**Professional Self-Assessment and Project Introduction**

**Revised: 20 Aug 2022**

**Professional Self-Assessment**

This Professional Self-Assessment serves as a summation of my experiences navigating through the Computer Science program at SNHU and introduces the artifacts chosen for my final project in the Capstone class of the SNHU Computer Science program.

**Program Reflection**

Completing my coursework throughout the Computer Science Program over the last four years and developing an ePortfolio has helped to showcase my strengths and shape my professional goals and values and helped me to be more employable in the computer science field.

Learning to create effective security policies that implement structures that enforce coding standards and best practices, as well as how the Software Development Life Cycle (SDLC) functions has been instrumental in my understanding of team environment collaboration. Understanding how each team member has their own classification and assigned duties, how they all link together to create solid planning, creating, testing and deployment stages was beneficial. understanding how everyone had their own assignment or branch off the same project yet linked to the team for peer code review and testing, then finally submitted coding from their branches into the main structure for further development. Understanding these concepts and being able to employ them will make me a more effective team member in the future. For example, the CS-310 Collaboration and Group Project course introduced me to the GIT process and understanding version control, another definition of team collaboration using centralized project storage, with all members of a team accessing the latest version and merging it into their project branches, making the process seamless.

Another concept involving SDLC, communicating with stakeholders is an integral part of the process for a successful project. As learned throughout this program, not gaining transparency and insight with the client or stakeholders will lead to process delay and possibly failure. Keeping an open line of communication early in the process will clarify client requirements, work out any misunderstandings or gaps in communication, as well as minimize rehashing and altering development late in the cycle. After all, the clients are the ones you are creating these projects for, so it is essential to understand one another’s mindset.

Multiple classes throughout this program involved data structure and algorithm creation, as these are the building blocks for more advanced code to grow from. Understanding how these are associated with each other and having the ability to express this through code is an essential component of the computer science program. Most programs incorporate these concepts and would cease to function without them. The concepts of databases and software engineering were found to be the most interesting, as they are the most visual with interactive GUI and understanding all that goes into ensuring a database program will function properly. Included in this would be the addition of error handling and security measures to ensure minimal issues. Security is a cornerstone of any project development to ensure no outside or inside forces gain unauthorized access to privileged data. This security can come in many shapes, to include multi factor authorization protocols, to physical security policies that companies can enforce to ensure data integrity.

In summary, I was introduced to Python in CS-200, CS-250 taught about Scrum, and I learned testing and its importance in CS-310. I learned how to incorporate Discrete Math, Stats, Calculus, Physics, and Linear Algebra into computer science. I gained knowledge on databases and analyzing data. I learned about computational graphics and design. I now understand how to use Python, C++, Java, and other useful applications. All these skills learned with be essential when utilizing them in my future endeavors.

**Final Portfolio Summary**

For this project, it was important to include artifacts that addressed the following topics: collaborating in a team environment, communicating to stakeholders, data structures and algorithms, software engineering and database, and security. I chose three artifacts that made the most educational impact on me. I selected projects that I really believe I would continue to use in my professional life after the requirements for this class were completed. From software design and testing to base algorithm coding for working data structures, and the creation of user-friendly databases centered on the client requirements, all these artifacts fit together to create a fully rounded demonstration of skills and abilities learned throughout this program.

In summary, through the enhancement of three key artifacts, I achieved the following required course outcomes:

