# Project Two READ ME

## About the Project/Project Title

*Grazioso Salvare is a company that trains dogs for search-and-rescue operations, and requires a method to identify and categorize potential dogs for this purpose. To facilitate the company’s needs, this project is intended to implement functionality and interaction between client and server; specifically, an application that “can work with existing data from the animal shelters to identify and categorize available dogs”.*

**Functionality**

*This project combines the client side, server side, and middleware into full stack development. This project allows for filtered viewing of data, retrieved from the database using a Python module, and shown in an easy-to-use web dashboard. This dashboard comes with company branding, as well as widgets that offer visual representation of the data. The dashboard is also interactive, allowing for filtering options that respond dynamically.*

**Tools Used**

***Database is in MongoDB (***[***https://www.mongodb.com/***](https://www.mongodb.com/)***).***

*MongoDB is a great tool for use in scalable applications and projects emphasizing agile methodologies. It is both document-oriented and NoSQL, which offers great flexibility and stability with many concurrent users. Combined with Python, database applications and modules can be developed quickly and relatively easily.*

***Middleware was created with Python (***[***https://www.python.org/***](https://www.python.org/)***).***

*Combined with built-in functionality to MongoDB, Python is a great choice for programming language. The driver library PyMongo allows for compatibility between dictionaries and lists, allowing for easy manipulation and iteration functionality.*

***Client-side dashboard was created with Jupyter Notebook (***[***https://jupyter.org/***](https://jupyter.org/)***).***

*Jupyter Notebook is part IDE, part data science environment. It supports many languages, and as a web source application, offers ease-of-use and access as a web tool.*

***Maps, graphs, and UI were created with Plotly and Dash (***[***https://plotly.com/***](https://plotly.com/)***)***

*Plotly is a graphing and analytics library that is functional with Python, among many other languages. Dash is a Python framework that is used for building web-based applications. Combined, these tools offer graphing and widgets within a web-based dashboard.*

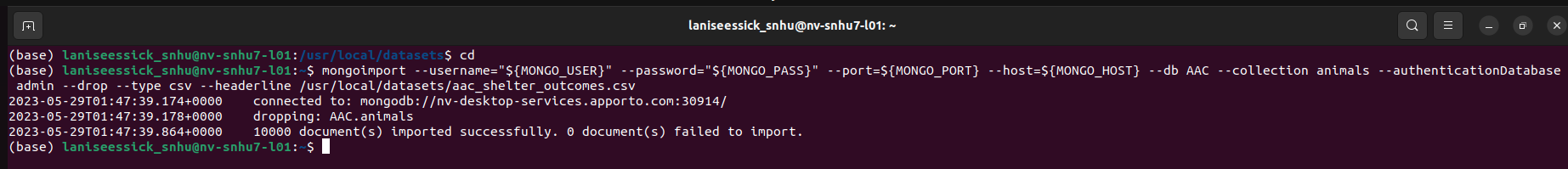
**Steps To Project Completion**

**Database Commands (Mongo Shell and Screenshots): MongoImport and Authentication**

First, we import a CSV file containing the information required for the database. This is done through the following command:

mongoimport --port <PORT#> --db AAC --collection animals ./aac\_shelter\_outcomes.csv

This command will import the .csv file into the database. A successful import is shown below:



We will then create an administrator account and a user account within mongo shell to ensure proper user authentication to this database. First, an admin account is created within admin database, using the following commands:

db.createUser ({

user: "admin",

pwd: passwordPrompt(), // This will create a prompt for entering the password

roles: [{ role: "userAdminAnyDatabase", db: "admin" }, “readAnyDatabase"]

})

We run a similar command to create a user for the “AAC” database:

db.createUser ({

user: "aacuser",

pwd: passwordPrompt(), // This will create a prompt for entering the password

roles: [{ role: "readWrite", db: "AAC" }]

})

Login is done using the following command:

Mongo --authenticationDatabase “<DATABASE>” -u “<USERNAME>” -p

Confirmation of account login is shown below for both “admin” and “aacuser”:

A screenshot of a computer screen

Description automatically generated with medium confidence

**CRUD Python Module: CREATE, READ, UPDATE, DELETE Functionality**

This portion covers the CRUD functionality of the Python module, which will serve as the middleware “glue” for this software. CRUD is an acronym for CREATE, READ, UPDATE, and DELETE, and are the basic operations to manipulate the database and view its contents. The methods for this functionality are shown in screenshots below:

Create:

# method used to create records in animals collection

def create(self, data):

if data is not None:

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

if insert != 0:

return True

else:

return False

else:

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

Read:

# method used to read records in animals collection

def read(self, criteria=None):

if criteria is not None:

data = self.database.animals.find(criteria,{"\_id": False})

for document in data:

print(document)

else:

data = self.database.animals.find({},{"\_id": False})

return data

Update:

# method used to update records in animals collection

def update(self, initial, change):

if initial is not None:

if self.database.animals.count\_documents(initial, limit = 1) != 0:

update\_result = self.database.animals.update\_many(initial, {"$set": change})

result = update\_result.raw\_result

else:

result = "No document was found"

return result

else:

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

Delete:

# method used to delete records in animals collection

def delete(self, remove):

if remove is not None:

if self.database.animals.count\_documents(remove, limit = 1) != 0:

delete\_result = self.database.animals.delete\_many(remove)

result = delete\_result.raw\_result

else:

result = "No document was found"

return result

else:

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

**Dashboard Interactivity**

This portion covers the interactivity of the dashboard, serving as the client-side. The user is able to interact with the dashboard and move around the map, as well as select individual data points within the dashboard. The dashboard also allows for filtering, based on 5 key requirements: Water Rescue, Mountain or Wilderness Rescue, Disaster or Individual Tracking, Elderly 15+, and a reset function, returning all widgets to an unfiltered state. These requirements are shown below:

**Dashboard**A screenshot of a computer

Description automatically generated with medium confidence

**Challenges/Potential Issues**

The biggest challenge I faced making this project was getting the data to populate in the dashboard, and I believe this in retrospect caused my callback errors as well.

Data populates in Jupyter, even when using filter options in dashboard.

See image:

A screenshot of a computer

Description automatically generated with medium confidence

Unfortunately, I was not able to resolve this issue or find a solution.

Other issues you may face is authentication:

If a user runs into issues with authorization when running the dashboard, check these possible solutions:

* Make sure the username and password exist
* Make sure the username and password are correct
* Check the port and host information
* Make sure the dashboard and/or database process closed successfully. If not seek out the process id and stop the process

## Contact

LaNise Essick

lanise.essick@snhu.edu

CS-340 Client/Server Development