Unit-3

27/03/2024
Wednesday
Guided

Python for CC

Python for Cloud

Python for Amazon web services, Python for Google cloud platform, Python for windows Azure, Python for MapReduce, Python packages of Interest, Python web application frame work, Designing a RESTful web API

Python for Amazon Web Services:-

Boto is a Python package that provides interfaces to Amazon web services (AWS).

Boto supports the following services.

→ Compute
- Amazon Elastic Compute Cloud (EC2)
- Amazon Elastic Map Reduce (EMR)
- Auto Scaling

Content Delivery
- Amazon Cloud Front.

→ Database.
- Amazon Relational Data service
- Amazon DynamoDB
- Amazon Simple DB
- Amazon Elastic Cache
- Amazon Redshift.

→ Deployment and Management
- AWS. Elastic Beanstalk

AWS Cloud Formation
AWS Data Pipeline

→ Identity & Access
  AWS Identity and Access Management (IAM)

→ Application services
  Amazon cloud search
  Amazon Simple Workflow service (SWF)
  Amazon Simple Queue service (SQS)
  Amazon Simple Notification server (SNS)
  Amazon Simple email service (SES)

→ Monitoring
  Amazon cloudwatch

→ Networking
  Amazon Route 53
  Amazon Virtual Private Cloud (VPC)
  Elastic Load Balancing (ELB)
  Payments and Billing.
  Amazon Flexible Payment service (FPS)

→ Storage
  Amazon Simple Storage service (S3)
  Amazon Glacier
  Amazon Elastic Block Store (EBS)
  Google Cloud Storage.

Amazon EC2:

Amazon EC2, an IaaS provided by Amazon. EC2 delivers scalable, pay-as-you-go compute capacity in the cloud.
EC2 is a web service that provides computing capacity in the form of virtual machines that are launched in Amazon's cloud computing

environment.

Amazon Autoscaling:

Amazon auto scaling allows automatically scaling of Amazon EC2 capacity up (or down according to user defined conditions.

Amazon S3:

Amazon S3 is an online cloud-based data storage infrastructure for storing and retrieving any amount of data.

S3 provides highly reliable, scalable, fast, fully redundant and affordable storage infrastructure.

Amazon RDS:

Amazon RDS is a web service that allows you to create instances of MySQL, Oracle (or MS SQL server in the cloud.

With RDS, developers can easily set up, operate, and scale a relational database in the cloud.

Amazon Dynamo DB:

It is a fully-managed, scalable, high performance No-SQL database service.

Amazon SQS:

It offers a highly scalable and reliable hosted queue for storing messages as they travel b/w distinct components of applications.

Amazon EMR:

It is a web service that utilizes Hadoop framework running on Amazon EC2 and Amazon S3. EMR is suitable for massive scale data processing for applications such as data mining, data warehousing, scientific simulations etc.

Python for Google Cloud Platform:-

Use the Google APIs Python client library for accessing Google's APIs.

Google Compute Engine:

It provides scalable and flexible virtual machine computing capabilities in the cloud.

Google Cloud Storage:

It is a cloud service, for storing data in the Google's cloud. Data stored on Google cloud storage is organized into buckets.

It uses OAuth and credentials in the credentials file to request a refresh and access token, which is then stored in the oauth2.dat file.

After completing the OAuth authorization, an instance of the Google cloud storage service is obtained.

Google Cloud SQL:

Google cloud SQL is a MySQL DB in the Google cloud.

Google BigQuery:

It allows querying massive datasets with SQL like queries.

The BigQuery queries are run against append-only tables and use the processing power of Google's infrastructure for speeding up queries.

Google Cloud Datastore:

It is a No-SQL (i.e., schema-less) object datastore that provides robust and scalable storage.

Data objects in the datastore are known as entities. An entity has one (or more named properties, each of which can have one (or more values.

~~Entities has one (or more named properties, each of which can have one (or more values.~~

Entities of the same kind need not have the same properties, and an entity's values for a given property need not all be of the same data type.

Google App Engine (GAE):

GAE is a web app^n hosting service. App^{n}s hosted in GAE are easy to build, maintain, and scale.

Applications run in a secure sandbox environment that provides limited access to the underlying OS.

The sandbox isolates an app^n in its own secure, reliable environment that is independent of the h/w, OS and physical location of the web server.

The benefit of the sandbox is that it allows GAE to distribute web requests for the app^n across multiple servers, and start and stop servers to meet app^n workloads.

GAE includes a simple web app^n framework called webapp2. App Engine also supports any framework written in pure Python that speaks WSGI, including Django, CherryPy, Pylons, web.py, and web2.py.

Every AppEngine app^n requires a configuration file i.e, app.yaml.

App Engine SDK comes with a dev web server that allows you to test the appn locally.

Python for Windows Azure:

Windows Azure provides 3 compute models that you can use to host web appns web sites, cloud services and virtual machines.

Azure cloud service:

An appn that is run in Windows Azure is called a Windows Azure cloud service that includes the appn code and configuration.

To deploy an appn in windows azure as a cloud service, 3 components are needed — server def file, service configuration file and service package.

Azure Virtual Machines:

Windows Azure Virtual Machines allows you to provision on-demand, scalable compute infrastructure

To create a virtual machine, a cloud service is first created. Virtual machine is created using the create_virtual_machine_deployment method of the Azure service management API.

Azure Storage:

Windows Azure Storage services allow you to store and access various forms of data (Blobs, Tables and Queries).

Blobs are used to store unstructured binary and text data.

Tables are used to store non-relational structured data.

Queries are used to store messages that a client can access.

Azure Blob service:

Azure Blob service allows you to store large amounts of unstructured text or binary data such as video, audio and images.

Azure Table service:

Azure Table service provides No-SQL capabilities for apps that require storage of large amounts of unstructured data.

A table is a collection of entities.

Azure Queue service:

Window Azure Queue store large numbers of messages.

Python for MapReduce:-

To create a MapReduce job in Python and run it on a Hadoop cluster.

Let us create a MapReduce job for computing an inverted index from a set of text files.

An inverted index consists of a number of rows where each row holds a unique term and list documents identifiers in which the term occurs.

The map function reads the data from the standard I/p and splits the tab-limited data into document-ID and contents of the document.

The map function emits key-value pairs where key is each word in the document and value is the document-ID.

The key-value pairs emitted by the map phase are shuffled to the reducers and grouped by the key.

The reducer reads the key-value pairs grouped by the same key from the stdin and creates a list of document-IDs in which the word occurs.

The o/p of reducer contains key value pairs where key is for a unique word and value is the list of document-IDs in which the word occurs.

A Hadoop streaming job is then created by specifying the i/p mapper and reducer programs and the locations of the i/p and o/p

Python Packages of Interest:

JSON:

JSON - Java Script Object Notation is an easy to read and write data-Interchange format

JSON is used as an alternative to XML and easy for machines to parse and generate.

JSON is built on two structures

→ a collection of name-value pairs
  eg. Python dictionary.

→ ordered lists of values.
  eg. Python list.

JSON format is often used for serializing and transmitting structured data over a n/w connection, for example, transmitting data b/w a server and web application.

Exchange of information encoded as JSON involves encoding and decoding steps. The Python JSON package provides functions for encoding and decoding JSON.

XML:

XML - Extensible Markup Language is a data format for structured document interchange.

The Python minidom library provides a minimal implementation of the Document object Model interface and has an API similar to that in other languages.

HTTPLib & URLLib:

These are Python libraries used in n/w/ Internet programming.

HTTPLib2 is an HTTP client library and URLLib2 is a library for fetching URLs.

SMTPLib:

Simple Mail Transfer Protocol (SMTP) is a protocol which handles sending email and routing e-mail b/w mail servers.

The Python smtplib module provides an SMTP client session obj that can be used to send email.

Numpy:

Numpy is a package for scientific computing in Python. It provides support for large multi-dimensional arrays and matrices.