1. I employed strategies for building collaborative environments that enabled diverse audiences to support organizational decision making in the field of computer science. **This was achieved in my portfolio in Artifacts 2 and 3.** Both of these artifacts revolved around visual dashboards that were initially unresponsive and allowed for no interactivity. Enhancements allowed for expanded interactivity and could meet the requirement of diverse clients with changing needs after the code was enhanced to allow for simple alteration and additions with minimal revisions needed to ensure the button and query functionality was sufficient.
2. I designed, developed, and delivered professional-quality oral, written, and visual communication that were coherent, technically sound, and appropriately adapted to specific audiences and contexts. **This was achieved in my portfolio in Artifacts 1, 2, and 3** using descriptive coding comments that would allow future developers reviewing the code to understand and be able to add to or enhance. In all code enhancement, there was no junk code, unused or outdated comment and code section that would confuse developers trying to understand its purpose.
3. I designed and evaluated computing solutions that solved a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trad-offs involved in design choices. **This was achieved in my portfolio in Artifact 2** through the following enhancements: I enhanced the efficiency and complexity of an API that displays working button controls, frame and panel attributes, and image and text selection. This involved using multiple data structures to store image and sizing data, as well as associated descriptive text. These have associated algorithms which employ postfix increment and decrement operators linked to button controls that in ultimately control which data and image set to display. Design choices remained code flexible, allowing for adjustments to color, text and imagery determined by client demands. Defensive programming techniques were used to ensure all variables were assigned, and all loops operated on the correct data for desired results. This demonstrated my skills with algorithms and data structures, as well as my ability to effectively use Javascript.
4. I demonstrated an ability to use well founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals. **This was achieved in my portfolio in Artifact 1**, where constructors were created and enhanced to allow for accurate query results and ensured data integrity was followed by integrating test methods to validate or determine illegal arguments.
5. I developed a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources by completing specific enhancements. **This was achieved in my portfolio in Artifact 3** through the following enhancements: I enhanced a mongoDB based database written in Python that can interface with a Node JS application. This application now provides a user-friendly interface that pulls data from a central .csv file, with the objective of helping a client identify good dog candidates for search and rescue training through a dynamic dashboard web application that can be run either with MongoDB or Jupyter Notebook. I incorporated what I learned about mongoDB and added in enhanced functionality to the password interface for added access security, and expanded coding for data creation, alteration, and deletion into an existing data structure. This incorporated additional error notifications and loops to enhance error handling and demonstrated my skills with databases, as well as my ability to use Python. **This was also achieved through Artifact 1**, where it centered around testing automated coding in Javascript with adequate error handling through JUnit tests. To accomplish this, I applied additional testing methods to include assertEquals, assertThrows and assertUpdate, as well as added constructor variations that effectively test variable combinations entered by the user.

**Written Narratives:**

**Artifact 1: Software Engineering and Design Written Narrative**

For Artifact One in Category One: Software Engineering/Design, I selected work done in CS320 – Software Testing, Automation and Quality Assurance, which focuses on locating and resolving software security vulnerabilities by creating secure code and testing procedures to locate issues before code implementation. My enhancement plan resulted in expanding the complexity of my Unit Test files created in Java and edited and implemented using Visual Studio, Terminal and XCode. In addition, enhancements were made to the associated ContactTest.java and ContactServiceTest.java files linked to their respective java files. This demonstrated skills learned in altering Java code to be more secure, as well as in the creation of efficient test files, created specifically for their respective java files for testing.

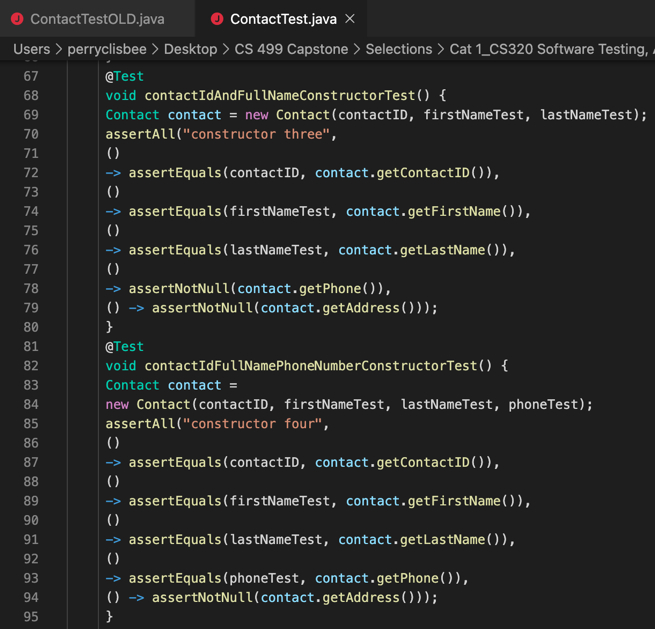
The artifact includes a ContactTest.java file that implements JUnit methods to test the attributes of the Contact class. I created tests that verified the ID is not null, updateable, and less than or equal to 10 characters. Tests were created to determine a potential client’s needs for other variables such as firstName, lastName, phoneNumber and address. These JUnit tests were adjusted to be used with any additional parameters needed to validate the Contact class data.

