DataCloud Analytics

Data Analytics as a Service

Akash Waghela, Pranshu Gupta

Introduction to Software Engineering [CS455A] – Course Project
Indian Institute of Technology Kanpur
Department of Computer Science & Engineering

INTRODUCTION

The software we have implemented is a web service that will be used by businesses to visualize their consumer and product usage data to inform and measure their marketing. It is an instance of Analytics as a Service.

DATA ANALYTICS AS A SERVICE

Analytics as a Service (AaaS) refers to the provision of analytics software and operations through Web-delivered technologies.

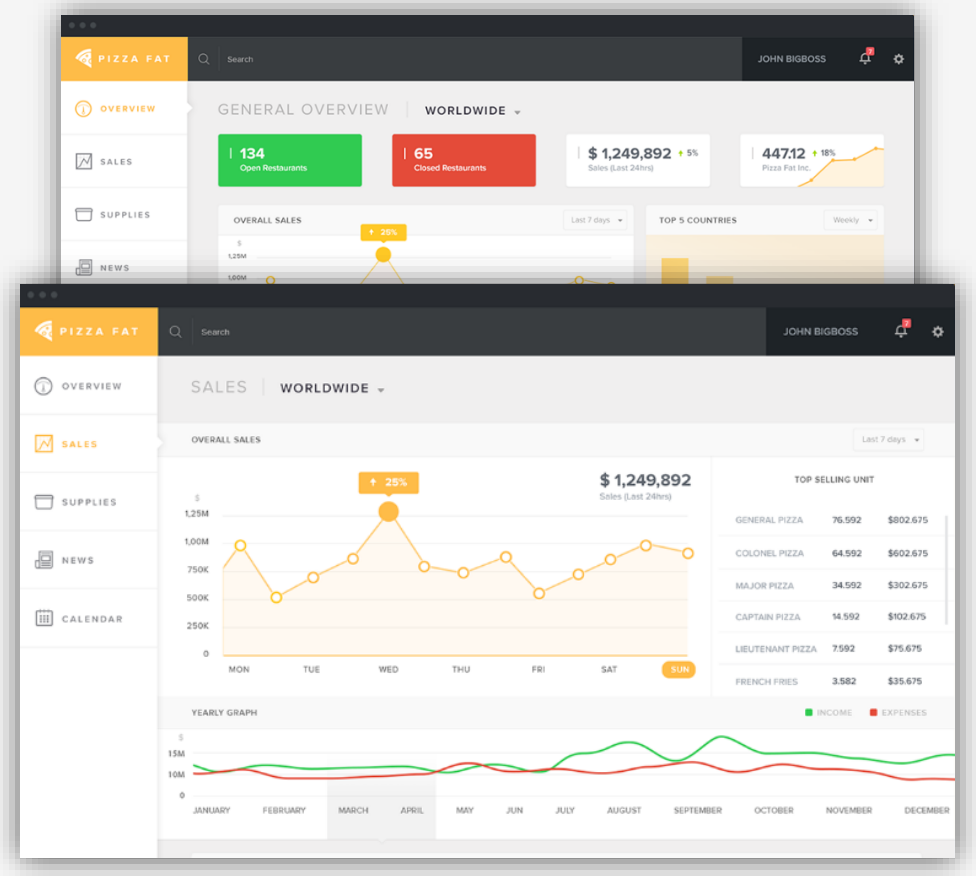
DESCRIPTION

USER INTERFACE

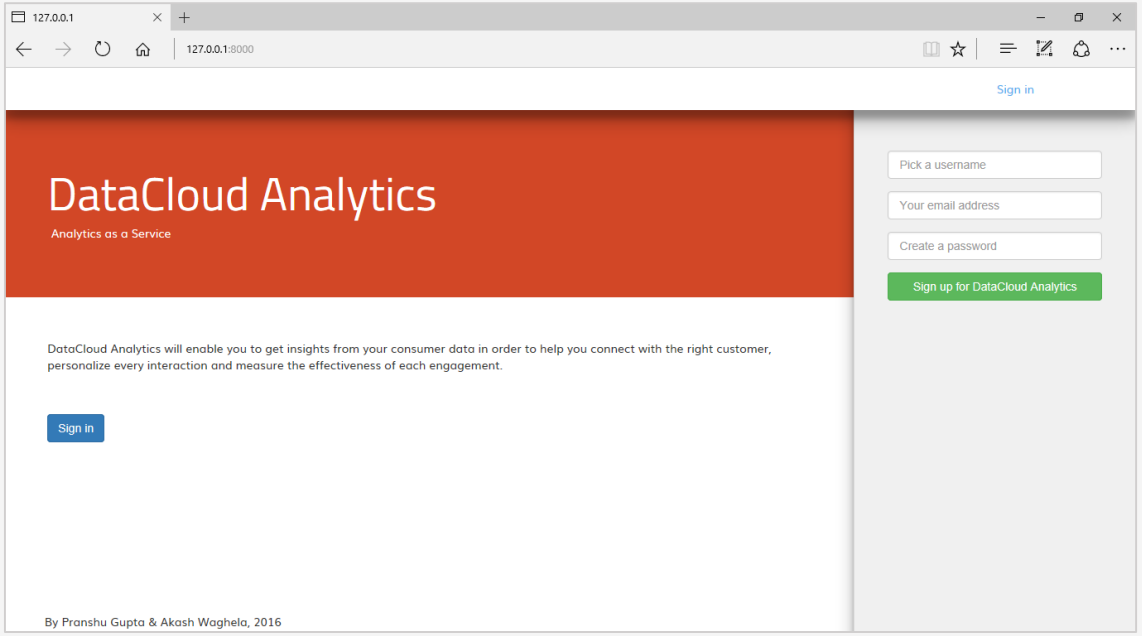
The user interface of our application consists of the following screens:

1. The User Registration Screen
2. The User Login Screen
3. The Dashboard
    a. General Analytics Screen
    b. Geographical Activity Heat Map Screen

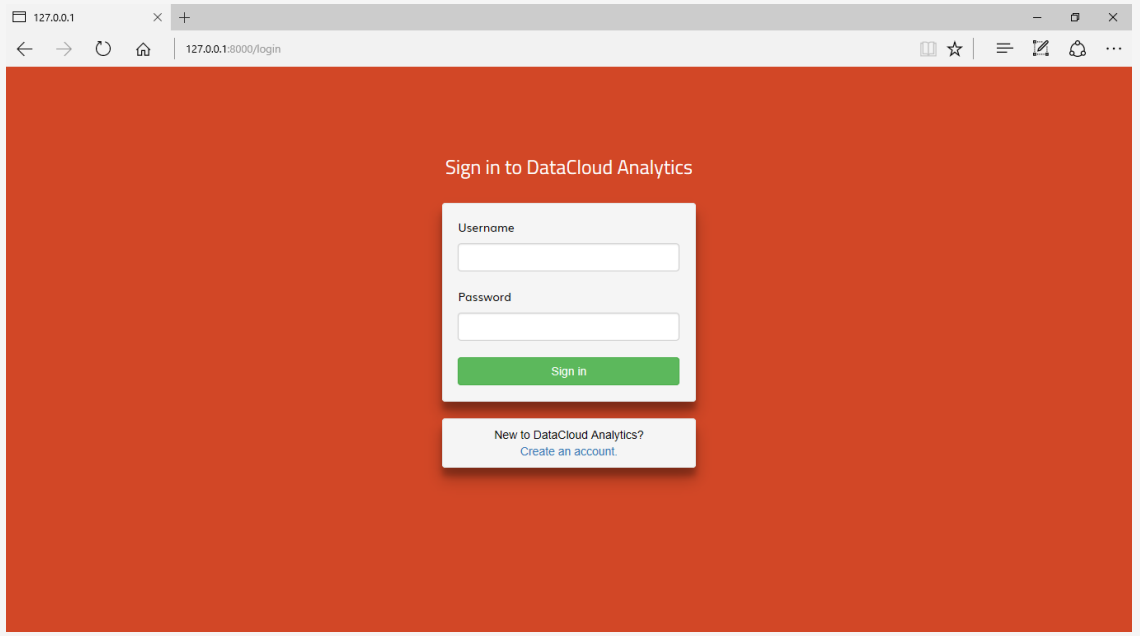
We will utilize the live product usage data to build insights and create dashboard for the business user. The dashboard will have rich data visualizations that will prove useful for building new marketing strategies.



