System Requirements Specification Index

For

Minecraft Score Converter

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1 Project Abstract

Mojang, the creator of Minecraft, requires a scoring system that can handle different types of score formats from various game activities. The Minecraft Score Converter is a Python console application developed to standardize different scoring formats into a unified system. This system converts mining points (stored as text from legacy systems), combat accuracy (stored as decimal numbers), and achievement bonuses (stored in hexadecimal) into a standardized integer format. The application ensures consistent score calculations across different game modes and allows for fair player rankings on the global leaderboard. This standardization tool is crucial for maintaining fair play and accurate player statistics across the Minecraft gaming ecosystem.

2 Business Requirements:

Screen Name	Console input screen
Problem Statement	1. User needs to enter different types of game scores into the application 2. The application should convert all scores to standard integer format 3. The console should handle the following inputs: Mining points (as text), Combat Accuracy (as decimal), Achievement Bonus (as hexadecimal) 4. Format output clearly with proper units

3 Constraints

3.1 Input Requirements

1. Mining Score:

- Must be stored as string in variable **mining_score**
- Must contain only digits
- Example: "100"

2. Combat Score:
   - Must be stored as float in variable **combat_score**
   - Must be non-negative
   - Example: 98.7
3. Achievement Bonus:
   - Must be positive numbers
   - Must be stored as string in **achievement_hex**
   - Must contain valid hexadecimal characters (0-9, A-F)
   - Example: "1F"
4. Player Information:
   - Must store name as string in **player_name**
   - Cannot be empty
5. Player Stats Format:

   A. Calculate sum of all converted scores

   B. Must be stored as list in **player_stats**

   C. Must contain exactly **[player_name, total_score]**

   D. Example**: ["Steve", 250]**

3.2 CONVERSION CONSTRAINTS

1. Mining Points Conversion:
   - Convert string to integer using int()
   - Store result in **mining_points**
2. Combat Score Conversion:
   - Convert float to integer using int()
   - Store result in combat_points
   - Must truncate decimal part

3. Achievement Bonus Conversion:
    - Convert hex to integer using int(x, 16)
    - Store result in **achievement_bonus**
4. Total Score Conversion:
    - Calculate sum of all converted scores
    - Store in **total_score**
    - Convert to string for display in **score_display**

3.3 Output Constraints

1. Display Format:
    - Show player name and all score components
    - Format total score as string in **score_display**
    - Store player record in **player_stats** list
    - Each score must be on a new line with proper labeling
    -
2. Required Output Format:
    - Show player name and all score components
    - Show "Mining Points: {value}"
    - Show "Combat Points: {value}"
    - Show "Achievement Bonus: {value}"
    - Show "Total Score: {value}"

4. Template code Structure:

1. Type Conversion Functions:
    - convert_string_to_int()
    - convert_float_to_int()
    - convert_hex_to_int()
    - convert_score_to_string()
    - create_player_list()

2. Input Section:
    - Get mining score (string)
    - Get combat score (float)
    - Get achievement bonus (hex)
    - Get player name

3. Conversion Section:

- Convert each score type
- Calculate total score
- Create player record

4. Output Section:

- Display all score values
- Show total score
- Format player info correctly

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. Enter mining points when prompted
3. Enter combat accuracy score
4. Enter achievement bonus in hexadecimal
5. Enter Minecraft username
6. View complete score breakdown
7. Verify all conversions
8. Check final player statistics