Numpy capabilities include:

- Creation and manipulation of arrays.
- Matrix support
- Binary and string operations
- Linear algebra
- Discrete Fourier Transform
- Statistics

→ Random Sampling
→ Sorting, searching and counting
→ Financial functions
→ Polynomials
→ Logic functions.

Sci-kit-learn:

It is an open-source machine learning library for Python that provides implementations of various ML algms for classification, clustering, regression and dimension reduction problems.

It is a very useful package for big data analytics appns. Scikit-learn capabilities include:

→ Supervised learning
  Generalized Linear Models
  Support Vector Machines-SVM
  Stochastic Gradient Descent
  Nearest Neighbors
  Gaussian Processes
  Partial Least Squares
  Naive Bayes
  Decision Trees
  Ensemble Methods
  Multi-class and multilabel algorithms
  Feature selection
  Semi-supervised
  Linear and Quadratic Discriminant Analysis
  Isotonic Regression
→ Unsupervised learning
  Gaussian mixture models.

Manifold learning
clustering
Decomposing signals in components
Covariance estimation
Novelty and outliers Detection
Hidden Markov Models.

→ Model selection and evaluation
Cross-Validation: evaluating estimator performance
Grid search: setting estimator parameters.
Pipeline: chaining estimators.
Feature Union: Combining feature extractors.
Model evaluation.

→ Data set transformations
Preprocessing data
Feature extraction
Kernel Approximation
Random Projection
Pairwise metrics, Affinities and Kernels

→ Dataset loading utilities
General dataset API
Sample images
Sample generators.

Python Web Application Framework :-

Django:

Django is an open source web application framework for developing web applications in Python.

A web app framework in general is a collection of solutions, packages and best practices that allows development of web apps and dynamic websites.

Django is based on the Model-Template view architecture and provides a separation of the data model from the business rules and the user interface.
Django provides a unified API to a database backend.
Thus web apps built with Django can work with different databases without requiring any code changes.
Django is best suited for cloud applications.
Django consists of an object-relational mapper, a web templating system and a regular-expression-based URL dispatcher.

Django Architecture:

Django is MTV - Model Template View framework. The roles of model, template and view are:

Model:

The model acts as a defn of some stored data and handles the interactions with the DB.
In a web app^n, the data can be stored in a relational database, non-relational database, an XML file etc.
A Django model is a Python class that outlines the variables and methods for a particular type of data.

Template:

In a typical Django web app^n, the template is simply an HTML page with a few extra placeholders.
Django's template language can be used to

create various forms of text files (XML, email, CSS, JavaScript, CSV etc).

View:
The view tier the model to the template. The view is where you write the code that actually generates the web pages.
view determines what data is to be displayed, retrieves the data from the DB and passes the data to the template.

Starting Development with Django:

Creating a Django project and App:

when you create a new django project a number of files are created:

__init__.py : This folder is a python package.
manage.py : This file contains an array of functions for managing the site.
settings.py : This file contains the web settings.
urls.py : This file contains the URL patterns that map URLs to pages.

A dango project can have multiple applications. Each app can be part of multiple projects.

model.py : This file contains the description of the models for the application.
views.py : This file contains the app views.

Django comes with built-in, light-weight web server that can be used for development purposes.
when the Django development server is started the default project can be viewed at the URL: http://localhost:8000.

Configuring a DB:

Most web app's are DB backend.
Developers have a wide choice of databases that can be used for web applications including both relational and non-relational databases.
Django provides a unified API for DB backends thus giving the freedom to choose the DB.
Django supports various relational DB engines including MySQL, PostgreSQL, Oracle, and SQLite3
Supports for non-relational DBs such as MongoDB can be added by installing additional engines.

Defining a Model:

Model acts as a defn of the data in the DB.
Model which contains all the functionality that allows the models to interact with the DBs.
The syncdb command is run the first time, it creates all the tables defined in the Django model in the configured DB.

Django Admin Site:

Django provides an administration system that allows you to manage the website without writing additional code.
The admin system reads the Django model and provides an interface that can be used to add content to the site.

Defining a View:

Views contains the logic that glues the model to the template.
The view determines the data to be displayed in the template, retrieves the data from the DB and passes it to the template.
Views can also perform additional tasks such as authentication, sending emails etc.

Defining a Template:

A Django template is typically an HTML file.
Django templates allow separation of the presentation of data from the actual data by using placeholders and associated logic.
A template receives a context from the view and presents the data in context variables in the placeholders.

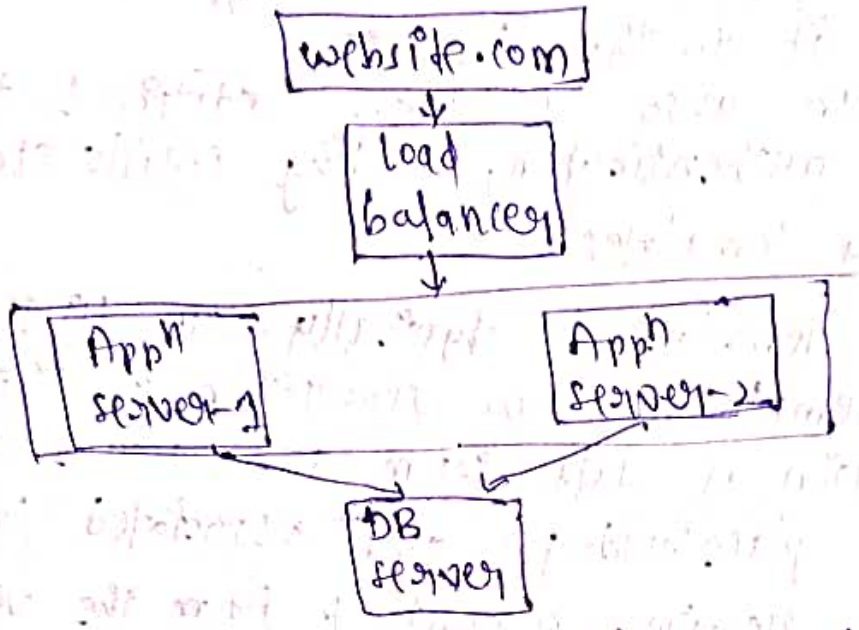
Defining the URL Patterns:

URL patterns are a way of mapping the URLs to the views that should handle the URL requests.
The URL requested by the users are matched with the URL patterns and, the view corresponding to the pattern that matches the URL is used to handle the request.

Django Case Study - Blogging App:

The deployment consists of a load balancer (HAProxy), application servers (Django) and DB servers (MongoDB).
MongoDB is a popular open-source No-SQL database. MongoDB stores data in JSON like documents with

dynamic schemas,
An element of data in MongoDB is called a document, and documents are stored in collections. A collection may have any number of documents.



Multi-tier deployment architecture for blogging app

Designing a RESTful Web API:-

REST - Representational State Transfer.
With the Django framework, the Django REST framework can be installed as follows:

```
pip install django.restframework
pip install markdown
pip install django-filter.
```

After installing the Django REST framework, create a new Django project named restful api, and then start a new app called myapp.

```
django-admin.py startproject restfulapi
cd restfulapi
python manage.py startapp myapp
```

View sets are used for the views that allow you to combine the logic for a set of related views in a single class.

serializers allow complex data to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types.

serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the incoming data.

Routers automatically determining how the URLs for an appn should be mapped to the logic that deals with handling incoming requests.

After creating the Books REST API source files, the next step is to setup the DB and then run the Django development web server as follows:

python manage.py syncdb
python manage.py runserver

28/03/2024 [signature]

10:28 PM

[signature]

Cloud Appⁿ Dev in Python

Design Approaches, Image Processing App, Document Storage App, MapReduce App, Social Media Analytics App.

Design Approaches :-

Appⁿ design approaches that leverage the IaaS and PaaS cloud service models.

Design Methodology for IaaS service model:

Traditional appⁿ design approaches such as Service Oriented Architecture (SOA) use component based designs.

The components in these approaches can span multiple tiers such as web, appⁿ and DB tiers which makes it difficult up map them to multi-tier cloud architecture.