The ContactTest.java file and its associated Contact.java file were included in the ePorfolio as the test file code displayed the test variations that could be employed to effectively test variable combinations entered by the user. The test file includes specific constructors that tested to ensure parameters for a new contact were asserted to have data, or a NotNull status. The first one tested that all necessary variables were not null. The next one tested to make sure the parameter entered matched the contactID, and that all other variables were NotNull. This continues with testing for assertEquals() for both contactID and firstNameTest. I added several other assert variations such as assert.Update and assertThrows() and assertEquals() methods to validate or notify of an illegal argument.

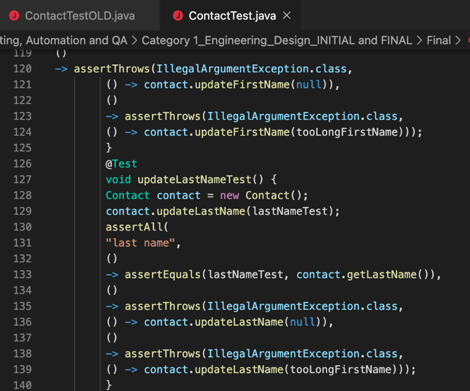
The enhancements met my goals for overall enhancement as it added JUnit tests that covered a wide range of possible outcomes, to ensure minimal error on the backend. Additional code documentation was added that added clear description for each code segment. When modifying the artifact, I did a lot of code cross referencing online to better understand the purpose of the different assert methods, and how to implement them. I used available software such as XCode to view and alter as necessary.

***Artifact I Code Enhancements***

Text

Description automatically generatedThese display constructor tests adding additional assertEquals() parameters 

These three sections below of added code include assertThrows() methods pinpointing an error with an Illegal Argument notification under conditions. Documentation was added for each @test to ensure other developers will understand what was attempted here.

**Text

Description automatically generated**

**Text

Description automatically generated**

**Artifact II: Algorithms and Data Structures Written Narrative**

For Artifact Two in Category Two: Algorithms and Data Structures, I selected work done in CS250 – Software Development LifeCycle (SDLC), which focused on the Agile team roles and how their teamwork resulted in an overall rounded functional Java application where the code included added functionality and imagery for a better user experience. My enhancement plan included expanding the current API complexity of my TopFiveTravelDestination.java and SlideShow.java files with added resource folder imagery to give the user a better visual experience viewing destination information through a simple GUI that is user friendly. This was done through the enhancement of current Java code to include better code documentation to elaborate on code functionality resulting in minimal errors and a fluid transition between screens. This demonstrated knowledge and skills gained in the use of Java coding to create simple applications that the user can easily manipulate but not corrupt. Code was altered and tested using Java IDE, XCode application and Terminal applications for Mac.

The artifact is a compilation of five vacation locations displaying an image along with a description that is supposed to entice the client to go there. The artifact includes a TopFiveDestinationList.java and SlideShow.Java file that are two variations on the same project, that implements JFrames, JLabels and JPanels along with Button Controls to assist in maneuver.

