Competencies

In this project, you will demonstrate your mastery of the following competencies:

- Apply database systems concepts and principles in the development of a client/server application
- Develop client-side code that interfaces with databases

Scenario

You work for Global Rain, a software engineering company that specializes in custom software design and development. Your team has been assigned to work on a project for an innovative international rescue-animal training company, Grazioso Salvare. You have been made the lead developer on this project.

As part of its work, Grazioso Salvare identifies dogs that are good candidates for search-and-rescue training. When trained, these dogs are able to find and help to rescue humans or other animals, often in life-threatening conditions. To help identify dogs for training, Grazioso Salvare has reached an agreement with a non-profit agency that operates five animal shelters in the region around Austin, Texas. This non-profit agency will provide Grazioso Salvare with data from their shelters.

In meeting with the client, you have discovered that they look for certain profiles in dogs to train. For example, search-and-rescue training is generally more effective for dogs that are no more than two years old. Additionally, different breeds of dogs are proficient at different types of rescue, such as water rescue, mountain or wilderness rescue, locating humans after a disaster, or finding a specific human by tracking their scent.

Grazioso Salvare is seeking a software application that can work with existing data from the animal shelters to identify and categorize available dogs. Global Rain has contracted for a full stack development of this application, including a database and a client-facing web application dashboard through which users at Grazioso Salvare will access the database.

In the initial phases of this development, you developed a database and a Python module enabling CRUD functionality for MongoDB. For Project Two, you will complete the development of this project by coding the dashboard and the database interface logic. This will include dashboard attributes. The dashboard must be a user-friendly, intuitive interface that will reduce user errors and training time.

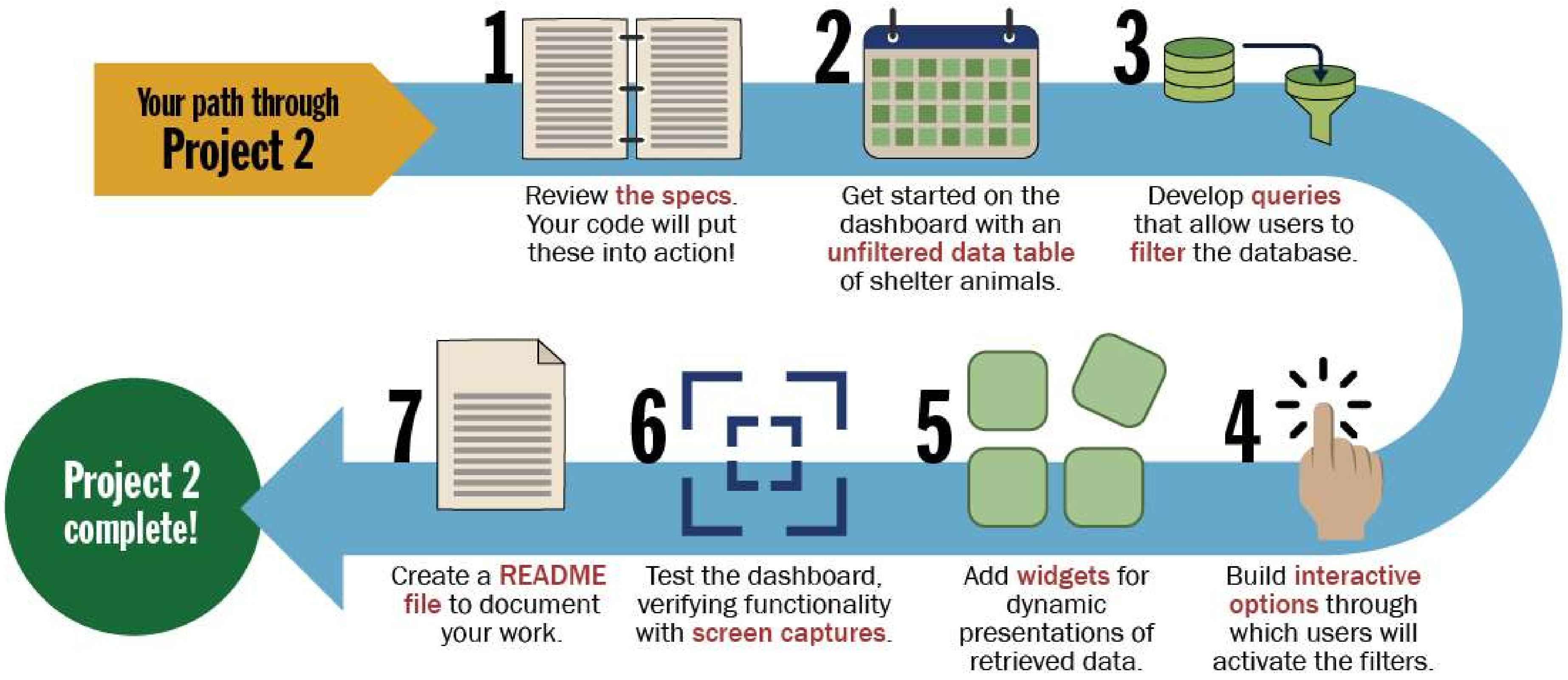
Additionally, Grazioso Salvare has requested that the code for this project be open source and accessible on GitHub, so that it may be used and adapted by similar organizations. To that end, they have asked that you also create a README file that documents and provides instructions for reproducing the project.