CCM - Cloud Component Model approach for designing cloud appⁿs, which is a more recent approach that classifies the components based on the types of functions performed and types of cloud resources used.

The CCM design approach is suited for appⁿs that use the IaaS cloud service model.

With the IaaS model the developers get the flexibility to map the CCM architecture to cloud ~~service model~~ deployments.

Virtual machines for various tiers can be provisioned and auto scaling options for each tier can be defined.

CCM approach is based on loosely coupled and stateless designs.

Messaging queues are used for asynchronous communication. CCM components expose functional interfaces such as REST APIs for loose coupling and performance interfaces for reporting the performance of components.

An external status DB is used for storing the state.

The benefits of CCM approach are:

Improved Application Performance:

CCM use loosely coupled components that communicate asynchronously. It makes possible to scale-up (boost) the app^n components that limit the performance.

Savings in Design, Testing & Maintenance Time:

The CCM methodology achieves savings in app^n design, testing and maintenance time.

Reduced Application cost:

App^{n}s designed with CCM can leverage both vertical and horizontal scaling options for improving the app^n performance.

Reduced Complexity:

A simplified deployment architecture can be more easier to design and manage.

Depending on app^n performance and cost requirements, it may be more beneficial to scale vertically instead of horizontally.

Component Design

→ Identify the building blocks of the app^n and the functions to be performed by each block.

→ Group the building blocks based on the functions performed and type of cloud resources required.

→ Identify the i/ps and o/ps of each component
→ List the interfaces that each component will expose.
→ Evaluate the implementation alternatives for each component.

Architecture design

→ Define the interactions b/w the appn components.
→ Guidelines for loosely coupled and stateless designs - use messaging queues, functional interfaces, and external state DB.

Deployment Design

→ Map the appn components to specific cloud resources such as web servers, appn servers, DB servers etc.

Design Methodology for Paas service model:

For appns that use the Paas cloud service model, the architecture and deployment design steps are not required since the platform takes care of the architecture and deployment.

In the component design step, the developers have to take into consideration the platform specific features.

Eg: Appns designed with GAE can leverage the GAE Image Manipulation service for image processing tasks.

Different Paas offerings such as GAE, Windows Azure Web sites etc provide platform specific SDKs for developing cloud application.

Appns designed for specific PaaS offerings run in sandbox environments and are allowed to perform only those actions that do not interfere with the performance of other appns.

The deployment and scaling is handled by the platform while the developers focus on the appn dev using the platform-specific SDKs.

Portability is a major constraint for PaaS based appns as it is difficult to move the appn from one cloud vendor to the other due to the use of vendor specific APIs and PaaS SDKs.

Image Processing App:-

This appn provides online image filtering capability. Users can upload image files and choose the filters to apply. The selected files are applied to the image and the processed image can then be downloaded.

→ The Web tier for the image processing app has front ends for image submission and displaying processed images.

→ The appn tier has components for processing the image submission requests, processing the submitted image and processing requests for displaying the results.

→ The storage tier comprises of the storage for processed images. This is the component design step for the image processing app.

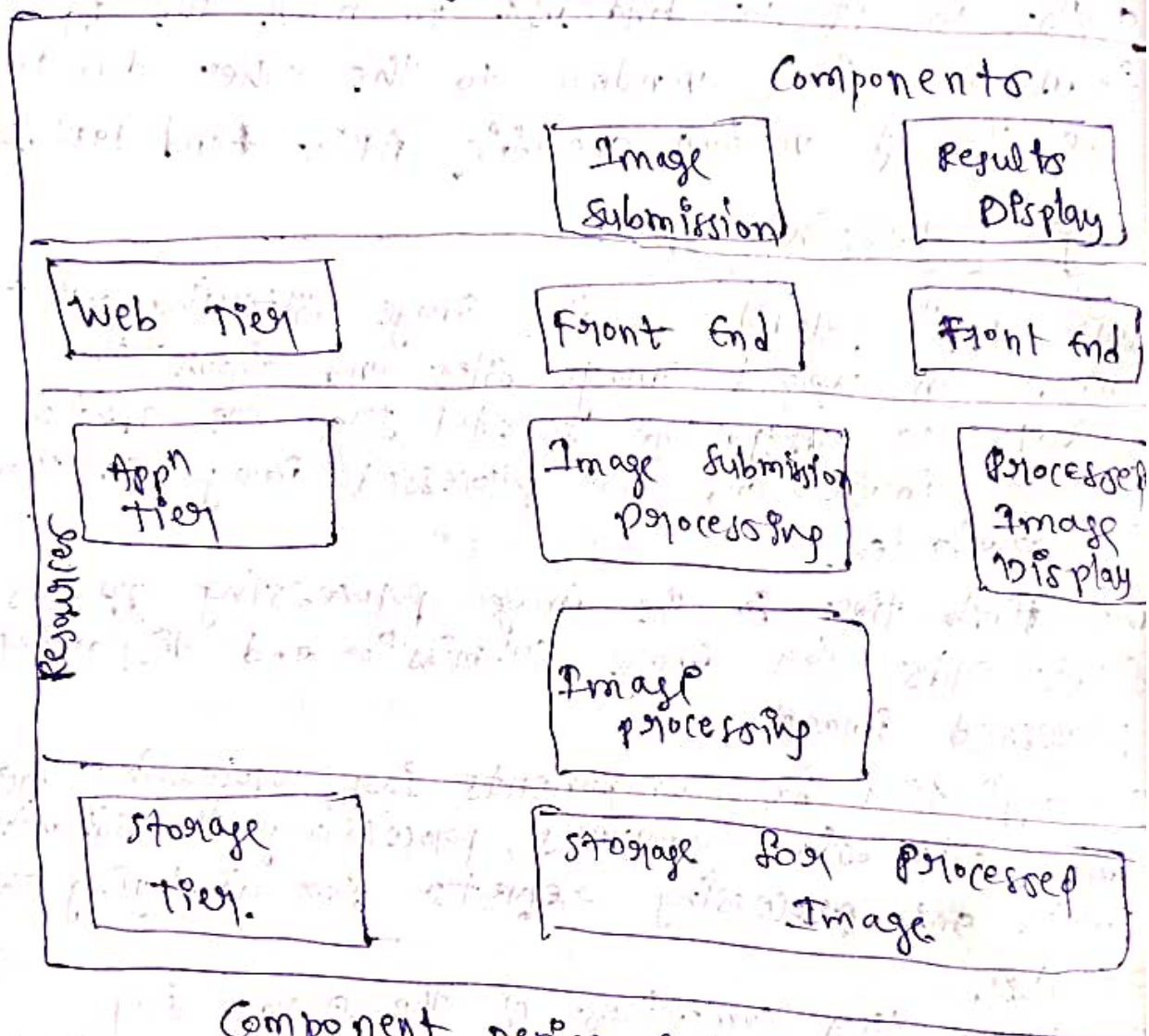
→ The architecture design defines the interactions b/w the appn components.

→ This appn uses the Django framework, therefore,

the web tier components map to the Django templates and the app tier components map to the Django views.