UI - USER REGISTRATION/LOGIN SCREEN

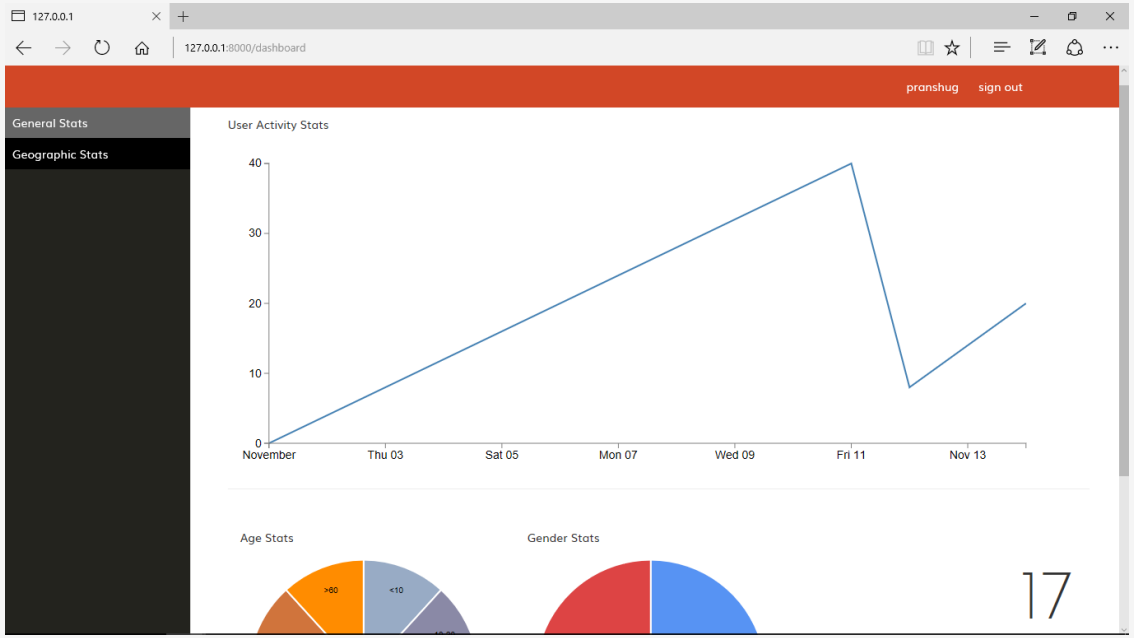


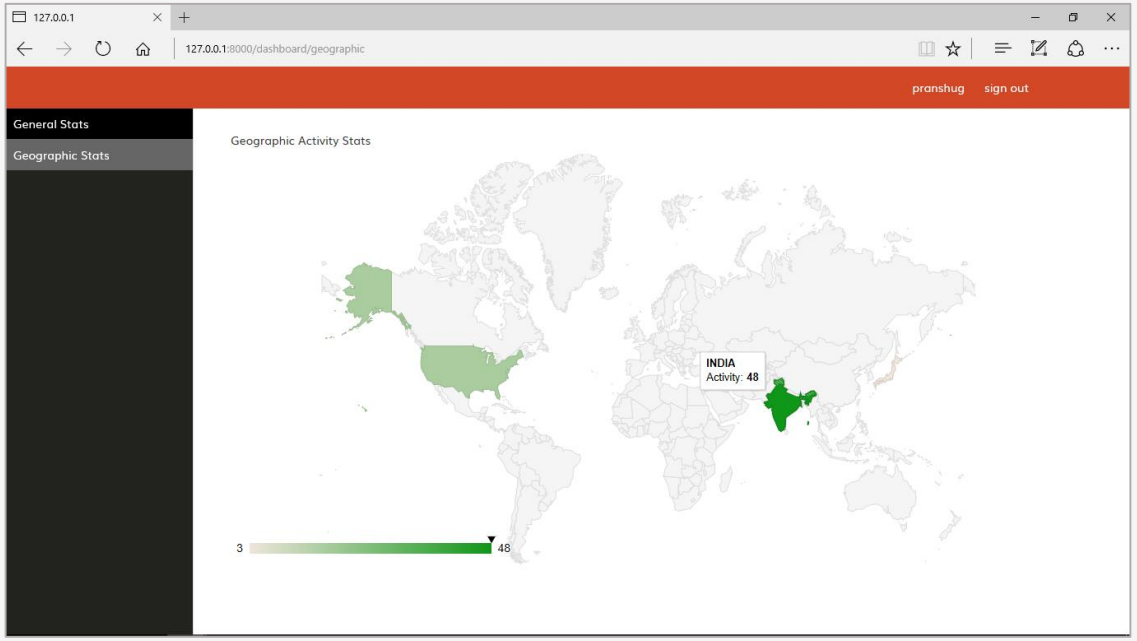
This is the landing page of our application. It provides some general information about the service along with a user registration form