Directions

Web Application Dashboard (Python Code)

After successful completion of previous milestones and Project One, you have created a database and established successful CRUD routines in Python for MongoDB. For Project Two, you will now create a fully functional MongoDB dashboard. This will allow the client, Grazioso Salvare, to interact with and visualize the database.

To create the dashboard, you will create the different components in Python code. Dashboard web applications lend well to the MVC design pattern. Using this design pattern, the model is contained and accessed in MongoDB, the views are dashboard widgets, and the controller uses your CRUD Python module for queries as part of the interaction between components. You will also be required to test and deploy the dashboard, taking screenshots to show that each of the components executes successfully.


Your path through
Project 2
1
Review the specs. Your code will put these into action!
2
Get started on the dashboard with an unfiltered data table of shelter animals.
3
Develop queries that allow users to filter the database.
4
Build interactive options through which users will activate the filters.
5
Add widgets for dynamic presentations of retrieved data.
6
Test the dashboard, verifying functionality with screen captures.
7
Create a README file to document your work.
Project 2 complete!

1. Before developing the Python code for the dashboard, be sure to review the Dashboard Specifications Document provided by the UI/UX developer at Global Rain. This document is located in the Supporting Materials section and will provide you with examples of the different dashboard widgets you will create:
    - Interactive options to filter the Austin Animal Center Outcomes data set
    - A data table which dynamically responds to the filtering options
    - A geolocation chart and a second chart of your choice that dynamically respond to the filtering options

    In addition to the widgets, you have been asked to include the **Grazioso Salvare logo and a unique identifier** containing your name somewhere on the dashboard. A high-resolution copy of the logo is included in the Supporting Materials section.

2. Next, you will begin developing the Python code for your dashboard. Starter code is contained in the ProjectTwoDashboard.ipynb file, linked in the What to Submit section. Start by creating a **data table** on the dashboard which shows an *unfiltered* view of the Austin Animal Center Outcomes data set. To populate the data onto your table, **utilize your previous CRUD Python module** from Project One to run a “retrieve all” query and bring in the data from MongoDB.

   Tip: Be sure to consider your client when creating the interactive data table. Consider optional features that will make the table easier to use, such as limiting the number of rows displayed, enabling pagination (advanced), enabling sorting, and so on.

   Note: If you completed the Module Six Milestone, you have already completed this step. Copy your code for the data table into the ProjectTwoDashboard.ipynb file.
3. Next, you will make sure that the dashboard filter options can properly retrieve data from the database. Start by **developing database queries that match the required filter functionality**. Refer to the Rescue Type and Preferred Dog Breeds Table, located in the Dashboard Specifications Document, to help you construct these queries.

   Note: Be sure to **utilize your previous CRUD Python module** (a PY file) from Project One to develop these database queries. You will need to hard code in the username/password for the “aacuser” account as part of the CRUD Python module class instantiation.
4. You must develop the controller pieces to **create interactive options that allow for the selection of data based on your filtering functions** (such as radio items or drop-downs). Develop these pieces in your IPYNB file, and be sure to import and use your CRUD Python module queries from Step 3. These interactive options will enable the **control of other dashboard widgets**, such as the data table and charts.