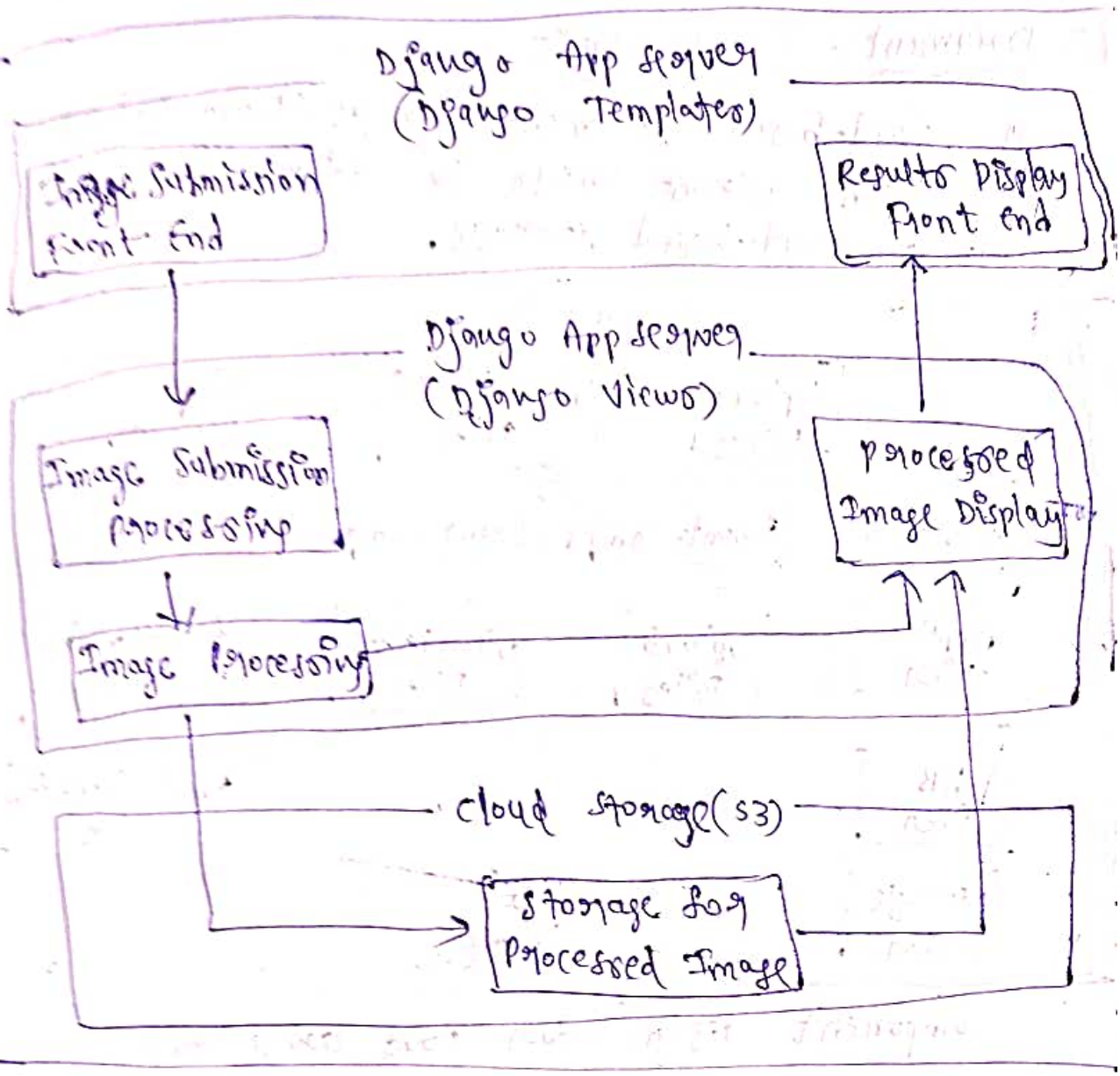
→ A cloud storage is used for the storage tier.

→ The deployment design, this is multi-tier architecture comprising of load balancer, app^n servers and a cloud storage for processed images.

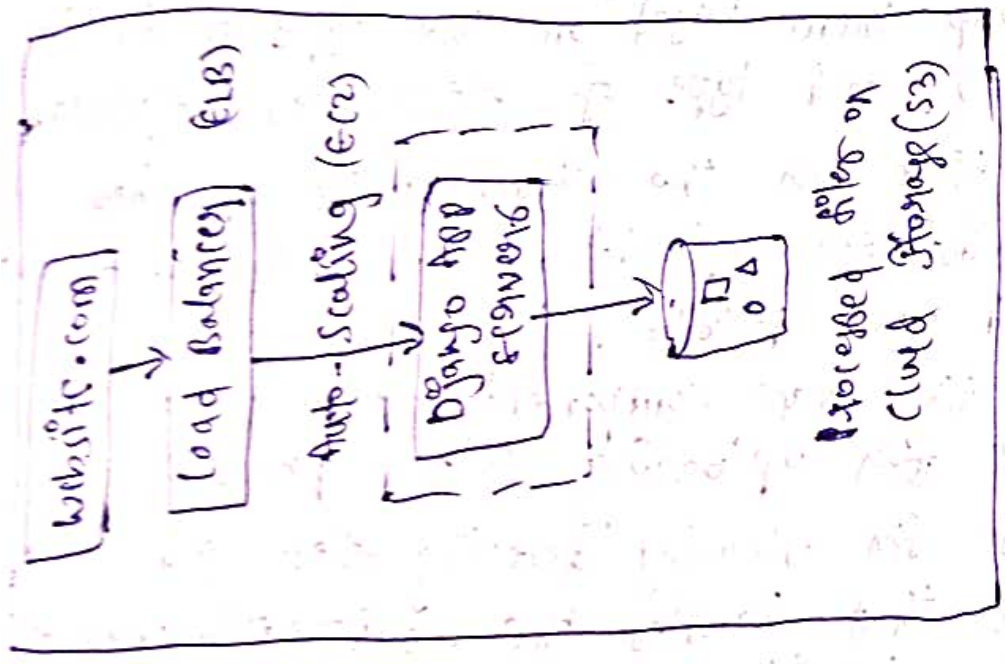


Component design for Image Processing App

PIL - Python Imaging Library



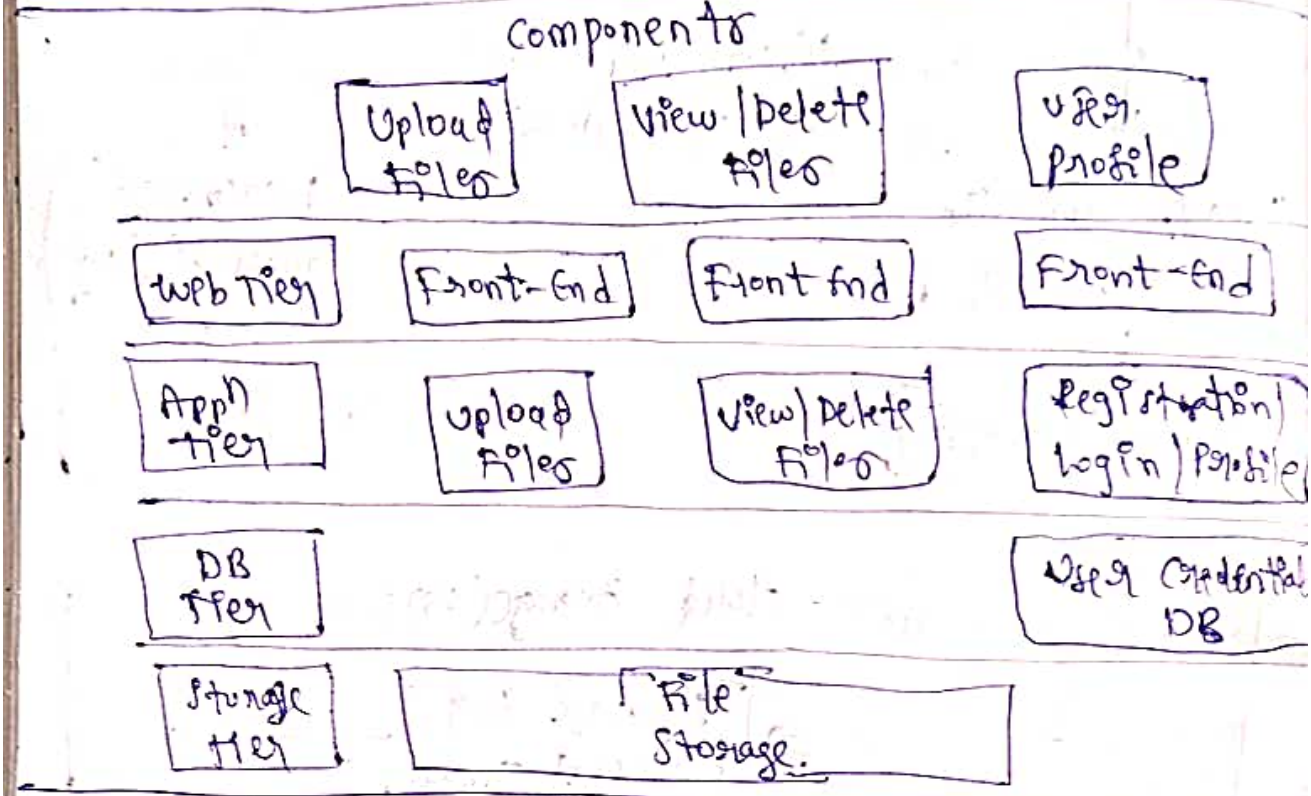
Architecture Design for Image Processing App



Deployment design for Image Processing App

Document Storage App:-

A cloud-based document storage (Cloud Drive) application allows users to store documents on a cloud-based storage.



