# CS 340 Project Two README Nick Burnette

## About the Project/Project Title

*This project is a full-stack application designed for Grazioso Salvare. It works with existing data from animal shelters to identify and categorize available dogs. The front end is designed to allow users a more practical interface without directly using MongoDB.*

## Motivation

*This project aims to help find different breeds good for rescue, water rescue, mountain rescue, locating humans after disaster, law enforcement, etc.*

## Getting Started

*Start by installing import-ipynb (see references for code and syntax), then you are able to import the module. The module uses my authentication and user password.*

## Installation

*Tools needed: jupyter notebook, pip install, MongoDB, Dash Leaflet, Panda*

## Usage

*Use the AnimalShelterClass to perform Create, Read, Update, Delete from CRUD. Perform queries to interact with the AAC database, ‘animals’ collection uploaded to MongoDB.*

### Code Example

*Methods for CRUD shown below:*

# Method to implement the C in CRUD

def create(self, data):

if data is not None:

try:

result = self.database.animals.insert\_one(data) # data should be dictionary

return result.acknowledged #returns True if insertion successful

except Exception as e:

print(f"Error occurred: {e}")

return False #Return false if exception occurred

else:

raise Exception("Nothing to save, because data parameter is empty")

# Create method to implement the R in CRUD.

def read(self, query=None):

#reads documents from 'animals' collection, if no query is made, returns all documents

try:

if query is None:

#no query? returns all documents

result = list(self.database.animals.find())

else:

#have query? returns matching document

result = list(self.database.animals.find(query))

#if no documents are found

if not result:

print("No documents found")

return None

return result

except Exception as e:

raise Exception(f"Error reading documents: {e}")

# Create method to update the database, implement the U in CRUD

def update(self, query, update\_values):

#updates document in the 'animals' collection from a query

try:

if not query or not update\_values:

raise ValueError("Both query and update values must be provided.")

#use update\_one

update\_result = self.database.animals.update\_one(query, {"$set": update\_values})

if update\_result.matched\_count == 0:

print("No matching documents found.")

return None

return {

"matched count": update\_result.matched\_count,

"modified count": update\_result.modified\_count

}

except Exception as e:

raise Exception(f"Error updating documents: {e}")

# Create method to delete from the database, implement the D in CRUDC

def delete(self, query):

#deletes query document in the 'animals' collection

try:

if not query:

raise ValueError("Query must be provided for deletion.")

#delete the document, using 'delete\_one'

delete\_result = self.database.animals.delete\_one(query)

if delete\_result.deleted\_count == 0:

print("No matching documents found, nothing deleted.")

return None

return {"deleted\_count": delete\_result.deleted\_count}

except Exception as e:

raise Exception(f"Error deleting documents: {e}")

### Tests

*Ran each method with the following code to test it:*

*#create instance of the class*

*shelter = AnimalShelter()*

*# Create a test animal data*

*animal\_data = {"name": "Fido",*

*"species": "Dog",*

*"breed": "Labrador",*

*"age": 3,*

*"adopted": False}*

*#call the create method*

*try:*

*shelter.create(animal\_data) # This will insert the animal into the 'animals' collection*

*print("Animal data has been successfully added to the database.") #redundant call for successful insertion*

*except Exception as e:*

*print("Error:", e)*

*'''*

*use the UPDATE method, update age and adopted field on the test animal to '5' and 'True', READ it,*

*then DELETE it*

*'''*

*#pass updated values to variable*

*update\_values = {"age": 5, "adopted": True}*

*#query the updated document to show successful update*

*#create a query variable to pass to read method*

*query = {"name": "Fido", "breed": "Labrador"}*

*#call the update method, add updated values to document*

*result = shelter.update(query, update\_values)*

*# Call the read method to fetch the updated document*

*updated\_document = shelter.read(query)*

*if updated\_document:*

*print("updated\_document:")*

*for doc in updated\_document:*

*print(doc)*

*else:*

*print("Document not found.")*

*#Now call the DELETE method using the same query*

*delete\_result = shelter.delete(query)*

*if delete\_result and delete\_result.get("deleted\_count", 0) > 0:*

*print("Document successfully deleted.")*

*else:*

*print("No Document deleted.")*

*# Verify deletion*

*deleted\_document = shelter.read(query)*

*if deleted\_document:*

*print("Deleted document still exists (unexpected):")*

*for doc in deleted\_document:*

*print(doc)*

*else:*

*print("Document not found (deletion confirmed).")*

### Screenshots

*The module code shown below:*

*A screen shot of a computer

Description automatically generated*

*A screenshot of a computer program

Description automatically generated*

*A screenshot of a computer code

Description automatically generated*

*A python script testing the create and read methods:*

A screenshot of a computer

Description automatically generated

*A python script that imports the module, see below:*

A screenshot of a computer

Description automatically generated

*The module code importing dash, panda, and correcting a previous issue importing the module as a .py file:*

*A screenshot of a computer program

Description automatically generated*

*The dash front end showing imported Grazioso logo, adjusted for proper size, also showing an imported current date and time for custom header, also showing customizable buttons for filtering the animal results (error on top right of screenshot has to do with map data and was not resolved, but dashboard still operates effectively):*

*A screenshot of a computer

Description automatically generated*

*The dash front end split view of pie chart and geolocation map, map updates to selected animal:*

A screenshot of a computer

Description automatically generated

## Contact

Nick Burnette