This is the screen where the user can log in to our application and utilize the services provided. If a new user directly lands on this page, there is an option to create a new account here too.

UI - DASHBOARD





The dashboard is what the user sees after login. The layout of this page has three parts, the top header, the navigation pane and the content view. The navigation pane provides the user with links that can be clicked to view different categories of analytics visualizations.

DESIGN PATTERN

MTV

We have followed the design pattern defined by Django Web Framework for developing our application. Django is said to follow MVC, although some say it is MTV. MTV (Model Template View) is very similar to MVC (Model View Control) but there is a subtle difference.

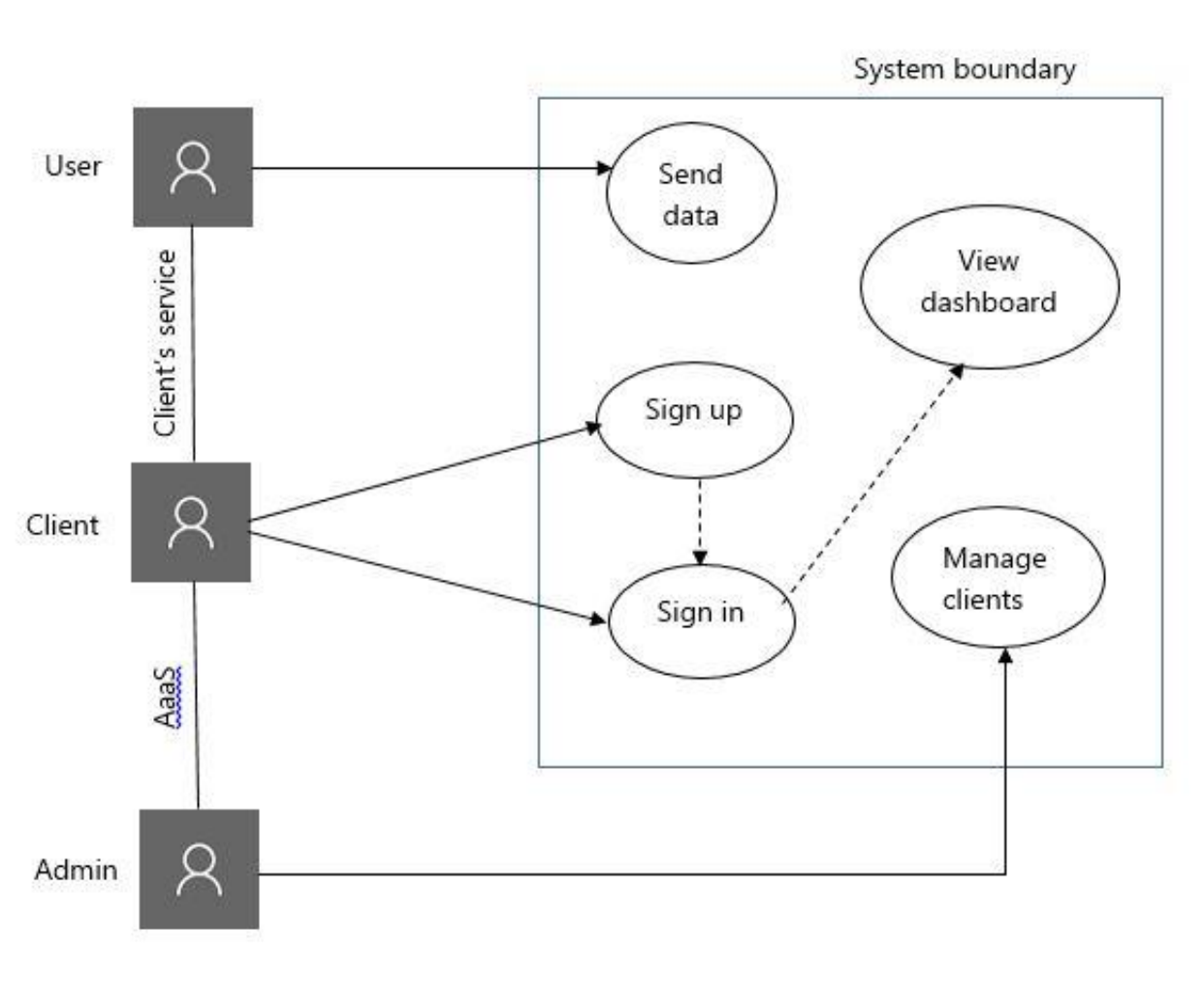
In MTV the template describes the data that gets presented to the user, not necessarily how it is presented, but what is presented. In contrast, MVC suggests that it is the controller's job to decide how the data is presented while the view is strictly responsible for how the data looks. So, a 'template' in MTV loosely replaces a 'view' in MVC. While the job of an 'MVC Controller' is handled partly by a 'MTV View' and partly by the framework itself.

