- **Title: Student Performance Analyzer**
- **Author: TANISHA BARIAR**
- **Course : CSE IN HEALTH INFORMATICS 25BHI10079**
- **Date of Submission 25TH NOVEMBER**

**Introduction**

- Brief description of the Student Performance Analyzer project purpose, objectives, and scope.
- High-level summary of what the tool does (record management, analysis, text menu interface, utilities).

## Problem Statement

- Define the problem: managing, analyzing, and visualizing student performance data is often manual and error-prone.
- Target users: Teachers, students, researchers looking for simple analysis tools.

## Functional Requirements

- Add, load, and save student records (manual input or from CSV)
- Compute total, average, and grade for each student
- Sort students and find kth highest/lowest
- Remove duplicate records by total score
- Prime, GCD, Factorial, Fibonacci utilities

- Text menu interface and report display

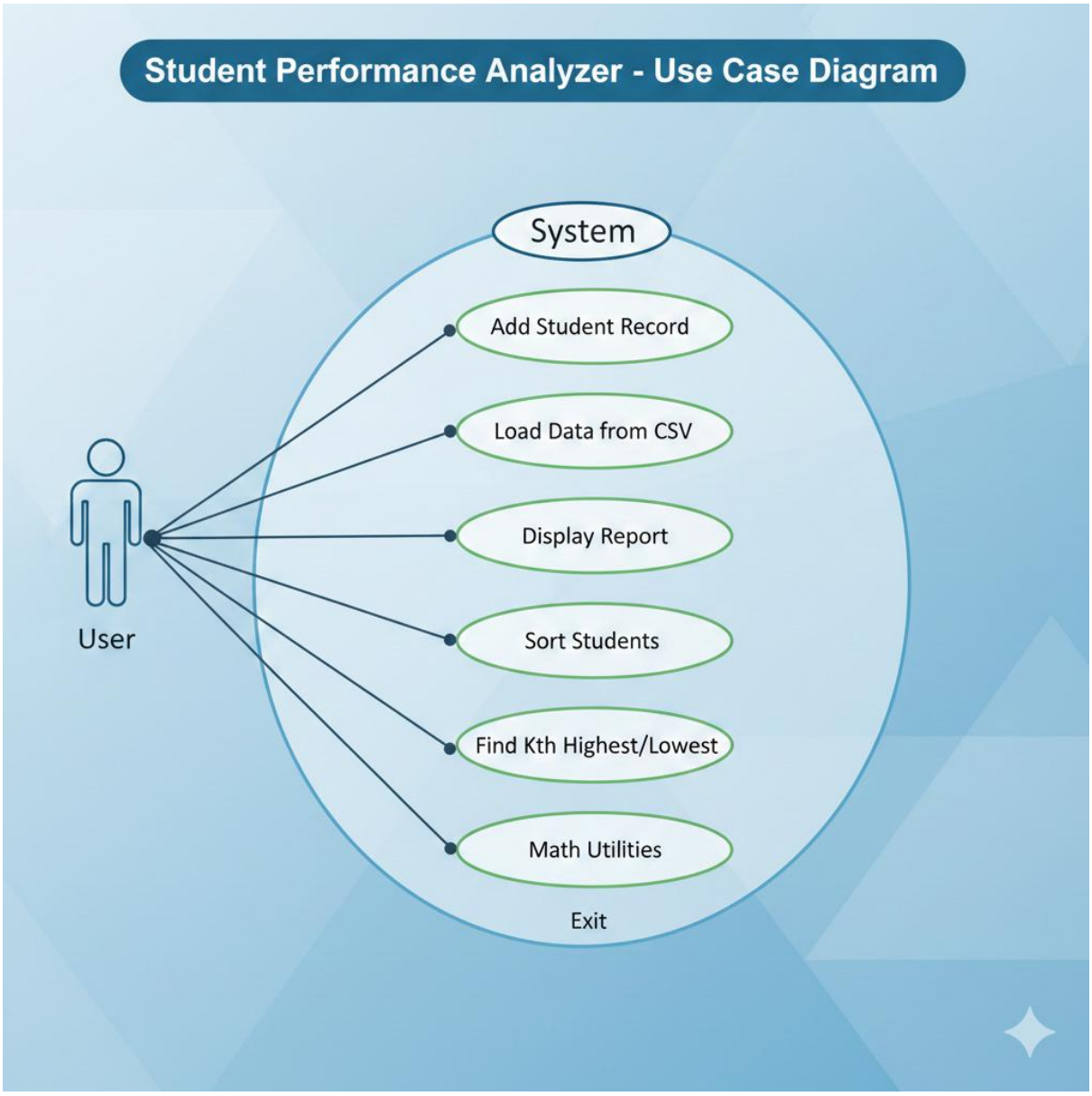
**Non-functional Requirements**

- Simple, user-friendly CLI
- Portability (runs on standard Python environment)
- Maintainable code structure (utility module for calculations)
- Efficient handling of small-to-medium datasets

**System Architecture**

- High-level overview diagram: Main script (menu controller), utility module, IO (CSV), and user interface
- Description of component interactions

**Design Diagrams**




## Design Decisions & Rationale

- Chose Python for accessibility and simplicity
- CSV file format for easy manual editing and sharing
- Text UI for rapid prototyping

## Implementation Details

- Summarize the code structure: main script, utils.py, CSV file operations
- Key function descriptions: load_from_csv, save_to_csv, input_student, display_report
- Any packages used (csv, os)

## Screenshots / Results

```
Student Performance Analyzer
1. Add student
2. Load sample CSV
3. Display report
4. Sort by total (desc)
5. Find Kth highest/lowest
6. Remove duplicate totals
7. Utilities (prime, gcd, factorial, fibonacci)
8. Save to CSV
9. Exit
Choose option:  3

Name    Total   Average Grade
----------------------------------------
Amit    450     90.00   A+
Sneha   380     76.00   B
Rahul   290     58.00   D
Priya   446     89.20   A
Vikram  225     45.00   F
----------------------------------------
```

## Testing Approach

- Manual testing: Input validation, boundary values for marks, CSV file integrity
- Functional testing for all menu features

## Challenges Faced

- Ensuring robust input validation
- Managing duplicate total scores correctly
- Providing useful menu-driven utilities

## Learnings & Key Takeaways

- Experience with Python file operations and modularity
- Understanding of basic algorithms (sorting, searching, math utilities)
- Importance of user interface clarity even in CLI tools

## Future Enhancements

- GUI version (Tkinter, web-based)
- More detailed analytics (trends, subject-wise ranks)
- Integration with databases (e.g., SQLite for persistent storage)

**References**

- Python official docs
- Project guidelines provided by the institution
- Any online documentation or learning resources you used