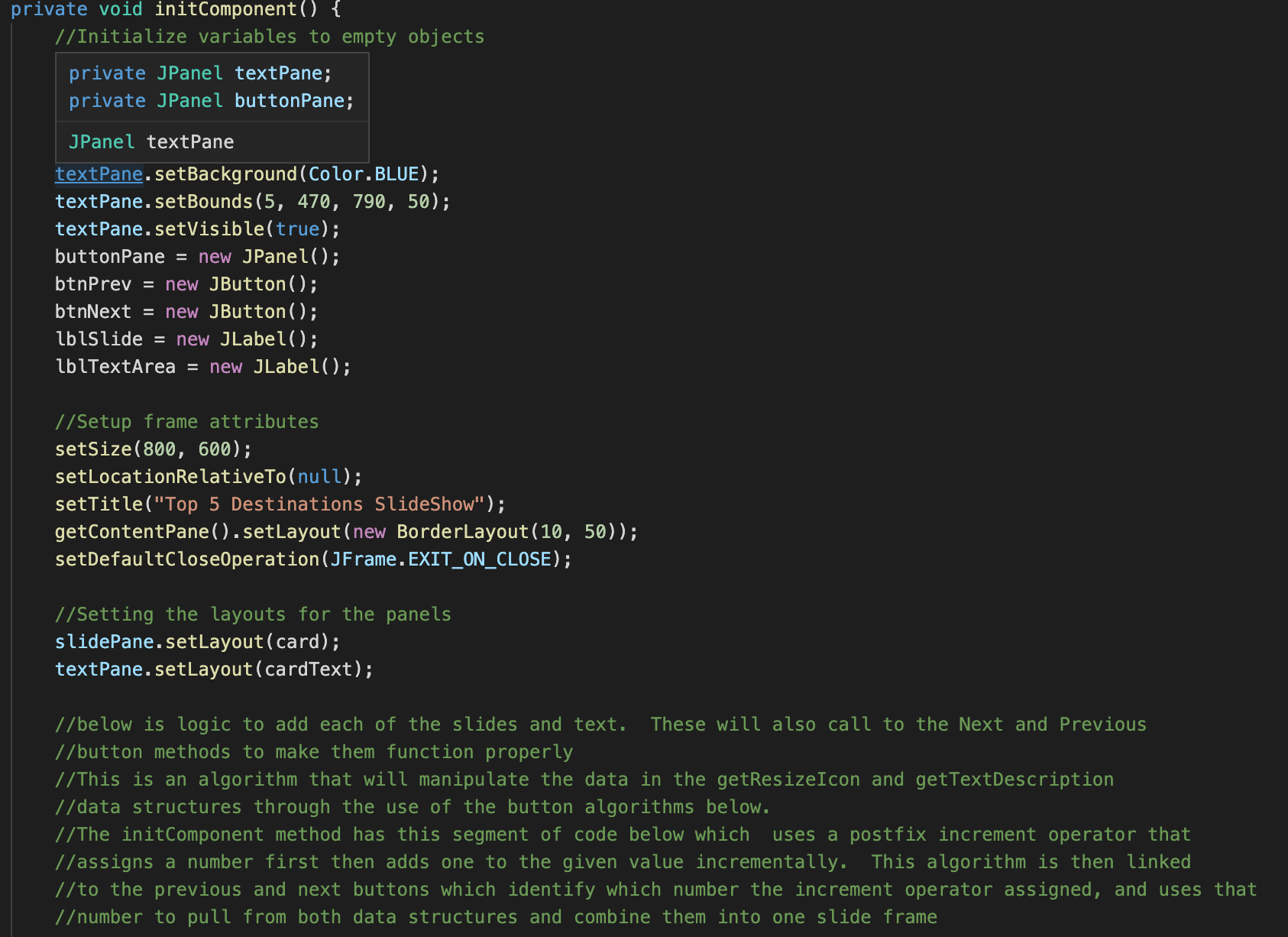
The TopFiveDestinationList.java and SlideShow.Java files were included in the ePorfolio as both showed the potential to demonstrate multiple displays of code functionality for the creation of working button controls, frame and panel attributes, and the insertion of images and text using if/else loops, the addition of colored backgrounds, borders, and the conversion of these files into a runnable jar file. The files included base code structures to allow for the insertion of images and text, with no declared variables to allow for panes and framework. The artifact was improved by declaring these variables, initializing them, setting up frame attributes and panel layouts, adding functionality for the previous and next buttons, and adding imagery and text with code to allow them to be displayed through a method that included if/else loops.

When modifying the artifact, I once again did a lot of code cross referencing online through stackoverflow.com and other sites to better understand the purpose of the different assert methods, and how to implement them. I used available software such as XCode to view and alter as necessary. This code was then placed into a downloaded version of Eclipse IDE for Mac, where I could further alter code, then create a runnable jar file in addition to the saved java file.

The code below demonstrates the insertion of imagery and text linked to images in the resource file, using if/else statements to determine which image to display. This is code from SlideShow.java. The two data structures shown are used to store image and image sizing data. This was enhanced to pull in images from the resource file and define their measurements. These are the Methods to get the images and text. It also links to imagery in the resource file folder. The button code pulls back to this structure, determining which image and text to display.



The code below displays Previous and Next Button algorithms, along with logic that is uses to determine the image and text slides to display:



Text

Description automatically generated

Text

Description automatically generated

The algorithm created to add each of the slides and text as seen here sets parameters for the Previous and Next buttons, adding to their functionality. This segment of code uses a postfix increment operator that assigns a number first then adds one to the given value incrementally. This algorithm is then linked to the previous and next buttons which identify which number the increment operator assigned and uses that number to pull from both data structures and combine them into one slide frame. The initComponent method used is an algorithm which identifies which number the increment operator assigned, by using both the previous and next buttons, determining the line number to pull from both data structures and combining them into one slide frame. Previous Button Functionality is linked to the previous code to add functionality. It uses the increment operator to go back one number from the currently assigned one, then go back to the previous pane.