ARCHITECTURE

USE CASE VIEW

We have three actors in our application, the *Service Provider* (admin), our *Client* and the *Users* of the client's service for which the client has opted for our "AaaS" offering.

- Sign UP/ Sign In
- View Dashboard
- Send Data
- Manage Clients

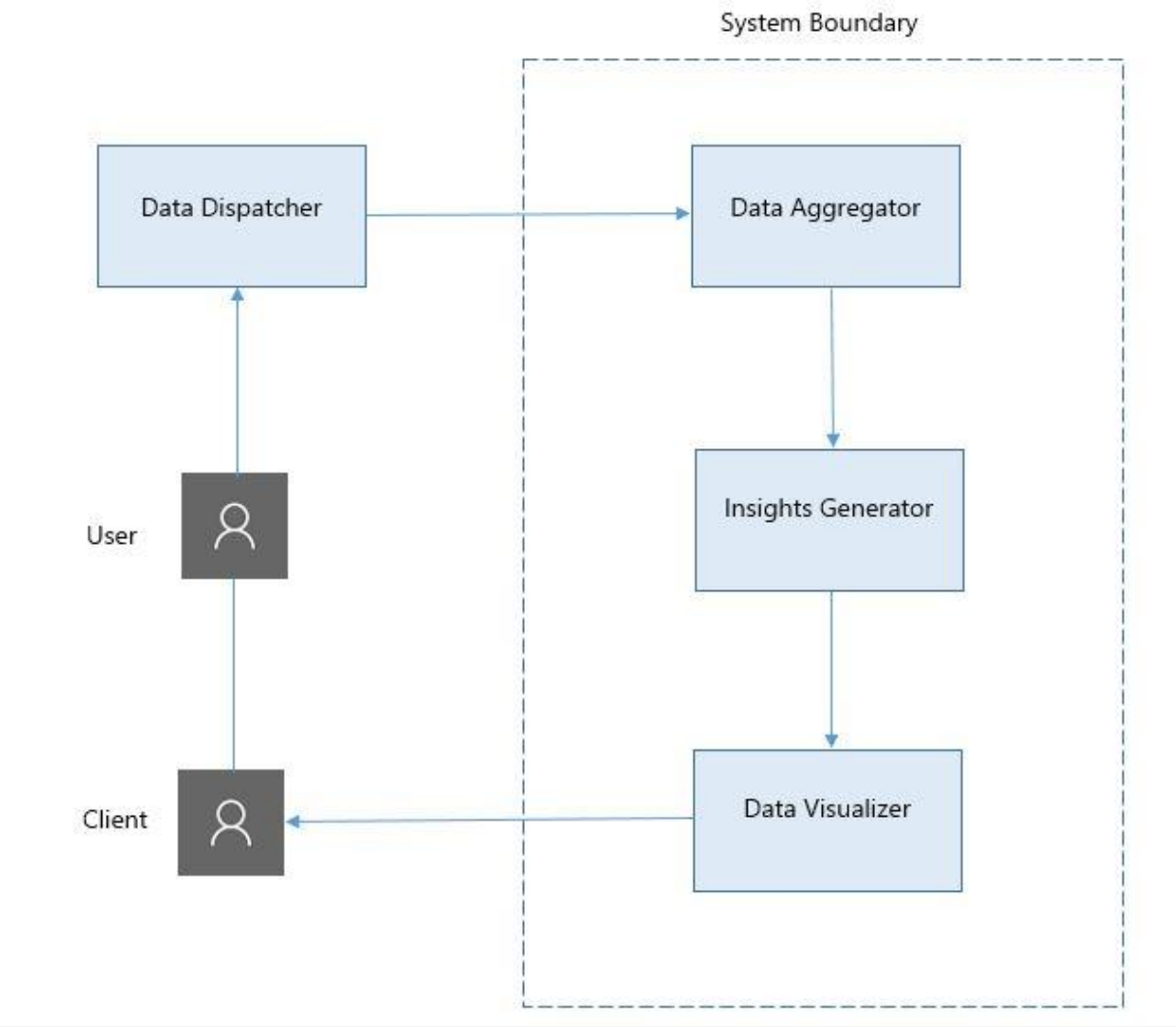


ARCHITECTURE

LOGICAL VIEW

Following are the different modules of our system:

- Data Dispatcher – A script that collects data from the user of the client's service and sends it to our system
- Data Aggregator – The module responsible for collecting the received data and populating the database.
- Insights Generator – The module responsible for generating insights that will be used to create visualization on the dashboard for the client.
- Data Visualizer – The module responsible for creating appropriate visualizations for the insights generated by the Insights Generator.



