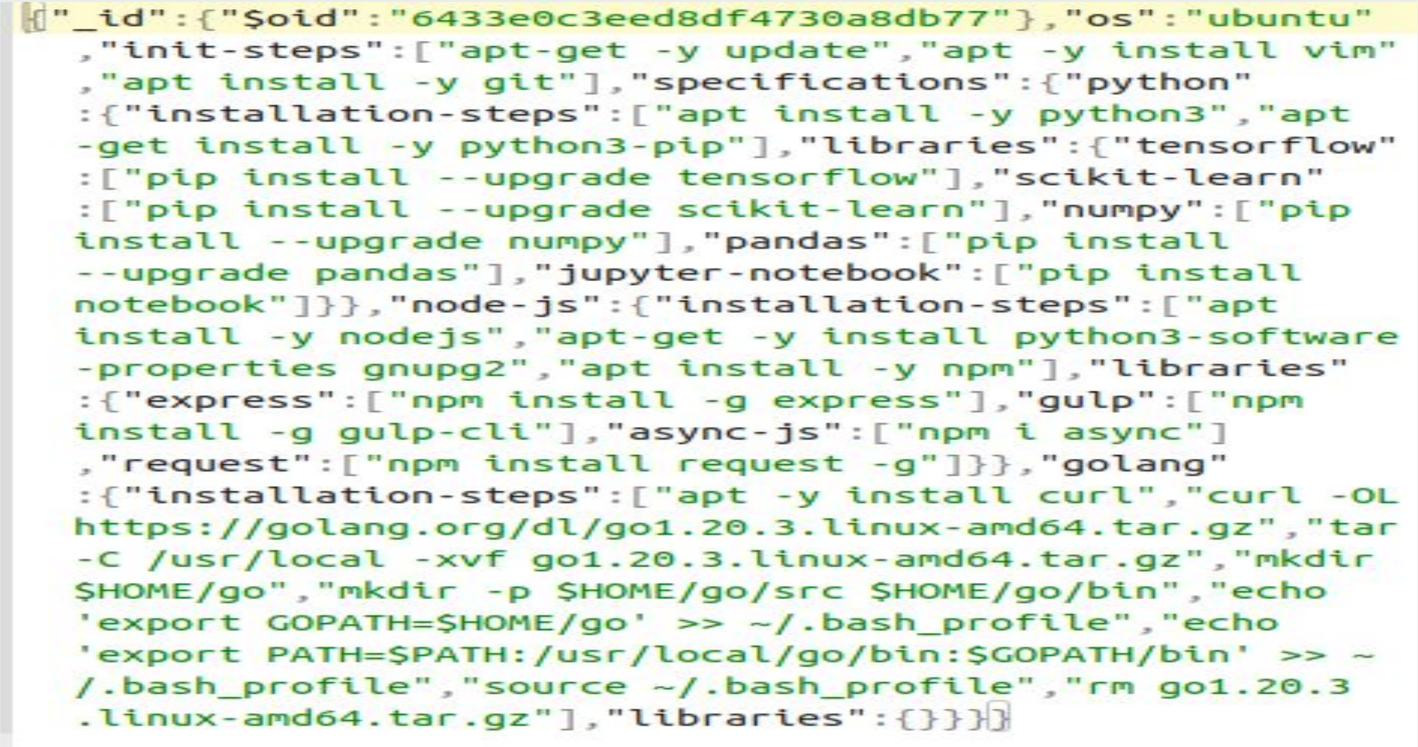
SERVICE SELF-REGISTER CONFIGURATION



```
{
  "_id": {...},
  "service-name": "kafka",
  "servers": [...]
}
```

```
{
  "_id": {...},
  "service-name": "dfs-server",
  "ip": "http://192.168.137.91",
  "port": 8003
}
```

