# CS 340 README Template

*Use this template to complete your README file. When completing the template, keep the headings as they are so that your document has a clear organization. Remove the italicized prompt text after you have completed each section for a polished final document.*

## About the Project/Project Title

*Grazioso Salvare’s Search-and-Rescue Training Canidates*

## Motivation

*This web application aims to offer a user-friendly interface for filtering data stored in a MongoDB database, helping users find the ideal combinations of breed, sex, and age for training rescue dogs. The project includes a database, an API, and a dynamic dashboard.*

**The purpose of the CRUD Python module**

*The CRUD Python module is designed to streamline and simplify database operations by providing a standardized way to create, retrieve, update, and delete data. It abstracts the complexities of database interactions, allowing developers to perform these operations without writing extensive SQL queries, thereby enhancing code readability and maintainability. The module accelerates development by offering predefined functions for common tasks, making it ideal for rapid prototyping and iterative development. Additionally, it ensures consistent and reliable database transactions through built-in error handling and data validation, ultimately facilitating efficient and effective data management in applications* (GeeksforGeeks, 2023).

## Getting Started

*To get a local copy up and running, download the project and the AnimalShelter data. Upload the csv file in the terminal. Run the project with the appropriate data to add or read.*

**Tools Used and Rationale For Why They Were Used**

***MongoDB***

*MongoDB was chosen as the model component of development due to its ease of setup from a CSV document, Python-friendly interface, and flexible schema design. These features simplify data import and interaction, allowing for more efficient development workflows. MongoDB also offers straightforward CRUD operations and a powerful aggregation framework, which reduces the complexity of data manipulation. Additionally, its scalability and performance capabilities make it a robust choice for handling large volumes of data and future growth, aligning well with Python's syntax and programming style (GeeksforGeeks, 2023).*

***Dash***

*For the view and controller structure of the web application, the Dash framework was used. Dash is a powerful Python framework that integrates with Plotly for data visualization, allowing developers to build interactive and data-driven web applications entirely in Python, thus simplifying the development process and enhancing the user experience (Dashboards - Atlas Charts, n.d.).*

## Installation

*AnimalShelter data csv file.*

*Linux based terminal*

*Project Two folder*

## Usage

*Use this space to show useful examples of how your project works and how it can be used. Be sure to include examples of your code, tests, and screenshots.*

### Code Example

*# change animal\_shelter and AnimalShelter to match your CRUD Python module file name and class name*

*from project2\_animal\_shelter import AnimalShelter*

*# Data Manipulation / Model*

*username = "aacuser"*

*password = "SNHU1234"*

*# Connect to database via CRUD Module*

*db = AnimalShelter(username, password)*

*# Dashboard Layout / View*

*app = JupyterDash(\_\_name\_\_)*

*image\_filename = 'GraziosoSalvareLogo.png' # replace with your own image*

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

*app.layout = html.Div([*

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

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

*html.Center(html.B(html.H1('CS-340 Dashboard'))),*

*html.Center(html.B(html.H1('Fred Wahab, June 2024'))),*

*html.Hr(),*

*html.Div(*

*dcc.RadioItems(*

*id='filter-type',*

*options=[*

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

*{'label': 'Mountain/Wilderness Rescue', 'value': 'mountain-wild'},*

*{'label': 'Disaster Rescue/Individual Tracking', 'value': 'disaster'},*

*{'label': 'Reset - returns unfiltered state', 'value': 'reset'}*

*],*

*value='reset',*

*)*

*),*

*html.Hr(),*

*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'),*

*#FIXME: Set up the features for your interactive data table to make it user-friendly for your client*

*#If you completed the Module Six Assignment, you can copy in the code you created here*

*editable=False,*

*filter\_action="native",*

*sort\_action="native",*

*sort\_mode="multi",*

*column\_selectable=False,*

*row\_selectable="multi",*

*row\_deletable=False,*

*selected\_columns=[],*

*selected\_rows=[],*

*page\_action="native",*

*page\_current=0,*

*page\_size=10,*

*),*

*html.Br(),*

*html.Hr(),*

*# Interaction Between Components / Controller*

*@app.callback([Output('datatable-id','data'),*

*Output('datatable-id','columns')],*

*[Input('filter-type', 'value')])*

*def update\_dashboard(filter\_type):*

*# Water*

*if filter\_type == 'water':*

*df = pd.DataFrame.from\_records(animals.read({*

*"animal\_type": "Dog",*

*"breed": {"$in": ["Labrador Retriever Mix","Chesapeake Bay Retriever", "Newfoundland"*

*]},*

*"sex\_upon\_outcome": "Intact Female",*

*"age\_upon\_outcome\_in\_weeks": {"$gte":26.0, "$lte":156.0}*

*}))*

*# mountain-wild*

*elif filter\_type == 'mountain-wild':*

*df = pd.DataFrame.from\_records(animals.read({*

*"animal\_type": "Dog",*

*"breed": {"$in": ["German Shepard","Alaskan Malamute","Old English Sheepdog",*

*"Siberian Husky", "Rottweiler"*

*]},*

*"sex\_upon\_outcome": "Intact Male",*

*"age\_upon\_outcome\_in\_weeks": {"$gte":26.0, "$lte":156.0}*

*}))*

*# disaster*

*elif filter\_type == 'disaster':*

*df = pd.DataFrame.from\_records(animals.read({*

*"animal\_type": "Dog",*

*"breed": {"$in": ["Doberman Pinscher","German Shepard","Golden Retriever",*

*"Bloodhound","Rottweiler"*

*]},*

*"sex\_upon\_outcome": "Intact Male",*

*"age\_upon\_outcome\_in\_weeks": {"$gte":20.0, "$lte":300.0}*

*}))*

*# reset*

*else:*

*df = pd.DataFrame.from\_records(animals.read({}))*

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

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

*return (data,columns)*

*def update\_graphs(viewData):*

*###FIX ME ####*

*# add code for chart of your choice (e.g. pie chart) #*

*dff = pd.DataFrame.from\_dict(viewData)*

*names = dff['breed'].value\_counts().keys().tolist()*

*values = dff['breed'].value\_counts().tolist()*

*return [*

*dcc.Graph(*

*figure = px.pie(*

*data\_frame=dff,*

*values = values,*

*names = names,*

*color\_discrete\_sequence=px.colors.sequential.RdBu,*

*width=800,*

*height=500*

*)*

*)*

**Tests**

*The project’s implementations can be tested using the ProjectTwoDashboard.ipynb file*

### Screenshots

*The biggest challenge I have tried to overcome is receiving connection errors when trying to run the project. I have attempted to add privileges, create new users, verified the csv data in the terminal, rebuild the .py file and nothing works. Affecting multiple assignments.*

A screenshot of a computer

Description automatically generated

*The only noticeable error is when Jupyter initially starts, the terminal shows these errors.*

A screenshot of a computer

Description automatically generated

*All other errors (syntax, etc.) were resolved.*

A screenshot of a computer

Description automatically generated

## Contact

Your name: Fred Wahab

**References**

*Dashboards - Atlas Charts*. (n.d.). https://www.mongodb.com/docs/charts/dashboards/

GeeksforGeeks. (2023, December 1). *MongoDB Advantages Disadvantages*. GeeksforGeeks. https://www.geeksforgeeks.org/mongodb-advantages-disadvantages/