**Graphical user interface

Description automatically generatedGraphical user interface, website

Description automatically generated**

Above are examples of the results of the code, with various vacation destinations displayed, **Graphical user interface

Description automatically generated**along with descriptive text, a title for the program, and functional buttons allowing the client to scan through all the slideshow pages.

Graphical user interface, text, application

Description automatically generated

The code above shows that Eclipse IDE environment was used to enhance data structure code for the TopFiveDestinationList.java file. This was also used to convert both java files to runnable jar files that you can open without the necessity of using an IDE or other environment.

The enhancements increased functionality in several different areas as determined in the Code Review by identifying, enhancing, and commenting on data structures in the code and their associated algorithms. Data structures were also used to store and organize data. The identified algorithms were used to manipulate the data in their associated structures.

**The two data structures** in the **SlideShow.java file** were identified as:

a. The getResizeIcon method, which is used to store image and image sizing data. This was enhanced to pull in images from the resource file.

b. The getTextDescription method, which is used to store descriptive data for each image.

**The two associated algorithms** in the **SlideShow.java file** were identified as:

a. The initComponent method, which has a segment of code which uses a postfix increment operator that assigns a number first then adds one to the given value incrementally. This algorithm is pulls from the two data structures to determine which data to pull from it and display.

b. The second associated algorithm originated from the same initComponent method, where the code has the button action identify which number the increment operator assigned and used that to pull from both data structures and combine them into one slide frame. The button algorithm determines whether to subtract one number from the postfix increment operator or add one number, depending on the button activated, then go to the next or previous slide and text pane.

**Artifact III for Category III: Algorithms and Data Structures Written Narrative**

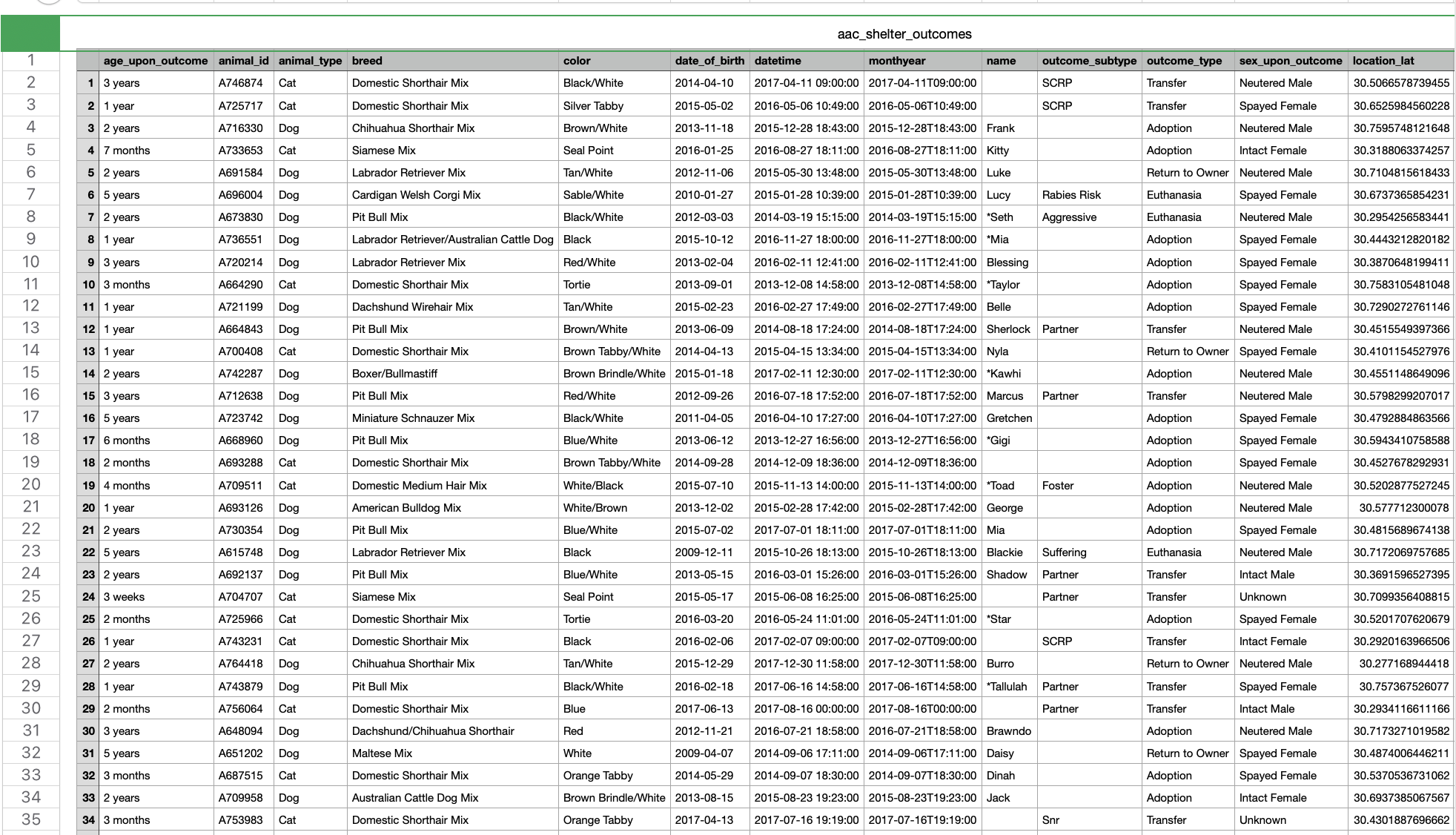
For Artifact three in Category Three: Databases, I selected work done in CS340 – Advanced Programming Concepts in Client/Server Development. For this course created a dynamic dashboard linking several databases in the Jupyter Notebook using .csv, .py, and .ipynb Python and Javascript language files in MongoDB. The key databases used were ProjectTwoDashboard.ipynb and Animal\_App.py, with a aac\_shelter\_outcomes.csv file. The objective was to apply database systems concepts and principles to create a client/server database application that interfaces with client-side code. This code pulled data to display geo-mapping, statistics, and other user data from a database powered by MongoDB. My enhancement plan included expanding on the current MongoDB API (application programming interface) via JavaScript to make it more user friendly, as well as providing more selection options through improving on the current code. This included enhancing CRUD (create, read, update, delete) elements of my code.