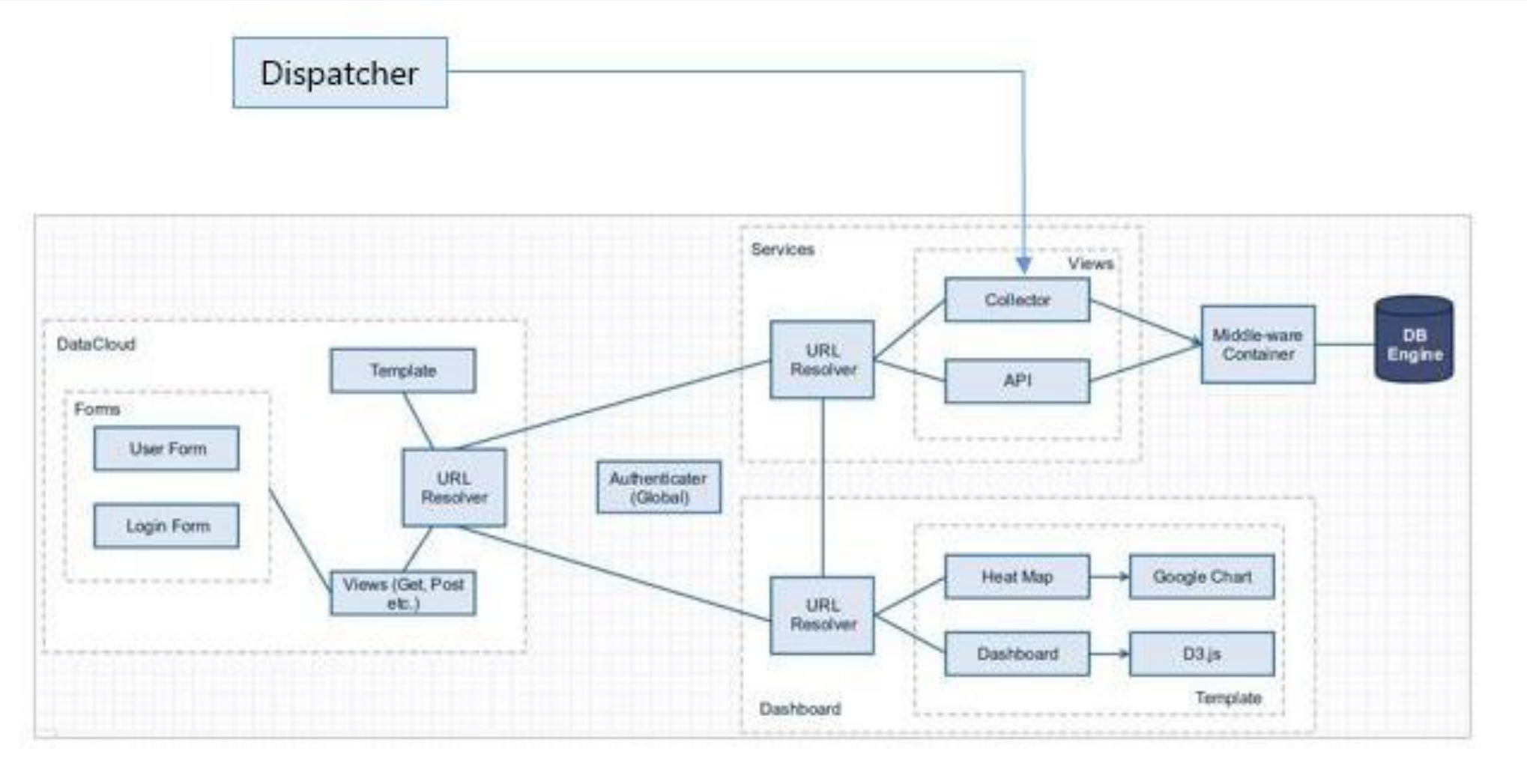
ARCHITECTURE

PROCESS VIEW

The entire application can be divided into the following components:

1. Authenticator/Authorizer – Globally accessible, used for authentication while writing and reading data.
2. URL Resolver – Every Component has this block to redirect http requests to appropriate end point.
3. Data Collector – This process collects data sent by the data dispatcher.
4. Service API – This process is activated by the web-application while querying Database for various entries based filtered on the basis of a given parameter. This is the insights generator at run time.
5. Forms – Provides UI based functional requirements required by Login/Sign Up.
6. Views – Binding data to the templates and sending http responses.
7. Templates – Define layout of the pages visible to the client.
8. Dashboard – The run time component for the data visualizer. It has two parts the Geographical Heat Map that uses Google Charts and a general data visualizer which uses D3.js

ARCHITECTURE



DEVELOPMENT DECISIONS

WHY DJANGO? – DJANGO ORM

- An object-relational mapper (ORM) is a code library that automates the transfer of data stored in relational databases tables into objects that are more commonly used in application code.
- The Django ORM is an incredibly powerful database tool. It supports multiple databases - MySQL, PostgreSQL, Oracle & SQLite are all supported out-of-the-box assuming you have the relevant Python libraries installed
- *We have tested our application for both SQLite3 and PostgreSQL without any code modifications*.

WHY DJANGO? – DJNAGO FORMS

