# Project Report: GradeBook Analyzer

**Course:** Programming for Problem Solving using Python

**Assignment:** Mini Project - Analysing and Reporting Student Grades

**Submitted By:** [Your Name Here]

**Date:** [Current Date]

## 1. Introduction

The **GradeBook Analyzer** is a Python-based Command Line Interface (CLI) tool designed to assist educators in automating the process of calculating student grades and statistics. Traditionally, grading is a manual process prone to calculation errors and time inefficiencies. This project aims to digitize this workflow, allowing for data ingestion via manual entry or CSV files, followed by automated statistical analysis and report generation.

## 2. Problem Framing (CRISP-DM Approach)

In alignment with the Cross-Industry Standard Process for Data Mining (CRISP-DM), the problem is framed as follows:

* **Business Understanding:** Lecturers need a fast, consistent, and error-free method to process student marks. Manual calculation of averages and letter grades for large classes is unfeasible.
* **Data Understanding:** The data consists of student names (Strings) and their corresponding marks (Float/Integer). The data source can be direct user input or structured CSV files.
* **Data Preparation:** Data must be validated (marks between 0-100) and cleaned (removing whitespace from names) before processing.
* **Modeling/Logic:** The core logic involves statistical aggregation (Mean, Median, Min, Max) and conditional logic for grade classification (A, B, C, D, F).

## 3. Objectives

The primary objectives of this project are:

1. To develop a modular Python application for data analysis.
2. To implement file I/O operations for reading CSV datasets.
3. To utilize Python's standard libraries (statistics, csv) for efficient computation.
4. To demonstrate proficiency in control flow (if-elif-else) and list comprehensions for data filtering.

## 4. System Design and Implementation

### 4.1. Technology Stack

* **Language:** Python 3.x
* **Libraries:** csv (File handling), statistics (Mathematical analysis), sys (System operations).
* **Interface:** Command Line Interface (CLI).

### 4.2. Key Features

1. **Dual Input Mode:** Users can input data manually for small batches or load a marks.csv file for bulk processing.
2. **Robust Error Handling:** The system handles ValueError for non-numeric inputs and FileNotFoundError for missing CSV files.
3. **Statistical Engine:** Automatically computes the Class Average, Median Score, Highest Score, and Lowest Score.
4. **Automated Grading:** Assigns letter grades based on a predefined rubric:
   * **A:** 90 - 100
   * **B:** 80 - 89
   * **C:** 70 - 79
   * **D:** 60 - 69
   * **F:** < 60
5. **Pass/Fail Filtering:** Identifies students who scored below the 40-mark threshold.

### 4.3. Code Structure

The solution is implemented in a single module gradebook.py containing the following key functions:

* calculate\_average() & calculate\_median(): Core statistical functions.
* assign\_grade(score): Maps numerical scores to categorical grades.
* main(): The driver function that handles the while loop for the application menu.

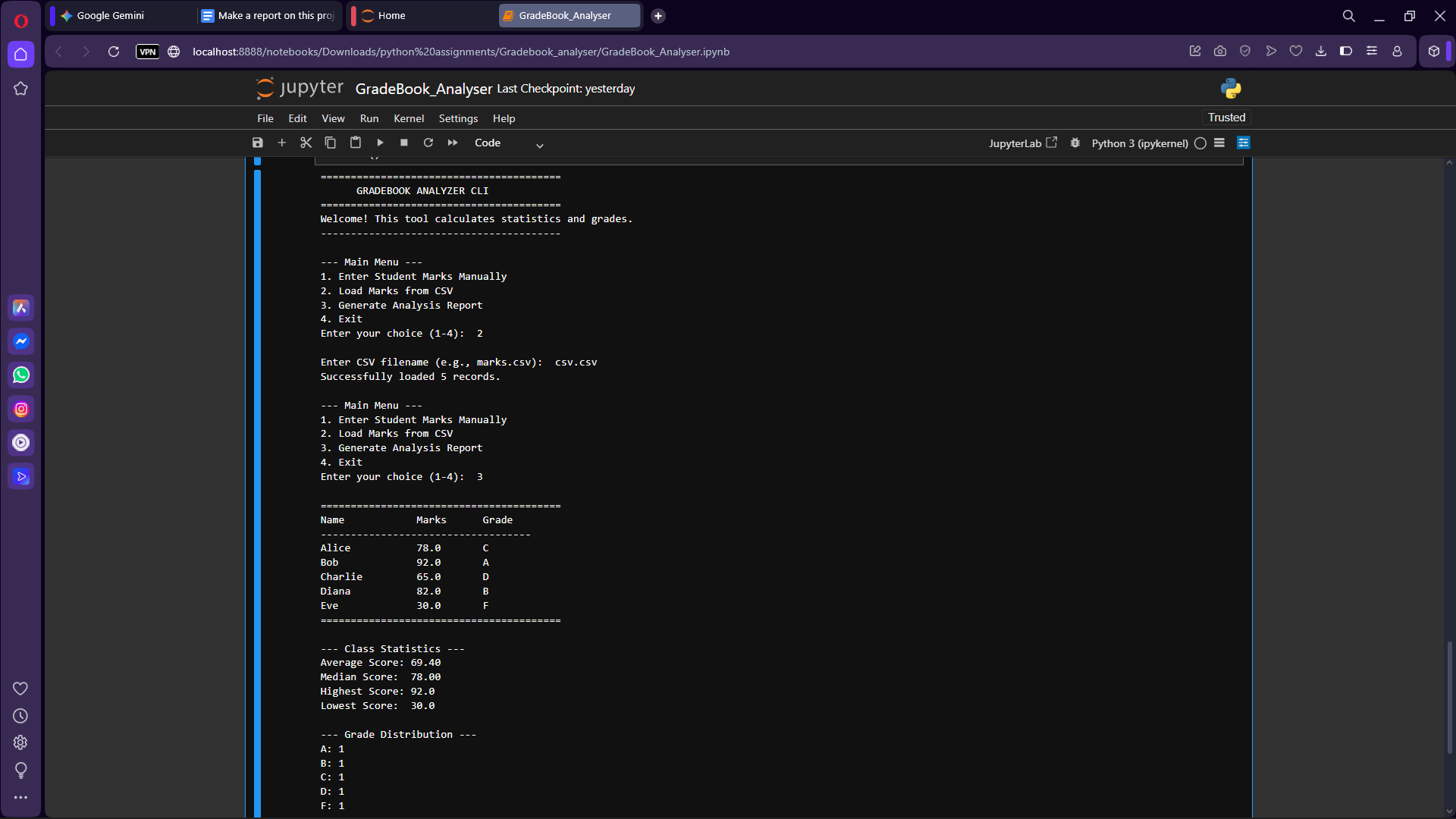
## 5. Implementation Logic (Code Snippet)

*The following snippet demonstrates the core logic used for generating the final report and filtering students.*

# Grade Assignment using Dictionary Comprehension  
grades = {name: assign\_grade(score) for name, score in marks.items()}  
  
# Filtering Data using List Comprehensions  
passed\_students = [name for name, score in marks.items() if score >= 40]  
failed\_students = [name for name, score in marks.items() if score < 40]

## 6. Results and Output

Upon execution, the system provides a menu-driven interface. Below is a sample output based on test data.

****

## 7. Conclusion

The GradeBook Analyzer successfully meets all assignment requirements. It provides a user-friendly interface for grading automation, reducing the time required for manual calculations. The use of Python's list comprehensions and dictionary structures ensures the code is both efficient and readable. Future enhancements could include exporting the results back to a CSV file or visualizing the grade distribution using matplotlib.