LIBRARY CONTRACT

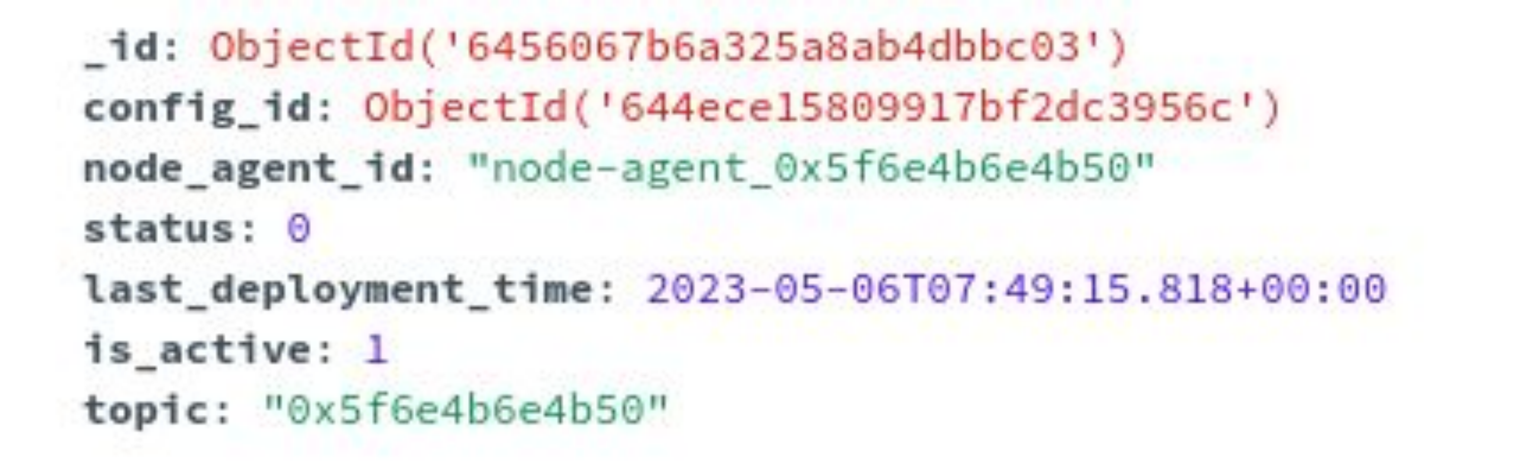


```
{"_id":{"$oid":"6433e0c3eed8df4730a8db77"},"os":"ubuntu"
 ,"init-steps":["apt-get -y update","apt -y install vim"
 ,"apt install -y git"],"specifications":{"python"
 :{"installation-steps":["apt install -y python3","apt
 -get install -y python3-pip"],"libraries":{"tensorflow"
 :["pip install --upgrade tensorflow"],"scikit-learn"
 :["pip install --upgrade scikit-learn"],"numpy":["pip
 install --upgrade numpy"],"pandas":["pip install
 --upgrade pandas"],"jupyter-notebook":["pip install
 notebook"]}},"node-js":{"installation-steps":["apt
 install -y nodejs","apt-get -y install python3-software
 -properties gnupg2","apt install -y npm"],"libraries"
 :{"express":["npm install -g express"],"gulp":["npm
 install -g gulp-cli"],"async-js":["npm i async"]
 ,"request":["npm install request -g"]}},"golang"
 :{"installation-steps":["apt -y install curl","curl -OL
 https://golang.org/dl/go1.20.3.linux-amd64.tar.gz","tar
 -C /usr/local -xvf go1.20.3.linux-amd64.tar.gz","mkdir
 $HOME/go","mkdir -p $HOME/go/src $HOME/go/bin","echo
 'export GOPATH=$HOME/go' >> ~/.bash_profile","echo
 'export PATH=$PATH:/usr/local/go/bin:$GOPATH/bin' >> ~
 /.bash_profile","source ~/.bash_profile","rm go1.20.3
 .linux-amd64.tar.gz"],"libraries":{}}}}
```