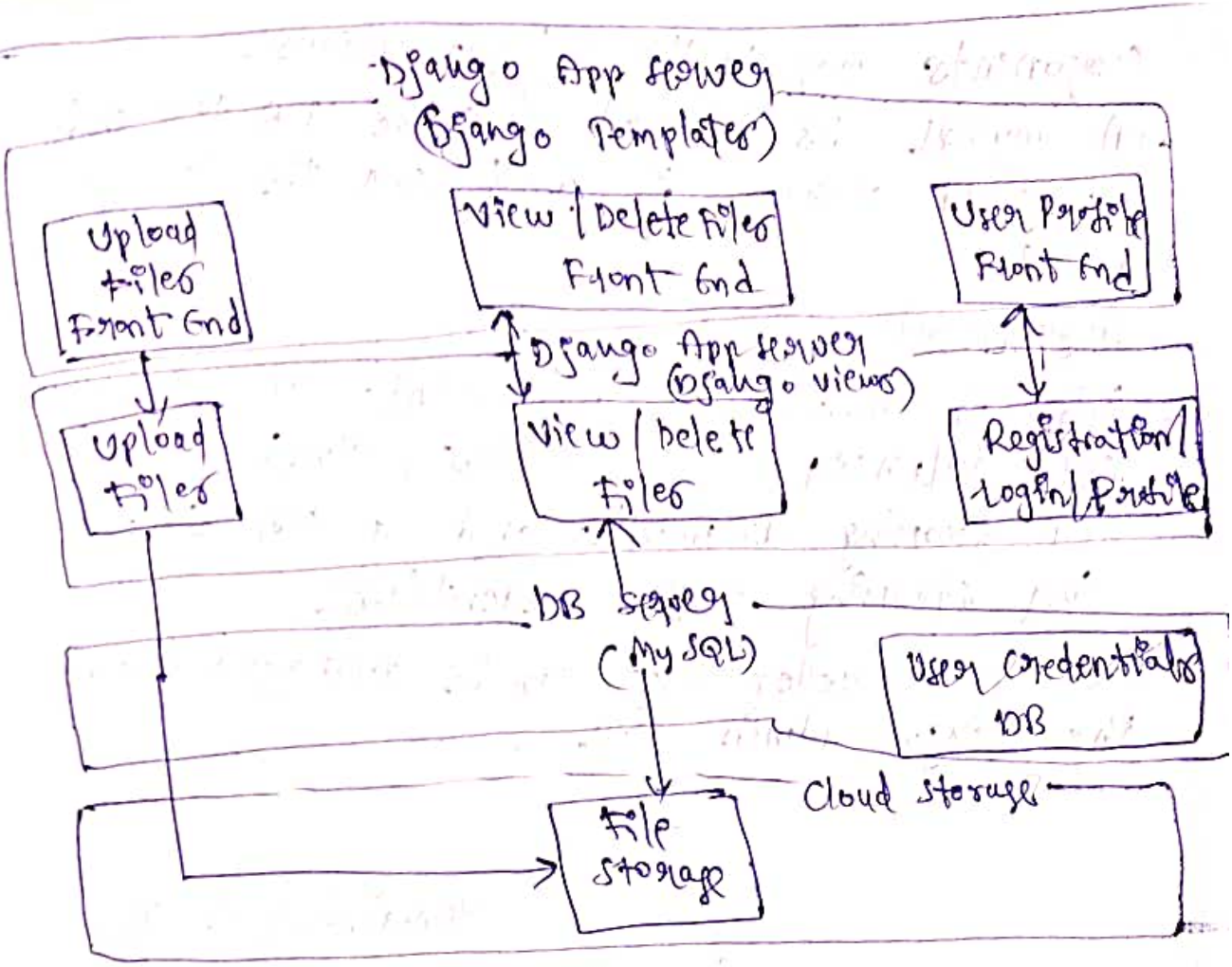
Component design for cloud drive App

Component design:

In this step we identify the appn components and group them based on the type of functions performed and type of resources required.

→ The web tier for the cloud drive app has front ends for uploading files, viewing/deleting files and user profile.

→ The appn tier has components for processing requests for uploading files, processing requests for viewing/deleting files and the component that handles the registration, profile and login functions.



Architecture design for cloud drive App

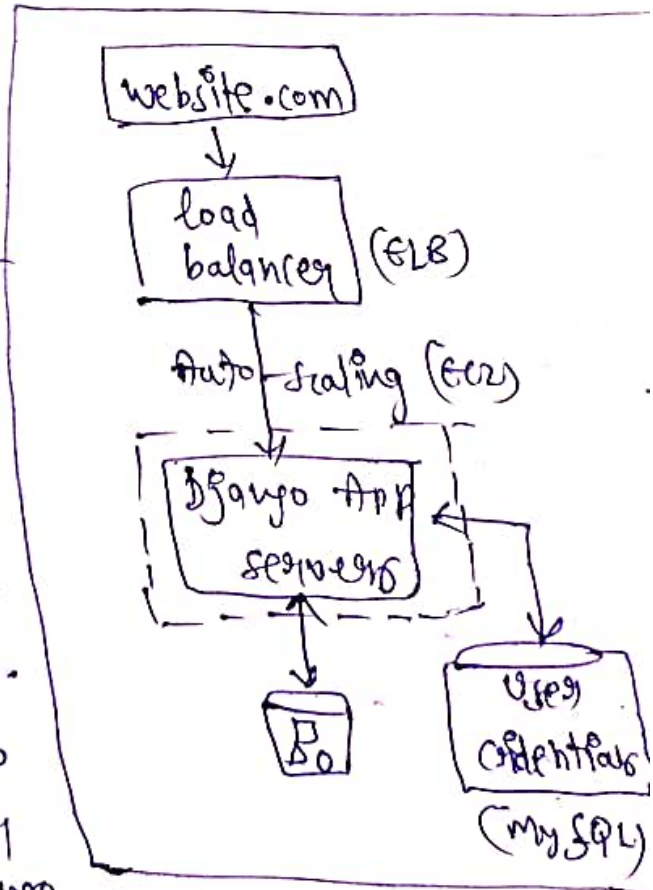
→ The DB tier comprises of a user credentials database.

→ The storage tier comprises of the storage for files.

Architecture Design:

In this step which defines the interactions b/w the app^n components.

→ This app^n uses the Django framework, the web tier components map to the Django templates and the app^n tier



Deployment design for cloud drive app

components map to the Django views.

→ A mySQL DB is used for the DB tier and a cloud storage is used for the storage tier.

Deployment Design:

This is multi-tier deployment comprising of load balancer, app^n servers, cloud storage for storing documents and a DB server for storing user credentials.

The user credentials can be managed from the Django admin site.

Remaining in the another book (a

Solved //

17:18 Fri, 29, May 2024.