This artifact is an application the purpose of which is to provide a user-friendly interface to pull information from a database written in Python and powered by MongoDB, to retrieve information from animals, namely dogs, and update or alter same database. The objective of this project was to create a database, an API and a dynamic dashboard designed for the client to identify good dog candidates for search and rescue training. This artifact was created in the CS 340 SNHU for Advanced Programming Concepts, initially with minimal functionality and no security.

This project can be used to query MongoDB by creating and reading data from MongoDB*.* The code was altered and run via Jupyter Notebook, as well as through the XCode application. Jupyter Notebook was accessed for Mac via the Anaconda.Navigator application.

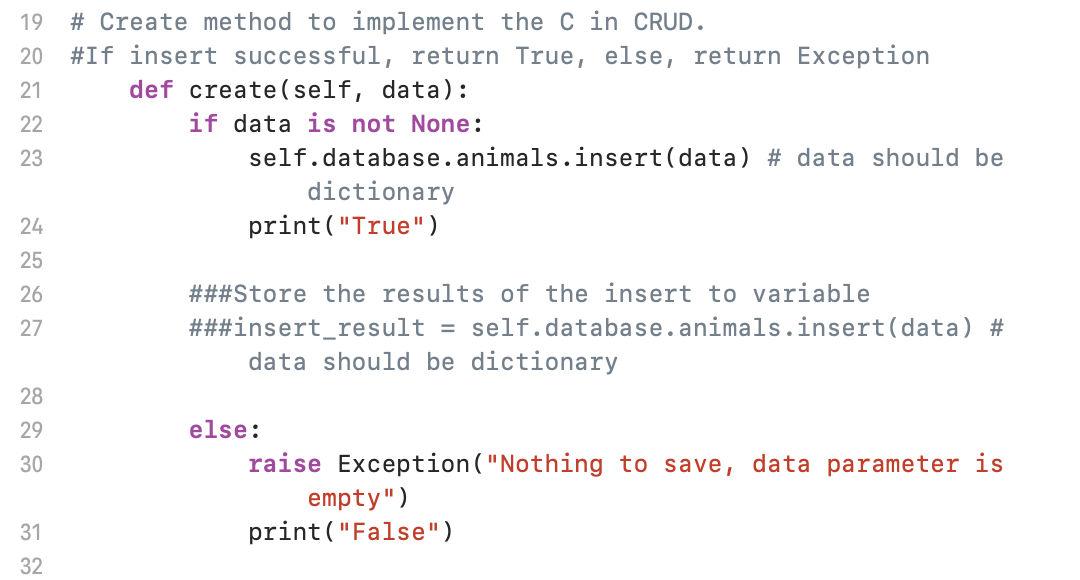
This artifact represents what a fully structured database and its code can do, from an effective API and a base data file to CRUD functionality that enables the creation, readability, update capability and delete functionality, all of which are essential in creating a functional database. Some specific enhancements included additions to the password interface for added security, expanded coding for data creation and insertion into existing .csv file, expanded search functions in the ReadData operation to effectively search for existing data and display it, expanded update operations with easy-to-understand input keys to include error notifications and if/else loops, and delete functionality to include error notifications and if/else loops to eliminate existing data.