DEPLOYMENT PIPELINE

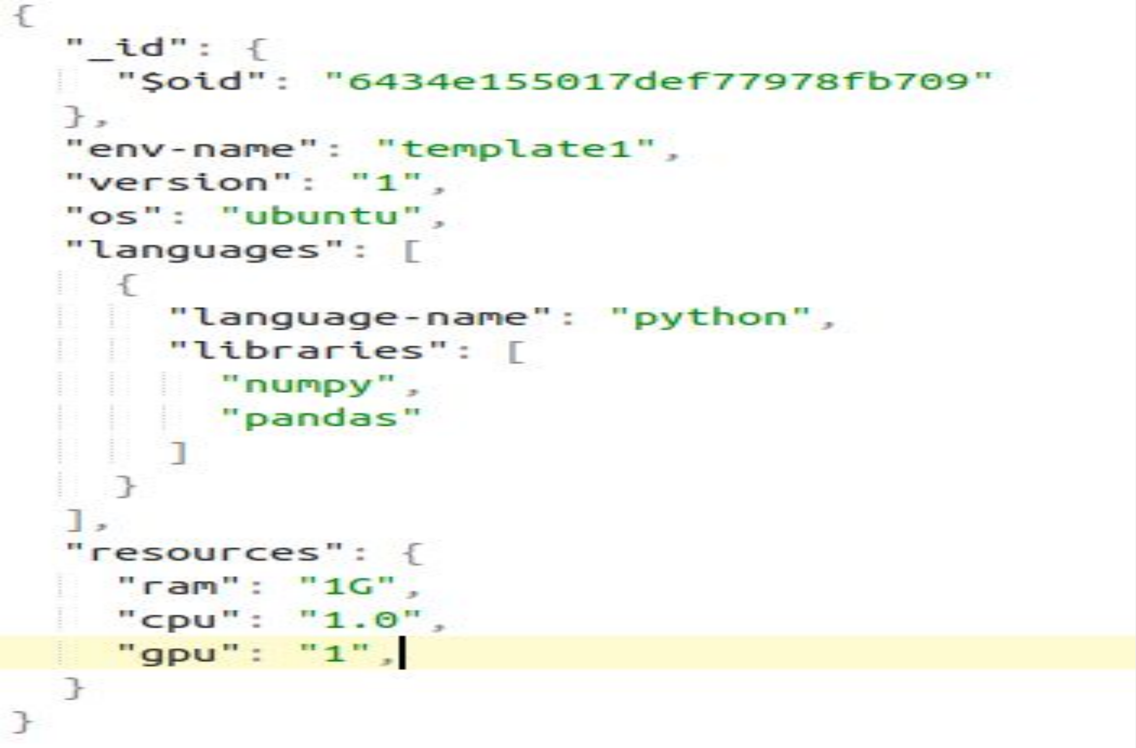
1) Request made through UI by selecting the configuration it wants to deploy.
2) It is received on the backend & sent for verification.
3) After validation requirement info is sent to the node manager.
4) Node Manager is a central entity that keeps records of the available resource info for all the nodes.
5) To keep these records on point Node manager communicates with node agent which lets node manager know the health of all their respective nodes & how much resources are currently under use on the node.
6) Then the most available node is selected that matches the current request specifications.
7) The request is then sent to the kafka queue of the respective node, topic of which is given by ‘mac address of the node’
8) The request is processed & the user can check the status of the deployment via the Status check button on UI.

DEPLOYMENT CONFIG



```
_id: ObjectId('6456067b6a325a8ab4dbbc03')
config_id: ObjectId('644ece15809917bf2dc3956c')
node_agent_id: "node-agent_0x5f6e4b6e4b50"
status: 0
last_deployment_time: 2023-05-06T07:49:15.818+00:00
is_active: 1
topic: "0x5f6e4b6e4b50"
```

TEMPLATES CONFIG



```json
{
  "_id": {
    "$oid": "6434e155017def77978fb709"
  },
  "env-name": "template1",
  "version": "1",
  "os": "ubuntu",
  "languages": [
    {
      "language-name": "python",
      "libraries": [
        "numpy",
        "pandas"
      ]
    }
  ],
  "resources": {
    "ram": "1G",
    "cpu": "1.0",
    "gpu": "1",
  }
}
```

PROJECT DELIVERABLE

The project will be delivered in phases:

- Prototype phase: Building a simple and efficient prototype to demonstrate the workflow for container ecosystem deployment

- Concurrency support: This phase will mostly focus on making the above prototype more rustic and handle traffic efficiently
- Logging and Fault Tolerance

- Final Backend: This phase will contain all above deliverables plus support for templates.

- UI Integration phase This phase deals with the Integration of whole backend with a UI Interface so that, it can be easily used by the end user