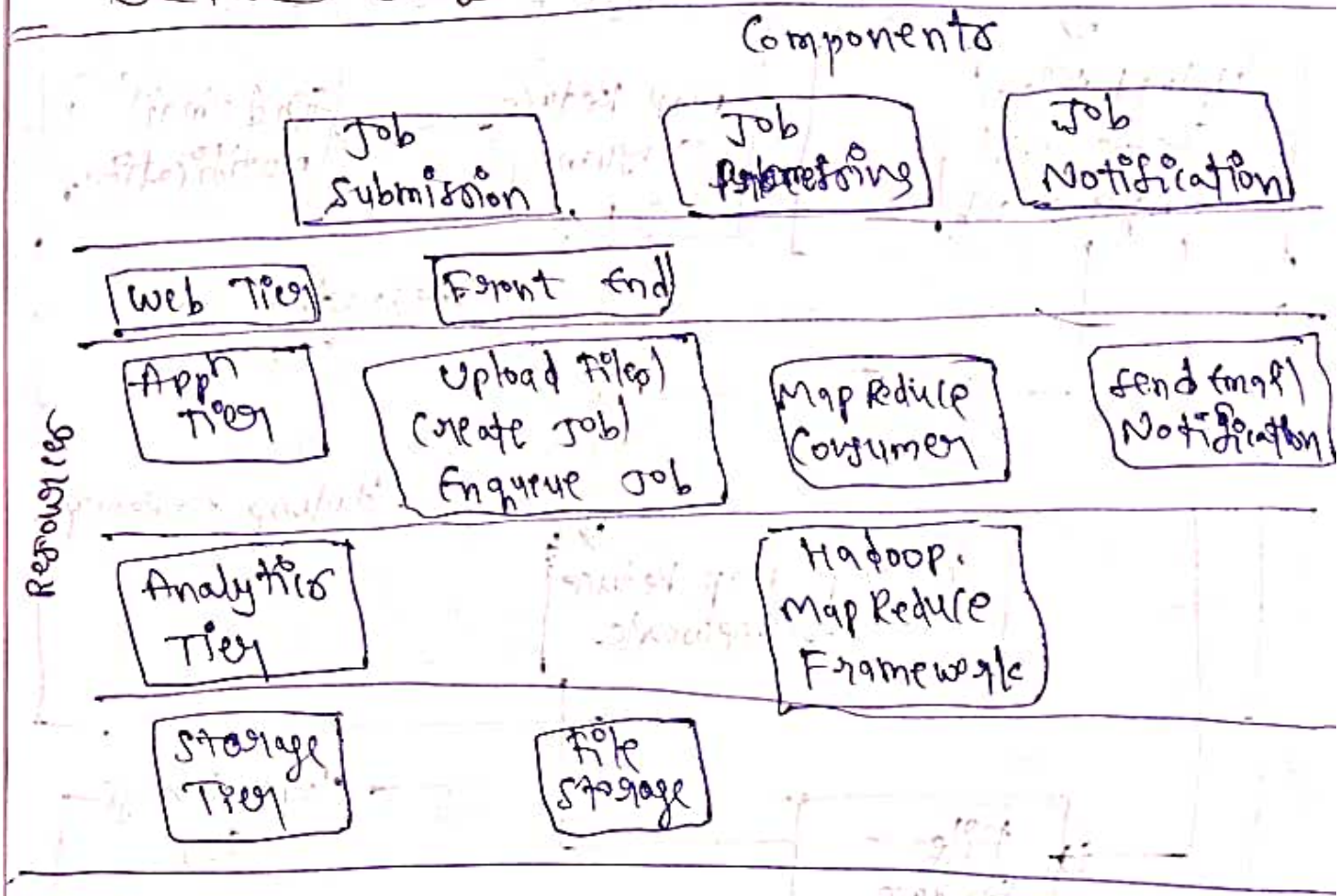
17:26, Fri, 29, May 2024, Gurgaon.

MapReduce App :-

A MapReduce app allows users to submit MapReduce jobs for data analysis. This app is based on the Amazon Elastic MapReduce (EMR) service.

Users can upload data files to analyze and choose/upload the Map and Reduce programs. The selected Map and Reduce programs along with the i/p data are submitted to a queue for processing.

Component design:

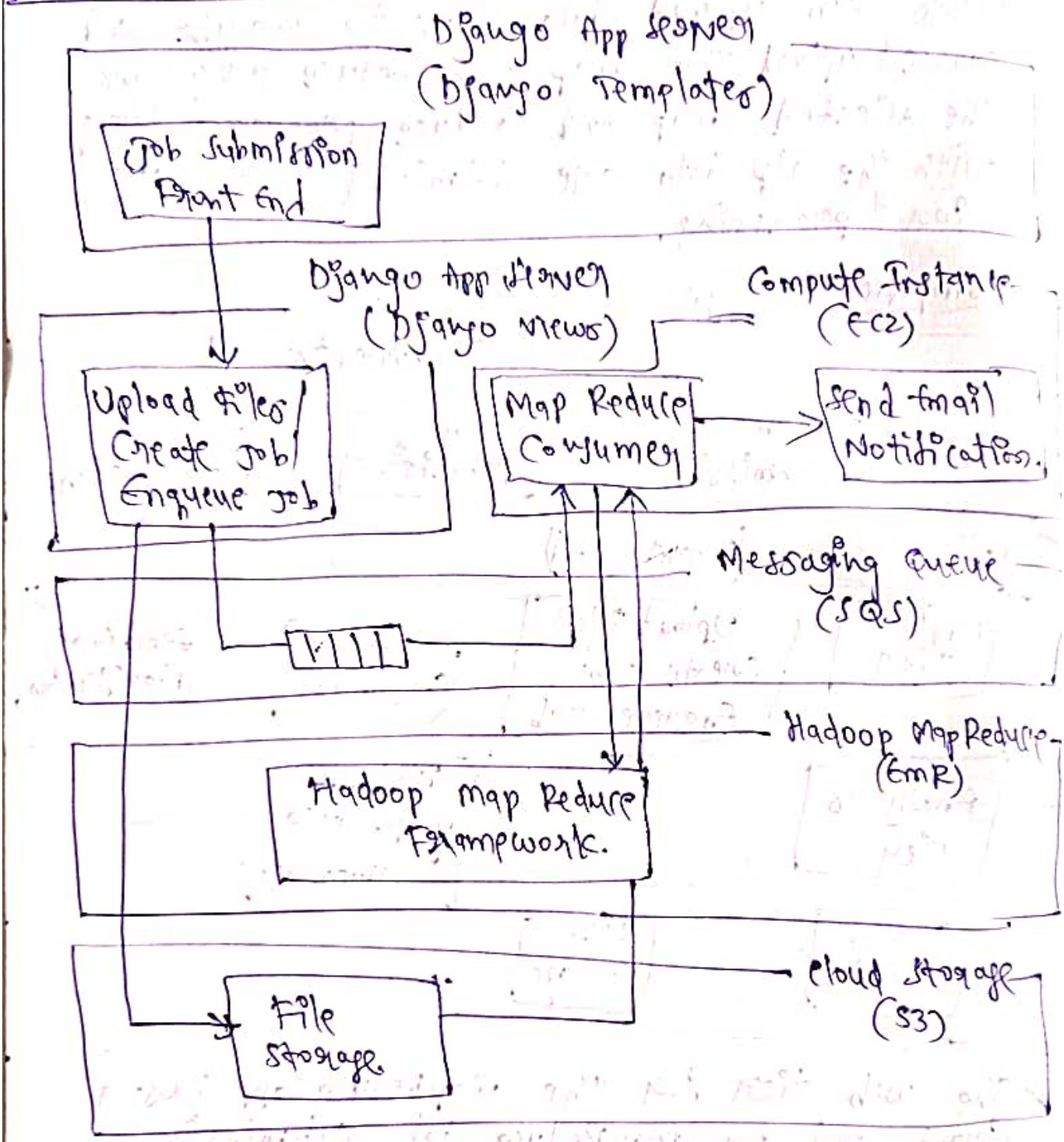


→ The web tier for the MapReduce app has a front end for MapReduce job submission.

→ The appn tier has components for processing requests for uploading files / creating MapReduce jobs and enqueuing jobs, MapReduce consumer and the component that sends email notifications.

→ The Hadoop framework is used for the analytics tier and a cloud storage is used for the storage tier.

