# Datavid Cake Tracker

Developed by Firisar Gabriel

Contents

[Datavid Cake Tracker 1](#_Toc169627770)

[1.Introduction 3](#_Toc169627771)

[2.Design 4](#_Toc169627772)

[3.Implementation 5](#_Toc169627773)

[3.1 Setting Up the Development Environment 5](#_Toc169627774)

[3.2 Project Structure 5](#_Toc169627775)

[3.3 Code Walkthrough 6](#_Toc169627776)

[3.4 Frontend 7](#_Toc169627777)

[4.Conclusion 10](#_Toc169627778)

## 1.Introduction

Datavid Cake Tracker is a web-based application designed to assist companies in managing and tracking employee birthdays. Built using Flask, a lightweight Python web framework, and MongoDB, a NoSQL database, this application provides a simple and efficient way to keep track of all employees' birthdays and notify the company in advance about upcoming birthdays.

The main features of the Birthday Tracker include the ability to add new members with their birthdates, view a list of all members sorted by their next birthday, and delete members as needed. The application ensures that only members who are at least 18 years old can be added, maintaining compliance with age-related company policies.

The application comprises several key components:

1. **Web Interface:** The user-friendly web interface allows easy navigation and interaction with the application. Users can add new members through a form, view the list of all members with their details, and delete members if necessary.
2. **Database Management:** MongoDB is used to store member information securely. Each member's details, including their first name, last name, birthdate, country, and city, are stored in the database. The application retrieves and updates this information as needed.
3. **Birthday Calculation:** The application calculates the number of days until each member's next birthday. This helps in planning and celebrating birthdays timely, enhancing employee engagement and satisfaction.

Datavid Cake Tracker is a nice tool for companies that value employee morale and want to foster a culture of recognition and celebration. By automating the tracking of birthdays, the application reduces administrative effort and ensures that no employee's special day is overlooked. Whether you are a small startup or a large corporation, the Birthday Tracker can be a valuable addition to your HR toolkit, making it easier to celebrate your team members and strengthen workplace relationships.

## 2.Design

The architecture of the Datavid Cake Tracker follows a typical Model-View-Controller (MVC) pattern. In this design, MongoDB serves as the model, storing all member information, including first name, last name, birthdate, country, and city. The view is represented by HTML templates rendered by Flask, which use Jinja2 for dynamic content. The controller is implemented through Flask routes that handle HTTP requests and define the application’s logic for adding, displaying, and deleting member records.

The application's data model is centered around the MongoDB collection named members. Each document in this collection contains fields for firstname (the member's first name), lastname (the member's last name), birthdate (the member's birthdate in 'YYYY-MM-DD' format), country (the member's country of residence), and city (the member's city of residence). This structure ensures that all necessary information for each member is stored in a consistent manner.

The backend of the Datavid Cake Tracker is implemented using Flask, a lightweight WSGI web application framework. Flask provides the tools and libraries needed to handle HTTP requests, render HTML templates, and interact with the MongoDB database.

The application includes several key routes. The index route (/) handles both GET and POST requests, displaying all members sorted by their upcoming birthdays and calculating the number of days until the next birthday. It fetches all members from the members collection, formats their birthdates, sorts them using a helper function called next\_birthday, and calculates the days remaining until the next birthday.

The add member route (/add\_member) provides a form for adding new members to the database. It handles GET requests by rendering the form and POST requests by validating the input data, ensuring the member is at least 18 years old, checking for duplicates, and inserting the new member into the members collection if all checks pass.

The delete member route (/<id>/delete/) handles POST requests to delete a member based on their unique identifier. This route uses the member's \_id to remove the corresponding document from the members collection, ensuring that the database remains consistent and up-to-date.

The frontend of the Datavid Cake Tracker is designed using HTML and styled with CSS. It leverages Jinja2 templating for dynamic content rendering based on the data passed from Flask. The main templates are index.html and add\_member.html.

The index.html template displays a list of all members, showing the number of days until the next birthday and including a link to the "Add New Member" form. Each member entry has a delete button, which allows for easy removal of members from the database. The add\_member.html template provides a form for inputting new member details, with fields for first name, last name, birthdate, country, and city. This template ensures that users can easily add new members while providing feedback on any errors or validation issues.

## 3.Implementation

The implementation chapter details the specific steps, code structures, and libraries used to build the Datavid Cake Tracker. This section will guide you through setting up the development environment, understanding the codebase, and deploying the application.

### 3.1 Setting Up the Development Environment

To start developing the Datavid Cake Tracker, you need to set up your development environment. The primary technologies used are Python, Flask, and MongoDB. Ensure you have Python and MongoDB installed on your machine.

1. **Install Flask**: Use pip to install Flask:

pip install flask

1. **Install PyMongo**: PyMongo is the library used to interact with MongoDB:

pip install pymongo

1. **MongoDB Setup**: Start your MongoDB server on the default port (27017). If MongoDB is not already installed, you can download and install it from the [official MongoDB website](https://www.mongodb.com/try/download/community).

### 3.2 Project Structure

The project follows a simple structure to keep the code organized:

datavid-cake-tracker

|-- app.py

|-- templates

| |-- index.html

| |-- add\_member.html

|-- static

| |-- styles.css

* **app.py**: The main application file containing the Flask routes and logic.
* **templates/**: Directory containing HTML templates rendered by Flask.
* **static/**: Directory for static files like CSS.