   Tip: You may choose any interactive option that you wish, such as radio items, a drop-down menu, and so on, as long as the client is able to intuitively use the interactive option to filter the data. Refer to the Dash Core Components reading from the module resources to help you set up these options.
5. Next, you must modify or create the dashboard widgets that receive input from the interactive options and present those dynamic updates to the client. Be sure to modify or create these widgets in your IPYNB file. Specifically, you must do the following:
   - Modify the data table you created in Step 2 so that it is **an interactive data table that responds to input** from the interactive options.
   - **Create charts that display data in response to updates** from the data table. As outlined in the Dashboard Specifications Document, you are required to create, at minimum, a geolocation chart and a second chart of your choice.

   Note: If you completed the Module Six Milestone, you have already begun work on this step by creating the geolocation chart. Copy your code into the ProjectTwoDashboard.ipynb file. You will need to make sure that this chart receives updates from the interactive options.
6. Finally, after developing all of your code, you must test and deploy the dashboard to make sure that all of your components work. To complete this step, run your IPYNB file. You must either **take screenshots or create a screencast of your dashboard and widget functionality**. Each of your screenshots or your screencast should contain the **Grazioso Salvare logo** and your **unique identifier**. Your screenshots or screencast must show the following:
   - The starting state of your dashboard, which should include your widgets for the **interactive options to filter data (such as radio items or drop-downs), the interactive data table, and the charts**
   - Executions of your dashboard, showing the widgets after **each** of the following data filters has been applied (four screenshots total):
     - Water Rescue
     - Mountain or Wilderness Rescue
     - Disaster or Individual Tracking
     - Reset (returns all widgets to their original, unfiltered state)

   You will include all of these screenshots, or your screencast, in your README file when describing the functionality of your project. These screenshots are **required** as they demonstrate proof of your dashboard's functionality.

README File
Grazioso Salvare has requested documentation to accompany the code for your dashboard. This will ensure that they are able to understand the work that was completed and more easily maintain the code for this project.

You have been asked to **create a README file that documents the project and includes instructions for reproducing the project**. Be sure to address all of the following areas:

- **Describe the required functionality** of the project. Include the screenshots or screencast taken while testing and deploying your dashboard (Step 6) as proof that you have achieved the required functionality.
- **Describe the tools used to achieve this functionality and a rationale for why these tools were used.**
    - Be sure to explain why MongoDB was used as the model component of the development, including what specific qualities or capabilities it provides for interfacing with Python.
    - Be sure to explain the Dash framework that provides the view and controller structure for the web application.
    - Be sure to include links to any resources or software applications that were accessed or used.
- **Explain the steps that were taken to complete the project.**
- **Identify any challenges that were encountered and explain how those challenges were overcome.**

What to Submit

To complete this project, you must submit the following:

Web Application Dashboard (Python Code)
Submit a zipped folder containing all of the code for your dashboard. The zipped folder should include your completed ProjectTwoDashboard.ipynb file containing the source code for your dashboard. Be sure to also include the code for the CRUD Python module (PY file) that you originally developed in Project One. All code files should follow industry standard best practices, including well-commented code.

README
Your submission should be a Word document that documents the project and provides instructions for reproducing it. In your README file, be sure to include all required screenshots (or your screencast) to demonstrate the functionality of your dashboard.

Supporting Materials

The following resource(s) may help support your work on the project:

Reading: CS 340 Dashboard Specifications Document
This document was created by your UI/UX developer and details the necessary functionality of your dashboard, as well as providing you with a prototype of the dashboard layout. This document also contains the Rescue Type and Preferred Dog Breeds table, which aligns dog breeds with their use in different types of rescue. Use this table to help structure the queries for your dashboard filtering options.

Image: Grazioso Salvare Logo
This high-resolution PNG file contains the Grazioso Salvare logo. Be sure to include this logo as part of your dashboard to ensure that the application is properly branded.