The enhancements increased functionality in several different areas as determined in the Code Review. When modifying the artifact, I did code referencing online to get ideas on how best to create a polished code. I learned several ways of writing the CRUD functionality to try and cover all possible errors, as well as how detailed the dashboard code needed to be to provide any sort of functionality.



Above is the aac\_shelter\_outcomes.csv file which is central to the mission of the database program. Here is existing data that can be adjusted, deleted, added to, or searched through. Below is an example initial and final code enhancements for the C operation for C in CRUD, to create new lines of data in the .csv file:

Initial code before enhancement for Create:



Create code as seen below after enhancements that involved adding code to ensure user enters correct data fields. The first part of the Create code collects the data, while the second part validates and inserts results, or issues Exceptions via if/else statements.

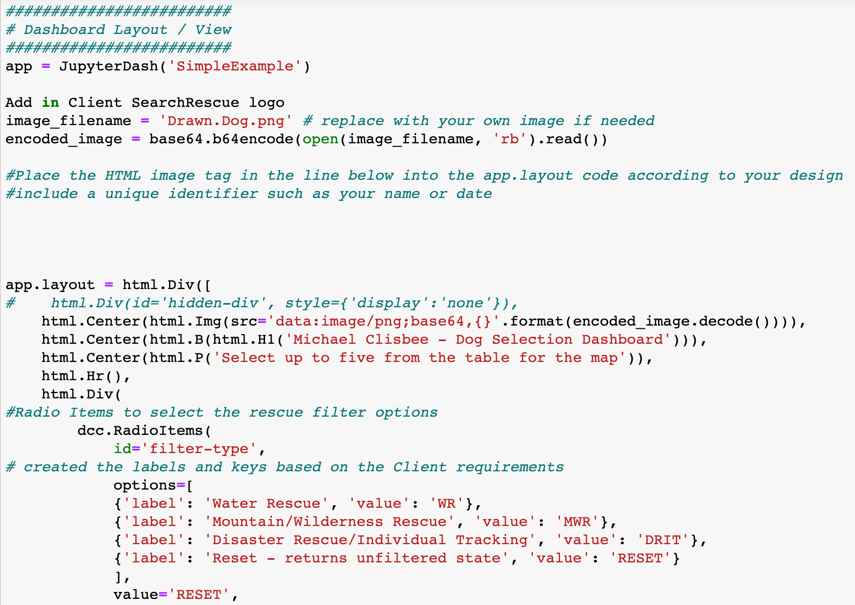


Code below adds security to application by adding username and password requirements.

Graphical user interface, text, application

Description automatically generated

The code below is part of the final dashboard code that includes unique client logo, dashboard title and radio items to select rescue filters based on the client requirements. This code can be altered to meet client needs.



The Dashboard code is set up to collect all data and depending on data selected, create a specific pie chart and geolocation chart.

**ePortfolio Overall**

By fulfilling my enhancement plans, I demonstrated skills learned in the languages of Java and Python that I can apply toward improving existing code. Namely the ability to create effective test files for their respective java files to ensure they function correctly. As well, creating detailed CRUD functionality, adding Create, Read, Update and Delete functionality with safeguards included.

**Link:** [**CS499 SNHU Computer Science ePortfolio**](https://michaelpclisbee.github.io/)

**Link:**  [**Informal Code Review (YouTube Link)**](https://youtu.be/gQ-wygnmFLA)