### 3.3 Code Walkthrough

#### 3.3.1 app.py

The main application logic resides in app.py. It starts by importing necessary modules and initializing the Flask application and MongoDB client.

from flask import Flask, render\_template, request, url\_for, redirect

from pymongo import MongoClient

from bson.objectid import ObjectId

from datetime import datetime

app = Flask(\_\_name\_\_)

client = MongoClient('localhost', 27017)

db = client.flask\_database

members = db.members

##### 3.3.2 Helper Functions

The application includes helper functions to format dates and calculate the next birthday.

def next\_birthday(birthdate):

today = datetime.today()

try:

birthdate = datetime.strptime(birthdate, '%d.%m.%Y')

except (ValueError, TypeError):

return today

birthdate\_this\_year = birthdate.replace(year=today.year)

if birthdate\_this\_year < today:

birthdate\_this\_year = birthdate\_this\_year.replace(year=today.year + 1)

return birthdate\_this\_year

def format\_date(date\_str):

try:

date\_obj = datetime.strptime(date\_str, '%Y-%m-%d')

return date\_obj.strftime('%d.%m.%Y')

except ValueError:

return 'Invalid date'

##### 3.3.3 Routes

The index route (/) handles displaying all members and calculating the days until the next birthday.

@app.route("/", methods=['GET', 'POST'])

def index():

all\_members = list(members.find())

for member in all\_members:

member['birthdate'] = format\_date(member['birthdate'])

all\_members.sort(key=lambda x: next\_birthday(x['birthdate']))

if all\_members:

next\_birthdate = min(next\_birthday(member['birthdate']) for member in all\_members)

days\_until\_next\_birthday = (next\_birthdate - datetime.today()).days

else:

days\_until\_next\_birthday = None

return render\_template('index.html', members=all\_members, days\_until\_next\_birthday=days\_until\_next\_birthday)

The add member route (/add\_member) handles adding new members to the database with input validation.

@app.route("/add\_member", methods=['GET', 'POST'])

def add\_member():

if request.method == "POST":

firstname = request.form['firstname']

lastname = request.form['lastname']

birthdate = request.form['birthdate']

country = request.form['country']

city = request.form['city']

try:

birthdate\_obj = datetime.strptime(birthdate, '%Y-%m-%d')

today = datetime.today()

age = today.year - birthdate\_obj.year - ((today.month, today.day) < (birthdate\_obj.month, birthdate\_obj.day))

if age < 18:

return render\_template('add\_member.html', error="Member must be at least 18 years old.")

except ValueError:

return render\_template('add\_member.html', error="Invalid birthdate format.")

existing\_member = members.find\_one({'firstname': firstname, 'lastname': lastname, 'country': country, 'city': city})

if existing\_member:

return render\_template('add\_member.html', error="A member with the same name and location already exists.")

members.insert\_one({'firstname': firstname, 'lastname': lastname, 'birthdate': birthdate, 'country': country, 'city': city})

return redirect(url\_for('index'))

return render\_template('add\_member.html')

The delete member route (/<id>/delete/) handles removing a member from the database.

@app.post("/<id>/delete/")

def delete(id):

members.delete\_one({"\_id": ObjectId(id)})

return redirect(url\_for('index'))

### 3.4 Frontend

The frontend is built using HTML and styled with CSS. The templates are rendered by Flask and use Jinja2 for dynamic content insertion.

The index.html template displays the list of members and the number of days until the next birthday.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Datavid Cake Tracker</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body>

<h2>Datavid Cake Tracker</h2>

<div class="content">

<div>

<a href="{{ url\_for('add\_member') }}">Add New Member</a>

</div>

{% if days\_until\_next\_birthday is not none %}

<p>There are <i>{{ days\_until\_next\_birthday }}</i> days until the next birthday.</p>

{% endif %}

{% for member in members %}

<div class="member">

<p><strong>{{ member['firstname'] }} {{ member['lastname'] }}</strong> <i>{{ member['birthdate'] }}</i></p>

<p>{{ member['country'] }}, {{ member['city'] }}</p>

<form method="POST" action="{{ url\_for('delete', id=member['\_id']) }}">

<button type="submit" onclick="return confirm('Are you sure you want to delete this entry?')">Delete member</button>

</form>

</div>

{% endfor %}

</div>

</body>

</html>

The add\_member.html template provides a form for adding new members.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Datavid Cake Tracker</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body>

<h2>Datavid Cake Tracker</h2>

<div class="content">

{% if error %}

<p style="color: red;">{{ error }}</p>

{% endif %}

<div>

<a href="{{ url\_for('index') }}">Back to Home</a>

</div>

<form method="POST" style="padding: 20px;">

<p><b><label for="firstname">First Name</label></b></p>

<p><input type="text" name="firstname" id="firstname" required></p>

<p><b><label for="lastname">Last Name</label></b></p>

<p><input type="text" name="lastname" id="lastname" required></p>

<p><b><label for="birthdate">Birth date</label></b></p>

<p><input type="date" name="birthdate" id="birthdate" required></p>

<p><b><label for="country">Country</label></b></p>

<p><input type="text" name="country" id="country" required></p>

<p><b><label for="city">City</label></b></p>

<p><input type="text" name="city" id="city" required></p>

<button type="submit">Submit</button>

</form>

</div>

</body>

</html>

## 4.Conclusion

The Datavid Cake Tracker is an efficient and user-friendly web application designed to manage and track the birthdays of company members. Utilizing Flask for the backend and MongoDB for data storage, the application provides a robust and scalable solution. The clear implementation and intuitive design ensure ease of use and maintenance, making it a valuable tool for any organization.