Module 5 Milestone 4

Hai Nguyen

CS 499 Computer Science Capstone

Professor Brook Goggin

24 Nov 2023

## Databases:

The artifact is a software application titled "ProjectTwoDashboard.ipynb," created on February 19, 2023. This application is part of the "Training and Motivation" project within the International Animal Shelter organization. The purpose of the software is to select specific profiles of dogs for training. The project utilizes Python, MongoDB Compass, and 'Jupyter Notebook' as its key technologies to sort out and identify available dogs from an existing database at the animal shelter. The primary objective is to contribute to the training and motivation of dogs within the shelter environment.

## 2. Justify the inclusion of the artifact in my ePorfolio:

The inclusion of the "ProjectTwoDashboard.ipynb" artifact in my ePortfolio is relevant to its objectives since it aligns with Course Outcome 3, which emphasizes practical solutions in multiple programming languages for real-world problems. The use of Python, MongoDB Compass, and 'Jupyter Notebook' reflects a diverse and modern technology stack. This demonstrates my proficiency in utilizing multiple tools and languages for software development, showcasing versatility in technology adoption.

This artifact stands out as a demonstration of my skills and abilities in the field of algorithm implementation and optimization. The artifact showcases my proficiency in sorting algorithms, with a specific focus on quicksort. This is crucial for a computer science professional as it reflects a solid understanding of fundamental algorithms, which is essential in various software development scenarios.

The artifact shows my ability to sort and identify dogs from an existing database implies proficiency in data handling and manipulation through multiple tools and programming languages. It also reveals my skills in querying databases and extracting relevant information for further processing. Furthermore, the software's functionality to select certain dog profiles for training suggests the application of algorithmic logic. This component showcases my ability to design and implement algorithms to meet specific project requirements. Finally, the use of Python, MongoDB Compass, and 'Jupyter Notebook' highlights my competence in integrating various technologies seamlessly. This integration is crucial for the project's success and underscores my skills in navigating a diverse tech field.

The "ProjectTwoDashboard.ipynb" artifact was improved through the integration of multiple database systems (both SQL and NoSQL databases). The back end was implemented using different programming languages, such as Node.js, highlighting versality in language selection for full-stack application development. In addition, security measures, including data encryption and access controls, were implemented, aligning the project with security best practices. In summary, these enhancements not only demonstrated technical proficiency but also addressed usability and performance considerations, contributing to a more robust and flexible software application.

## 3. Course expectation:

I think the objectives set for the course have been achieved through the enhancements implemented in Module One. Currently, my coverage plans for outcomes remain thorough. The incorporation of the 'ProjectTwoDashboard.ipynb' artifact in my ePortfolio is based on its successful demonstration of my skill in managing databases. This inclusion aligns seamlessly with appropriate course outcomes and underscores specific elements that spotlight my ability in software development.

4. Reflect on the process of enhancing and/or modifying the artifact:

During the process of enhancing the "ProjectTwoDashboard.ipynb" artifact, I gained valuable insights into various aspects of software development. Integrating multiple database systems, including SQL and NoSQL, expanded my understanding of handling diverse data sources. Learning to fetch and combine data from both types of databases enhanced my database management skills. Furthermore, implementing the dashboard's back-end using a different programming language, such as Node.js, broadened my skill set. Implementing security measures, especially data encryption and access controls, presented complexities in ensuring robust protection without compromising functionality. This experience highlighted the importance of language versatility in creating a full-stack application. In addition, balancing usability and performance is critical for delivering a seamless user experience. In summary, the enhancement process of "ProjectTwoDashboard.ipynb" provided a rich learning experience in database integration, multi-language implementation, security considerations, and usability.

The enhancement process of "ProjectTwoDashboard.ipynb" encountered many challenges. Integrating SQL and NoSQL databases posed compatibility problems, taking accurate consideration of data structures and query methods. Implementing a different programming language for the back end introduced complexities in language integration, demanding precise design to ensure smooth communication between Python and Node.js. The implementation of security measures, particularly data encryption and access controls, added another layer of complexity, requiring a delicate balance between robust protection and maintaining functionality.

## Original Code: ProjectTwoDashboard.ipynb

The original code will be attached to the zip file to submit.

