Employee Management System

Yu Zhao, Xiaojing Wang, Sujeeth Nilakantan

GitHub: https://github.com/zhaoyufrank/CloudComputing-Project

Project Introduction

This is an employee management project. The home page will display all employees, the next web page will show the detailed information of an employee. The user can create a new employee, update an employee's information and delete the employee from the system. The employee information will include name, gender, email, birthday, work department, description, start date, position, and so on.

We will use React.js to build the front end, and use Python with Flask framework to create the backend. The project will be built on the Google Cloud, using lots of cloud computing services, such as NoSQL Firestore database, Cloud Function, Kubernetes Engine.

Service/Application Overview

- Describes how the user(s) will interact with the service

    1. The user (admin) clicks the create button on the home page; the user will be re-directed to the 'create employee' page. The user will input the employee related information and click on the save button; the input details will be saved and stored in the firestore database.
    2. If the user clicks on the edit button on the home page; the user will be re-directed to the 'edit employee' page. The user can modify the existing employee details and click on the save button; the input details will be saved and updated in the firestore database.
    3. If the user clicks on an employee; the user will be re-directed to a new page which will display all the information of that employee.
    4. If the user clicks on the delete button, the application will delete the employee from the front end and the firestore database

- What does the service enable?

    The project can create, list, update, delete and display the details of employees

Components Used

- Which components will you be using?

Python, Flask, React.js, Kubernetes Engine, Pub/Sub, Firestore, dataflow Pipeline, Cloud Functions.

- What is the purpose of each one?

Programming language: Python

Frontend: React.js

Backend: Flask, Cloud Functions

Database: Firestore

Compute service: Kubernetes Engine

Enable services to communicate: Pub/Sub

Data processing service - Dataflow pipeline

Analytics - BigQuery

Architecture

- How is the data flowing through the service?
  A frontend application will capture the user input from HTML page. Upon submission, our frontend code will react to the event. In the meantime, we will use event producers and consumers created by Pub/Sub. Then dataflow Pipeline will process the input data and write the data to firestore. Finally, we can load data into BigQuery for further analysis.
- How will components connect to each other?
  - The front end code will capture the data which the user inputs on HTML webpage and submit them through configured Pub/Sub API.
  - We will create a subscription, and attach a topic to the subscription. So the dataflow pipeline will receive the messages that PubSub sends to this topic.
  - We can use a connector to send records into firestore by data pipeline.
  - For BigQuery, use SQL or GUI to query data from firestore

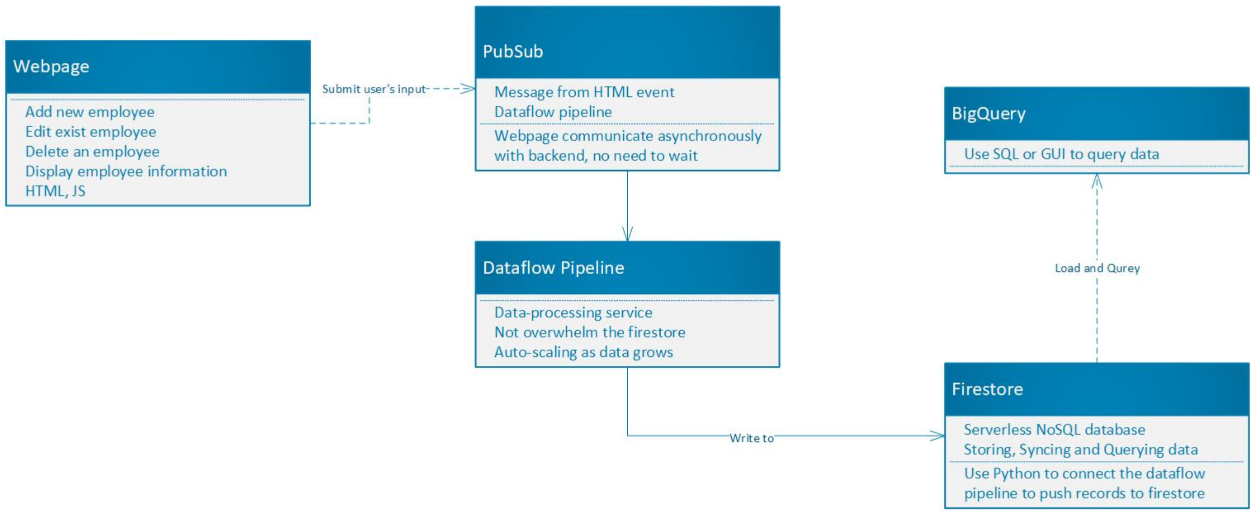
Design

- What language, game framework, etc. are you using?

HTML, React.js, Python, Flask

- What are the major code processes/workflows that will control functionality? And a description of major objects in the design

    For the major code processes/workflows and description of major objects, I drew the following UML to clarify them.


Webpage
Add new employee
Edit exist employee
Delete an employee
Display employee information
HTML, JS
Submit user's input
PubSub
Message from HTML event
Dataflow pipeline
Webpage communicate asynchronously with backend, no need to wait
Dataflow Pipeline
Data-processing service
Not overwhelm the firestore
Auto-scaling as data grows
Write to
BigQuery
Use SQL or GUI to query data
Load and Qurey
Firestore
Serverless NoSQL database
Storing, Syncing and Querying data
Use Python to connect the dataflow pipeline to push records to firestore

Implementation Plan

- What parts of the application and/or design do you plan to implement in what order?

    The order in which we plan to implement the components of the application are

    1. Compute service - Kubernetes Engine
    2. A serverless execution environment - Cloud Functions
    3. A micro web framework – Flask
    4. A cloud database - Firestore
    5. A web page – Front end
    6. Enables services to communicate: Pub/Sub
    7. Dataflow pipeline - Data processing service
    8. Analytics - BigQuery

- Who is responsible for which parts of implementation?

Yu -	Kubernetes Engine, Cloud Functions
Xiaojing -	Flask, Pub/Sub, Data flow pipeline
Sujeeth -	Firestore, Front end, BigQuery

-

● How will you know you are on schedule for finishing and/or a planned milestones timeline?

No.	Task	Due Date
1	Set up the cloud environment	10/16/2021
2	Home page	10/20/2021
3	Create employee page	10/28/2021
4	Mid-point demo	10/28/2021
5	Update employee page	12/02/2021
6	Delete employee function	12/02/2021
7	Test the web service	12/02/2021
8	Final presentation	12/02/2021

Test Plan

● How will you verify that the service game is working correctly?

The user will input/update/delete new employee details on the webpage. Upon submission, if the input details are stored/updated/deleted in the Firestore database accurately and the new employee information is displayed correctly on the webpage, then we can conclude that the web service is working correctly.

Yu Zhao: Project Introduction, Service/Application Overview, Components Used

Xiaojing Wang: Architecture, Design

Sujeeth Nilakantan: Implementation Plan, Test Plan

The team members mentioned above agree to the given project proposal.