Architecture design:



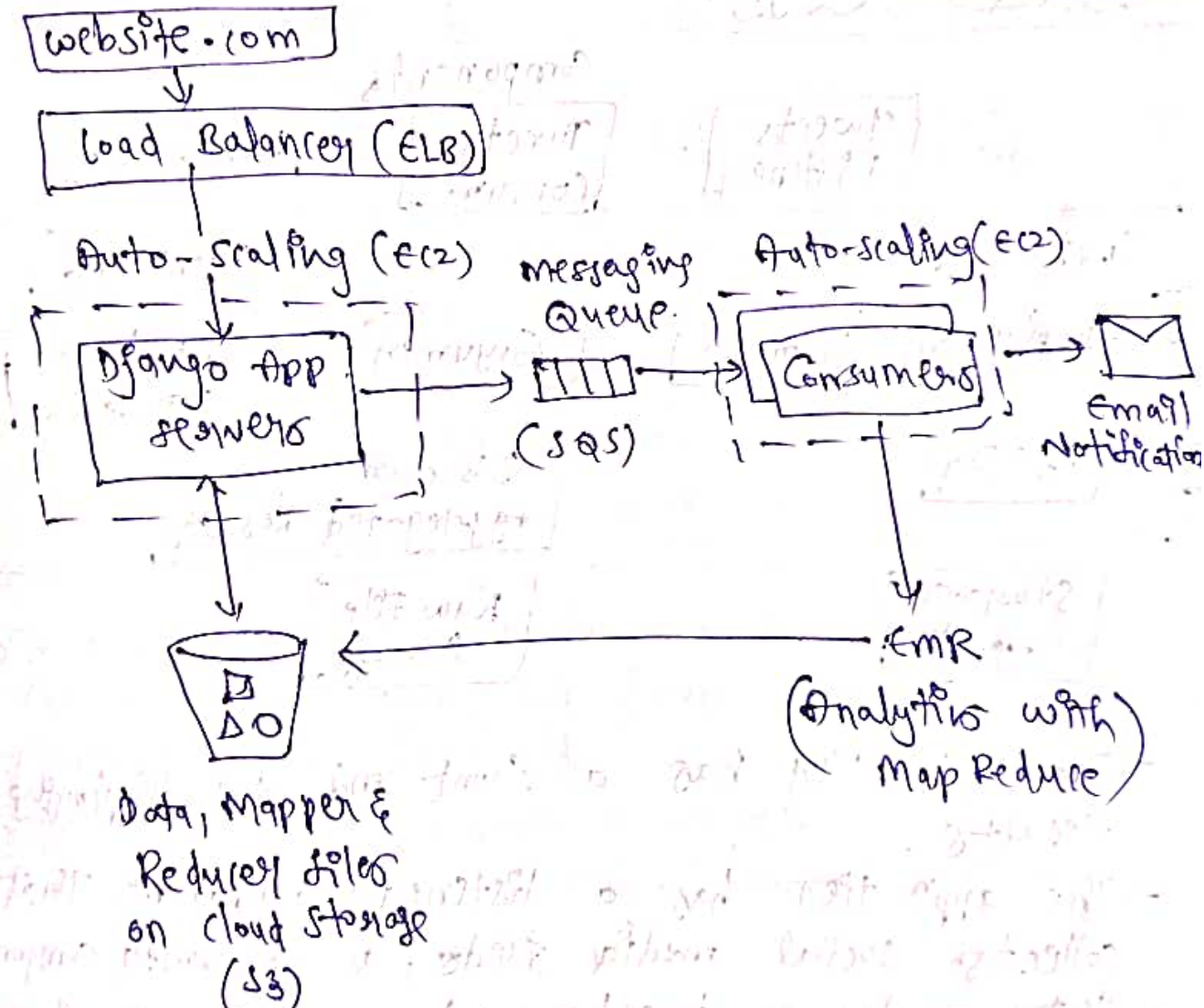
→ This appn uses the Django framework, the web tier components map to the Django templates and the ~~corresponding code~~ appn tier components map to the Django views.

→ To make the app^n scalable the job submission and job processing components are separated. The MapReduce job requests are submitted to a queue.

→ A consumer component that runs on a separate instance retrieves the MapReduce job requests from the queue and creates the MapReduce jobs and submits them to the Amazon EMR service.

→ The user receives an email notification with the download link for the ~~MapReduce app~~ results when the job is complete.

Deployment Design:



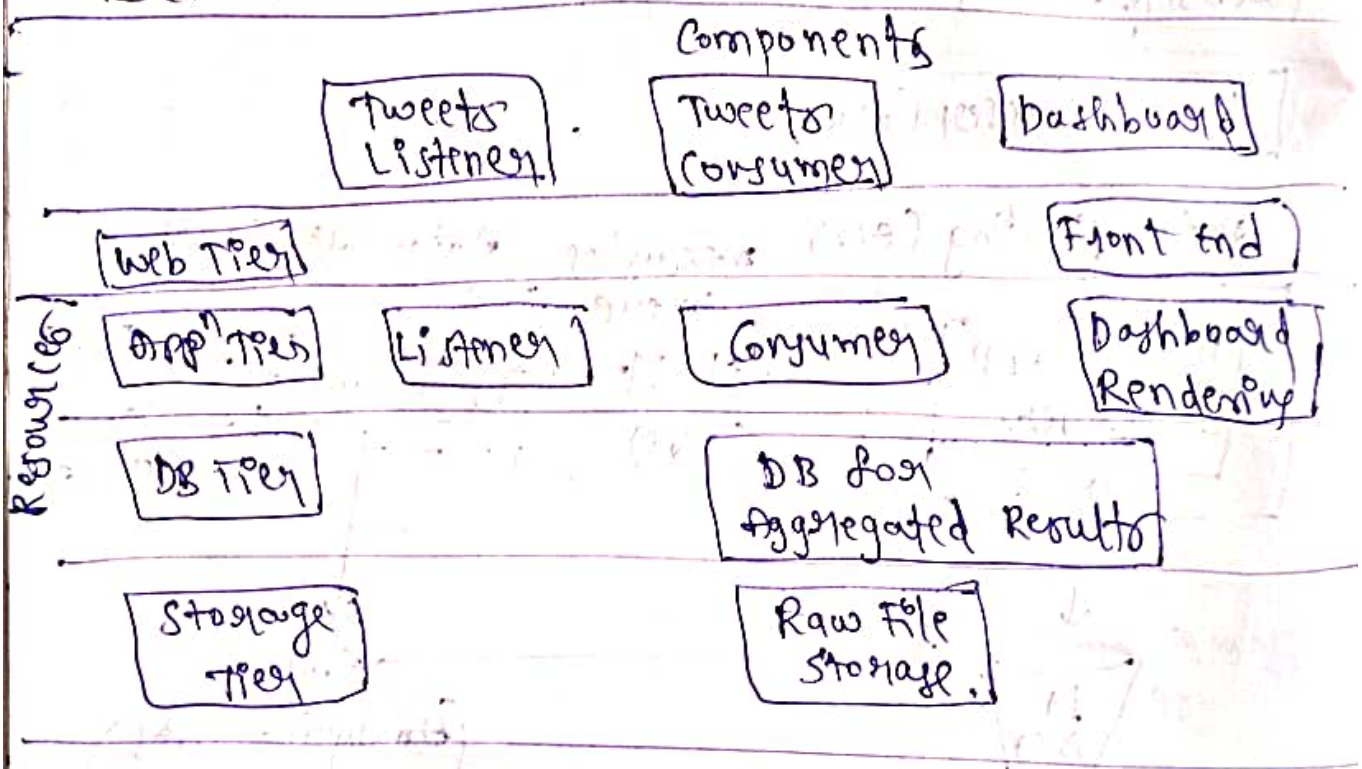
→ This is a multi-tier architecture comprising of load balancer, app^n servers and a cloud storage for storing MapReduce programs, i/p data and MapReduce o/p.

FIFO – First-In-First-Out
SNS – Simple Notification Service

Social Media Analytics App:-

A social media analytics app^n collects the social media feeds on a specified keyword in real time and analyzes the sentiments of the tweets and provides aggregate results.

Component Design:

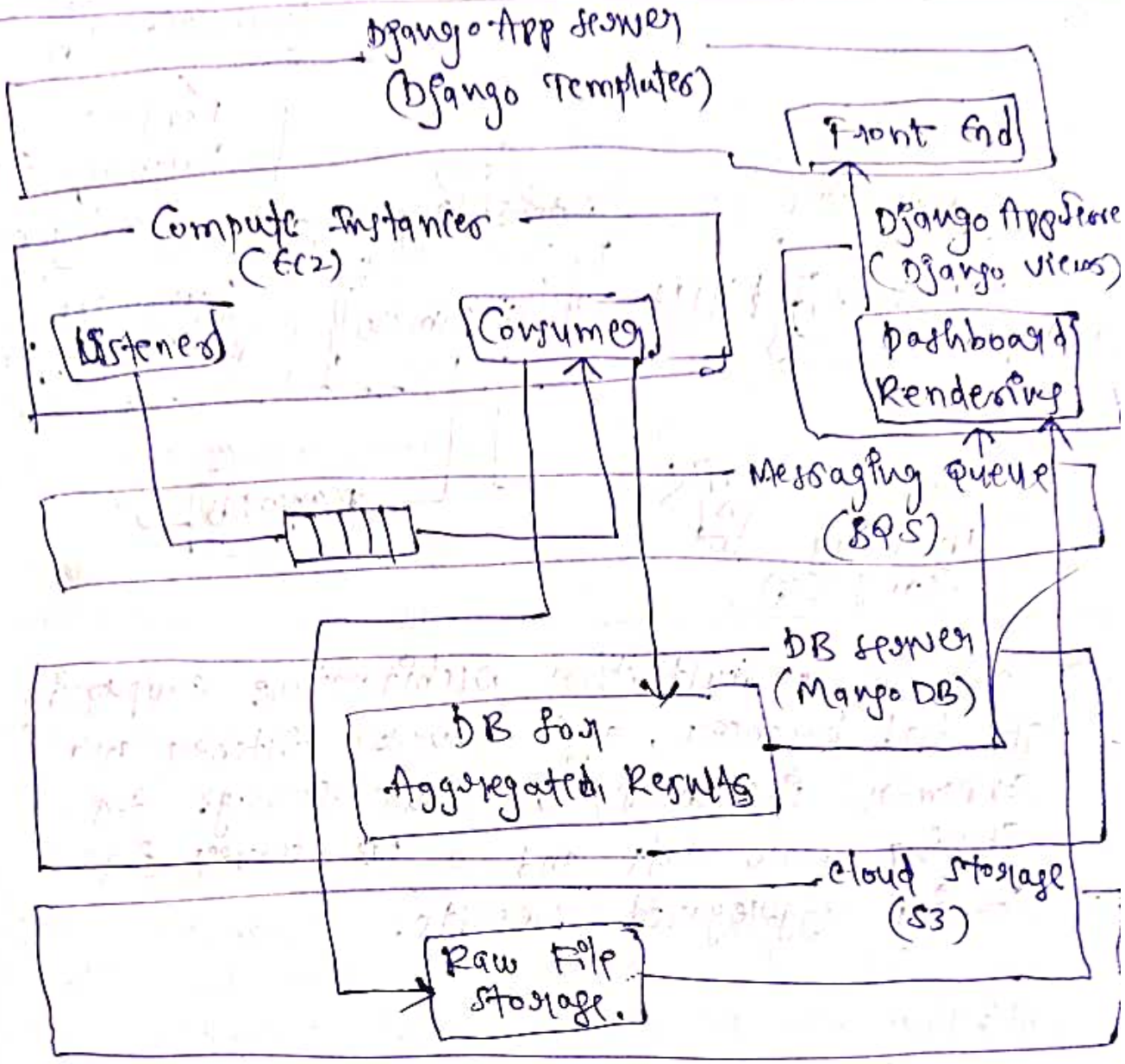


→ The web tier has a front end for displaying results.

→ The app^n tier has a listener component that collects social media feeds, a consumer component that analyzes tweets and a component for rendering the results in the dashboard.

→ A Mango DB database is used for the DB-tier and a cloud storage is used for the storage tier.

Architecture Design:



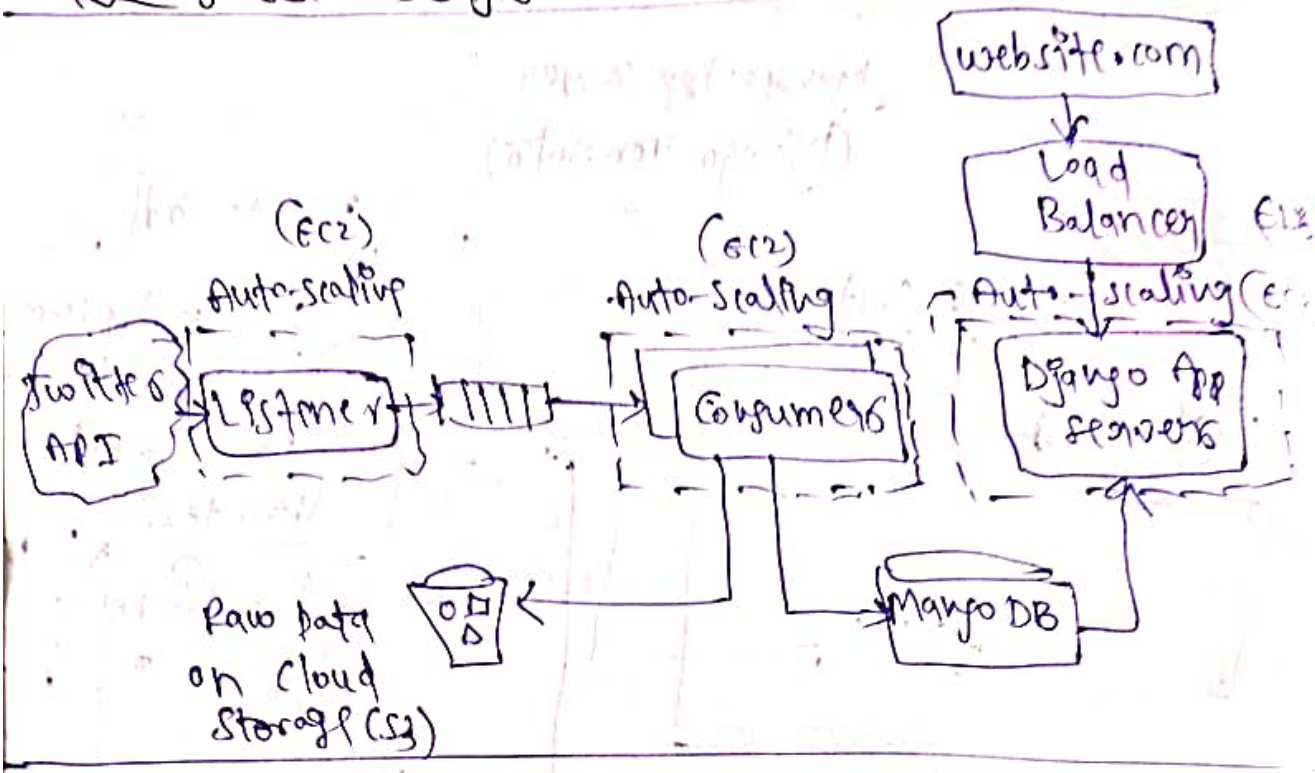
→ To make the app scalable the feeds collection component (Listener) and feeds processing component (Consumer) are separated.

→ The Listener component uses the Twitter API to get feeds on a specific keyword and enqueues the feeds to a queue.

→ The consumer component retrieves the feeds from the queue and analyzes the feeds and stores the aggregated results in a separate DB.

→ The aggregate results are displayed to the users from a Django appn.

Deployment Design:



→ This is a multi-tier architecture comprising of load balancer, appn servers, listener and consumer instances, a cloud storage for storing raw data and a DB server for storing aggregated results.