“from jupyter\_plotly\_dash import JupyterDash

import dash

import dash\_leaflet as dl

import dash\_core\_components as dcc

import dash\_html\_components as html

import plotly.express as px

import dash\_table

import base64

from dash.dependencies import Input, Output

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from pymongo import MongoClient

from animal\_shelter import AnimalShelter

###########################

# Data Manipulation / Model

###########################

# FIX ME update with your username and password and CRUD Python module name

username = "aacuser"

password = "130411"

shelter = AnimalShelter(username, password)

df = pd.DataFrame.from\_records(shelter.read\_all({}))

image\_filename = 'Grazioso Salvare Logo.png' # replace with your own image

encoded\_image = base64.b64encode(open(image\_filename, 'rb').read())

#########################

# Dashboard Layout / View

#########################

app = JupyterDash('Project Two')

app.layout = html.Div([

html.Div(id='hidden-div', style={'display':'none'}),

html.Center(html.B(html.H1('SNHU CS-340 Dashboard Project 2'))),

html.Hr(),

html.Center(html.Img(src='data:image/png;base64,{}'.format(encoded\_image.decode()))),

html.H5('Select one type of Rescue'),

html.Div(

dcc.Dropdown(

id='filter-type',

options=[

{'label': 'Water Rescue', 'value': 'Water Rescue'},

{'label': 'Mountain Rescue', 'value': 'Mountain or Wildernerss Rescue'},

{'label': 'Disaster Rescue', 'value': 'Disaster or Individual Tracking'},

{'label': 'Reset - default', 'value': 'Reset'}

],

value='Reset', multi=False, clearable=False, style={"width": "50%"}

)),

dash\_table.DataTable(

id='datatable-id',

columns=[

{"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns

],

data=df.to\_dict('records'),

editable=False,

filter\_action="native",

sort\_action="native",

sort\_mode="multi",

column\_selectable=False,

row\_selectable=False,

row\_deletable=False,

selected\_columns=[],

selected\_rows=[],

page\_action="native",

page\_current= 0,

page\_size= 10,

), ……”

## 4.2 Modified Code: dog.js and app.js

• Modified code: Node.js back-end exposes an endpoint /animal that fetches all animals from MongoDB. You can then build your Dash front-end to interact with this API. Provides an endpoint /dog to fetch all dogs from the MongoDB database. Uses the Dog model to define the

• Create RESTful API endpoints in Node.js to communicate with MongoDB, allowing the Dash application to fetch and update data.

• Adjust the current MongoDB operations in your Python code to utilize the API endpoints provided by your Node.js backend.

• Ensure that the Dash application interacts with MongoDB through your Node.js server rather than directly.

• Set up communication between your Dash application and the Node.js backend. Dash can make HTTP requests to the Node.js server to fetch data or send updates.

 This outcome aligns well with the development of a data dashboard using Dash, which involves integrating Python (Dash) for the front-end with MongoDB as a NoSQL database on the back end.

File app.js:

const express = require('express');

const bodyParser = require('body-parser');

const cat = require('cat');

const Dog = require('./models/Dog');

const app = express();

const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

// Connect to MongoDB

cat.connect('mongodb://localhost:27017/animal-shelter', { useNewUrlParser: true, useUnifiedTopology: true });

// Import the AAC CSV file into MongoDB

const csvFilePath = './aac\_shelter\_outcomes.csv';

const csv = require('csvtojson');

csv()

.fromFile(csvFilePath)

.then((jsonArrayObj) => {

Dog.insertMany(jsonArrayObj, (err, dogs) => {

if (err) throw err;

console.log(`${dogs.length} dogs inserted into MongoDB`);

});

});

app.get('/dogs', async (req, res) => {

try {

const dogs = await Dog.find();

res.json(dogs);

} catch (error) {

res.status(500).json({ error: 'Internal Server Error' });

}

});

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

File dog.js:

const cat = require('cat');

const dogSchema = new cat.Schema({

breed: String,

sex\_upon\_outcome: String,

age\_upon\_outcome\_in\_weeks: Number,

});

const Dog = cat.model('Dog', dogSchema);

module.exports = Dog;