# CS 340 README for Grazioso Salvare’s Custom Dashboard

## About the Project

*This application allows the customer to use a web-based dashboard to access a database using CRUD functions to navigate, read, and edit to fit their needs.*

**Required Functionality:**

This application/project is layered using MongoDB as a database, Python as middleware, and a Plotly/Dash/Leaflet based visualization layer to produce the html. Python is used to control the application and utilizes a module for CRUD operations using PyMongo. The module contains create, reade, update, and delete functions.

**CRUD Code From Module**

A screenshot of a computer code

Description automatically generated

**Work in Progress**

The application is nearly complete but is currently unable to run. We are in the testing and debugging phase. There appears to be a launching issue that we have not worked out yet….

**Tools**

MongoDB was the obvious choice because of its ease of use and compatibility with the middleware. Python, the middleware, is perfect for use with MongoDB because of the ability to modularize code and the dictionary capabilities. Plotly powers the Dash leaflet and is a popular graphing application. The component-based architecture allows the user to utilize dynamic and responsive visuals. The charts that depict the data are excellent and the map generation and interactivity are easy to use and effective.

**Steps**

The first step to completing this project was to create the database in MongoDB and add the data. Next, I had to create a username and password method inside the CRUD module to allow users to log in. The next step was to design the reusable code in the Python CRUD module for database navigation and editing. The following step was to test the CRUD module using the Jupyter application. Lastly, the web application needed to be written and set up with the customer’s logo.

**Challenges**

The challenges on this project consisted of mostly inexperience with this type of application and the software. The provided data was not very clean and created sorting and query issues. Pattern matching utilizing regular expressions was needed to get proper results. PyMongo does not accept regular expressions and required some editing to be acceptable. With more practice and experience, I feel I will find another way that will be more efficient and compact.

## Contact

Your name: Wesley Blackwell