- You define some fields and how you want the basic validation to work, and Django creates the HTML adds the error messages and cleans the data so you don't get anything unexpected.
- The Django forms framework can even generate and update your database from a database model you create, make your job even easier.
- *We have used this feature in our app's user registration form*

DEVELOPMENT DECISIONS

WHY D3.js

- Using D3.js we separate the data analysis and data visualization components of our web service. Data Analysis is done on the server side with Python and the results are sent to the client side in JSON format which can be used by D3.js library to make visualization independently and without putting the load of graphics rendering on the server.

Google Charts

- We have also used Google Charts. We found that the code we have to write is less verbose as compared to D3 and in a few cases quality of charts rendered by Google Charts is superior to D3 (e.g. Geographical Charts). We might consider moving entirely to Google Charts.

FUTURE IMPROVEMENTS

- Enable the client to add multiple service domains for analysis.
- Add more complex plots.
- Add Machine Learning components to predict future trends.
- Enable client to create custom plots

DESCRIPTION

IMPLEMENTATION DETAILS

We plan to use the following technologies and frameworks to build our software:

- Python/Django for Server side programming
- HTML5, CSS and JavaScript for front end UI
- D3.js & Google Charts for Data Visualization

GITHUB REPOSITORY

Following is the link for the project repository hosted on github:

https://github.com/Pranshu258/DataCloud_Analytics

THANK YOU

Presentation by Akash Waghela & Pranshu Gupta