**Gather requirements and initial information – Lonzo**

**Project Requirements and Specifications**

Our project involves analyzing a medical insurance dataset (provided in CSV format) using Java only, without relying on external libraries such as NumPy, pandas, or matplotlib. The assignment emphasizes using Java Collections classes to implement coded solutions for statistical calculations, histograms, fairness checks, and regression analysis.

The required features must be coded and tested in Java, including: storing subsets of records in custom objects, performing descriptive statistics such as count, mean, standard deviation, percentiles, minimum, and maximum, building text-based histograms (both horizontal and vertical) for attributes like age, BMI, children, and smoking status, conducting fairness checks by region, verifying hypotheses such as whether smokers have higher charges, and implementing simple linear regression models with Pearson correlation coefficients that are applied to new test values outside the dataset. In total, there are 22 required coded features that must be completed, with each assigned to specific team members based on their role.

All results must be displayed in text form only, meaning no graphical interfaces, charting libraries, or external visualization tools may be used. Every output from the program—whether statistical summaries, histograms, fairness results, hypothesis tests, or regression equations—must be printed as plain text to the console. For example, descriptive statistics should be listed in a readable table format using text, histograms should be represented by rows or columns of symbols (such as “#”) to approximate the distribution, and hypothesis or fairness checks should print the calculated values alongside a clear true/false result. Regression analysis should output the formula, correlation coefficient, and predicted values in text form. This requirement applies to all 22 features, ensuring the entire project remains console-based and consistent across team contributions.

**Team Roles and Responsibilities**

Our team is Team Yellow:

Lead (lhamilt3) – Manages the GitHub repository, reviews and merges all pull requests, ensures project progress, and co-authors the initial driver program and Software Design Document (Steps 1–2).

Designer (qabdul1) – Creates class diagrams using Microsoft Visio, collaborates on the driver program and design document, and writes 10 additional unit tests beyond SWE/Tester coverage.

Software Engineer (ebutler4) – Codes even-numbered features and writes Arrange-Act-Assert (AAA) unit tests for odd-numbered features.

Tester (ddawson3) – Codes odd-numbered features and writes AAA unit tests for even-numbered features.

Because only the Lead is a collaborator, the other team members must work through forks and pull requests. The Lead reviews, verifies, and merges these contributions.

**Technical Constraints and Limitations**

Language: Java (no Python or external statistical/plotting libraries).

Libraries: Only standard Java Collections and I/O are allowed.

Visualizations: Only text-based histograms are permitted (no GUI or plotting libraries).

Collaboration: Limited GitHub permissions—PR-based workflow with the Lead as gatekeeper.

Testing Framework: AAA (Arrange-Act-Assert) methodology must be followed, ensuring deterministic and repeatable unit tests.

**Stakeholder Expectations**

The client expects a clear division of labor aligned with team roles, a shared driver program that all members can run and extend, and a Software Design Document (Steps 1–2 initially) that outlines the system’s requirements, architecture, and roles. The client also requires reliable, coded implementations of all 22 statistical and analytical features, unit testing for each feature, additional edge-case coverage by the Designer, and a professional GitHub workflow with collaboration through pull requests, proper testing, and CI integration.

**Outline the system architecture - Qadir**

**General Overview**

The main driver will implement three major classes using the patient as the base object: the list creator, stats calculations, and a histogram displayer. The patient object will have the following attributes: age, sex, BMI, children count, whether they are a smoker or not, there region of treatment, and the charges for their services. The name of the game with these classes is versatility with the intention of making each class able to be repeated in the main driver so that multiple redundant classes are not made. These classes are few, but complex and each method should be tested individually to avoid making the debugging process unnecessarily long and tedious to sift through.

**Class Design and Relations**

Note that the wording in the design uses words pertaining to python as my knowledge of Java is lacking, despite this the major ideas should still apply.

Patient object

* The patient objects will be put into a list using the list creator that will either create a list of the entire patient data or individual attributes of each patient for easy separation and use in the statistic calculator.

List Creator

* The list creator will be designed in such a way that it can take in either all patient objects or their individual attributes.

Statistic Calculator

* The statistic calculator class will be used to calculate the minimum and maximum values, mean, standard deviations, average, and percentiles of all the patient objects integer attributes including number of children, BMI, age, and chargers.

Histogram Display

* The histogram displayer will display a histogram (of course). It will take the range, bin size, and frequency as integer values. It will count the list of items and can be displayed as either a horizontal or vertical set of strings.

Fairness

* A class that will focus on determining that the charges applied to patients with similar conditions/attributes do not have any disparities greater than 5%. For instance, if it cycles through a collection of patient objects with the parameters set to keep track of each attribute it should be able to identify if there are any with charges that differ more than 5%. It will return “fair” if not otherwise it will return the disparity.

Conditional Averages

* This class will follow the same principle as the average’s method in the statistical calculator class, but with the addition that it will take in attributes as parameters and use them to identify certain patients and find the averages of the given integer values.

Sort Regions

* Sort Regions will cycle through the patients list and sort them by their regions, returning the final result as a string. It will return the string displayed in order by their averages from greatest to lowest.

Compare Values

* It will take in two patient objects and compare the given integer values to each other. Returning whether one is greater than or less than the other based on the order of lowest to highest.

Correlation Coefficient

* This class will determine the correlation between two given integer attributes from two given patient object classes.