Data Set: Austin Animal Center Outcomes
Grazioso Salvare has provided you with this sample data set (CSV file) of animal center outcomes. This will become the basis of your database and can be used to test the functionality of your code. This data set has been modified for the purposes of this project. Specifically, the following columns have been added: location_lat (latitude), location_long (longitude), and age_upon_outcome_in_weeks (the age of the animal, given in weeks).

Reference: Austin Animal Center. (2020). *Austin Animal Center Outcomes* [Data set]. City of Austin, Texas Open Data Portal. https://doi.org/10.26000/025.000001

Reading: CS 340 Jupyter Notebook in Apporto (Virtual Lab) Tutorial
This tutorial will help you navigate the technology you will be using in this course. You will learn how to get into the Jupyter Notebook via the Virtual Lab (Apporto), as well as how to complete, save, and download your work.

Reading: CS 340 Mongo in Apporto (Virtual Lab) Tutorial
This tutorial will help you navigate the different Mongo tools needed for your development.

Textbook: Head First Python
This Shapiro Library textbook was designed to teach the Python programming language. Refer to this resource if you need a refresher on any Python syntax as you develop your code.

Reading: Style Guide for Python Code
Refer to this style guide when developing your Python code for this project. It is important that your code follows industry standard best practices, such as including clear variable names, exception handling, and in-line comments throughout your code.

Reading: Make a README
This reading describes the purpose behind README files, and will help you keep in mind the purpose and intended audience for your README file. You are not required to use the same sections as suggested in this reading. As a note, the examples in this article use the MD format, which is a common format for README files on GitHub. You have been asked to submit your README file as a Word document for this project.

Project Two Rubric

Criteria	Exemplary (100%)	Proficient (85%)	Needs Improvement (55%)	Not Evident (0%)	Value
Dashboard (Python Code): CRUD Python Module Queries for Filtering	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or efficient manner	Develops database queries that match the required filter functionality and utilize the previous CRUD Python module for direct MongoDB access	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include database queries that match the required filter functionality and utilize the previous CRUD Python module for direct MongoDB access	Does not attempt criterion	15

Criteria	Exemplary (100%)	Proficient (85%)	Needs Improvement (55%)	Not Evident (0%)	Value
Dashboard (Python Code and Screenshots): Interactive Filtering Options	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or efficient manner	Creates interactive options that allow for the selection of data based on filtering functions and the control of other dashboard widgets, and demonstrates successful executions by providing screenshots with a unique identifier	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include lack of interactive options for filtering data and other dashboard components, or lack of screenshots demonstrating successful completion of work	Does not attempt criterion	15
Dashboard (Python Code and Screenshots): Interactive Data Table	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or efficient manner	Creates an interactive data table that responds to input from other controls, and demonstrates successful executions by providing screenshots with a unique identifier	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include an interactive data table that does not respond to input from other controls, or lack of screenshots demonstrating successful completion of work	Does not attempt criterion	20
Dashboard (Python Code and Screenshots): Charts	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or efficient manner	Creates charts to display data in response to inputs from other controls, and demonstrates successful executions by providing screenshots with a unique identifier	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include charts not displaying data in response to input from controls, or lack of screenshots demonstrating successful completion of work	Does not attempt criterion	20

Criteria	Exemplary (100%)	Proficient (85%)	Needs Improvement (55%)	Not Evident (0%)	Value
README: Documenting the Project	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner	Creates a README file that documents the project by describing the required functionality, tools used to achieve this functionality, and a rationale for why these particular tools were used	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include a README file that does not document the required functionality and rationale for the particular tools used	Does not attempt criterion	15
README: Reproduction Instructions	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner	Creates instructions for reproducing the project by explaining steps that were taken to complete the project, any challenges that were encountered, and how those challenges were overcome	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include lack of instructions for reproducing and completing the project or explanations of how challenges were encountered and overcome	Does not attempt criterion	10
Articulation of Response	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner	Clearly conveys meaning with correct grammar, sentence structure, and spelling, demonstrating an understanding of audience and purpose	Shows progress toward proficiency, but with errors in grammar, sentence structure, and spelling, negatively impacting readability	Submission has critical errors in grammar, sentence structure, and spelling, preventing understanding of ideas	5
				Total:	100%