AP CS Principles
Abstraction Project

Tasks:
Using the abtractionProject_start starter code (found here), complete the project so that it does the following tasks:

1. Use the files in the data folder to create a database and tables.
2. The opening table should show the schools. Clicking on either a side nav item or dashboard view details link will display a table of the requested data.
    a. Schools -> school name, teacher name, teacher email
    b. Teams -> Team name, school name
    c. Programmers -> Name, grade, email, team name, school name
    d. Problems Solved -> Team Name, problem number, finish time
3. Fill in the Dashboard Graphics with the appropriate number of teams, schools, etc.
4. The graph should show the problem numbers and the time (in minutes) that it took each team to finish the problem. The graph should be a Combo Chart and, when clicked, should be viewed larger in a modal window.

Focus:
The focus of this project is abstraction. You will need to plan how you will read data, use procedural/data abstraction, separate functionality in an MVC style format, and utilize an API.

Deliverables:

- Functioning Program (one from the team)
- Document of all code written (not code provided by me). All code should be commented for readability and for attribution. This should be submitted as a PDF and in printed form. (one from the team)
- A second PDF that answers the following two questions (one from each person):
    - Describe at least two difficulties and/or opportunities that you encountered and how they were resolved or incorporated in the development of the program. One of the items must be independent, while the second can be collaborative or independent. (max 200 words)
    - Capture and paste a program code segment that contains an abstraction you developed individually (on your own). Mark it with a rectangle. Explain how your abstraction helped manage the complexity of your program. (max 200 words)

AP CS Principles
Abstraction Project - Grading

	2	1	0
Program Functionality	Program reads from the data tables. Nav links trigger the display of the appropriate information. Dashboard graphics show the appropriate numbers. All information is correct.	Program reads from the data tables. Nav links trigger the display of information (may have some errors). Dashboard graphics show numbers, though they may not be exactly correct.	Program may or may not read from the data tables. The information tables and dashboard numbers are not correct, or not present.
Data Chart	The data chart shows the requested information in the specified format. When clicked, the chart is enlarged in a modal window.	The data chart shows the requested information, though it may not be in the specified format. A modal opens, but it may not show an enlarged version of the chart.	The data chart is missing or does not show the required information. There may or may not be a modal.
Code Delivery	Code is delivered in two forms: printed and submitted as an electronic PDF.	Code is delivered in one of the required formats.	Code is not delivered, except within the project.
Code Style	Code is commented as required. Comments are used to increase readability, including method purposes and/or explains code to reader. Code is also marked up to cite author of specific parts.	Code contains comments. Code is marked to cite author of specific parts. Code may be somewhat lacking in documentation/explicative purposes.	Code commenting is limited or non-existent. Code does not clearly delineate author and does not do an adequate job of documentation and/or explanation.
Procedural Abstraction	Code uses procedural abstraction effectively, assigning methods to handle repetitive tasks. Methods are well named and efficiently written. Method names clearly indicate purpose.	Code uses procedural abstraction. Methods are used but may be over or under utilized. Methods are well named. Method names indicate purpose.	Procedural abstraction is not used or very minimally used. Methods, if they exist, may not be appropriately named and are not efficient. Method names may or may not indicate purpose.
Data Abstraction	Data abstraction is effectively used through the development of classes. Classes are well organized, efficiently written and use the concepts of encapsulation.	Data abstraction is used through the development of classes. Classes are organized, but may not use the concepts of encapsulation.	Data abstraction is not used or minimally utilized. Classes, if they exist, may be disorganized and may not use the concepts of encapsulation.
Program Flow	The program effectively separates data, functionality and display. The program would easily allow for a different type of output (display) and input (data) without major modification.	The program attempts to separate data, functionality and display, though it may not be done as effectively as possible. The program would allow for a different type of output and input, but may require some modifications.	The program does not attempt to separate data, functionality and display OR does so unsuccessfully leading to major required revisions, should the output or input be altered.
Final Product	The final product shows significant student learning and reflection. Design decisions show thoughtfulness and provide a logical and smooth flow of information. There is evidence of	The final project shows some evidence of student learning and reflection. Design decisions show some thoughtfulness and provide all a logical flow of information. There is evidence of student effort.	The final project shows minimal student learning and/or reflection. Design decisions seem unfounded and expedient. There is minimal evidence of student effort

	exemplary student effort.		
Question 1	At least two difficulties/opportunities are discussed. The process of resolution/incorporation in the development of the program is included. At least one of the items is an independent struggle/success. The answer is within the 200 word maximum.	At least two difficulties/opportunities are discussed. The process of resolution/incorporation in the development of the program may not be included. One of the items may or may not be independent. The answer is within the 200 word maximum.	There may or may not be two difficulties/opportunities discussed. The process of resolution/incorporation in the development of the program may not be included. One of the items may or may not be independent.
Question 2	A program segment containing an individually developed abstraction is delivered. It has been marked with a rectangle and there is a clear, effective description of how it helped manage the complexity of the program. The answer is within the 200 word limit.	A program segment containing an individually developed abstraction is delivered. It may be marked with a rectangle and there is a description of how it helped manage the complexity of the program. The answer is within the 200 word limit.	There may or may not be a program segment containing an individually developed abstraction. There may or may not be markup and/or a description of how it helped manage the complexity of the program.

---

20 -> 100	15 -> 85	11/12 -> 75	8/9 -> 65	5 -> 50	2 -> 20
18/19 -> 95	13/14 -> 80	10 ->70	6/7 -> 60	4 -> 40	1 -> 10
16/17 -> 90				3 -> 30	0 -> 0

NOTES: