System Requirements Specification Index

For

Worker Safety Eligibility Checker

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1 Project Abstract

The Worker Safety Eligibility Checker is a Python console application designed to streamline the evaluation of worker eligibility for various roles based on safety and performance criteria. This system helps managers assess workers for different positions by checking multiple qualification criteria, including safety training, experience, performance scores, and certifications. The system automatically evaluates eligibility for basic work, machine operation, safety supervision, night shifts, and trainer roles. By automating these assessments and providing instant results, the system ensures consistent application of safety standards and qualification requirements.

Business Requirements:

Screen Name	Console input screen
Problem Statement	1. System must validate worker eligibility for multiple roles 2. Application should combine multiple criteria using logical operators 3. Console should handle different validations using: AND operator for combining mandatory requirements, OR operator for alternative qualifications, Comparison operators (>, <, >=, <=, ==) for numeric thresholds

2 Constraints

2.1 Input Requirements

1. Boolean Inputs (yes/no converted to True/False):
    - safety_training: Must be stored as boolean, converted from user input 'yes'/'no'

- certification: Must be stored as boolean, converted from user input 'yes'/'no'
- first_aid: Must be stored as boolean, converted from user input 'yes'/'no'
- team_leader: Must be stored as boolean, converted from user input 'yes'/'no'
- night_shift_approved: Must be stored as boolean, converted from user input 'yes'/'no'
- Example: User enters 'yes', store as True and vice versa

2. Numeric Inputs:
    - experience: Must be stored as integer representing years of work
        - Must be non-negative and realistic (0-50)
        - Example: 5 (representing 5 years)
    - safety_score: Must be stored as integer representing assessment score
        - Must be between 0 and 100 inclusive
        - Example: 85 (representing 85%)
    - incidents: Must be stored as integer representing safety violations
        - Must be non-negative
        - Example: 2 (representing 2 incidents)
    - training_score: Must be stored as integer representing training performance
        - Must be between 0 and 100 inclusive
        - Example: 90 (representing 90%)
    - attendance: Must be stored as integer representing attendance percentage
        - Must be between 0 and 100 inclusive
        - Example: 95 (representing 95%)
3. Constants:
    - Must be stored as integer in variable MIN_EXPERIENCE = 2
    - Must be stored as constants MIN_SAFETY_SCORE = 75
    - Must be stored as constants MAX_INCIDENTS = 3
    - Must be stored as constants MIN_TRAINING_SCORE = 80
    - Must be stored as constants MIN_ATTENDANCE = 90

2.2 CALCULATION CONSTRAINTS

1. Basic Work Eligibility:
   - Must use AND, OR operators –
   - Store intermediate result in has_required_performance = (safety_score >= MIN_SAFETY_SCORE or experience >= MIN_EXPERIENCE)
   - Store intermediate result in has_safe_record = (incidents <= MAX_INCIDENTS)
   - Final check: safety_training AND has_required_performance AND has_safe_record
   - Return "Eligible" or "Not Eligible"
2. Machine Operation Eligibility:
   - Must use AND operator
   - Must validate basic_eligible is "Eligible" or "Not Eligible"
   - Store intermediate result in has_sufficient_training = (training_score >= MIN_TRAINING_SCORE)
   - Final check: basic_eligible == "Eligible" AND certification AND has_sufficient_training
   - Return "Eligible" or "Not Eligible"
3. Safety Supervisor Eligibility:
   - Must use AND operator
   - Must validate machine_eligible is "Eligible" or "Not Eligible"
   - Store intermediate result in has_good_attendance = (attendance >= MIN_ATTENDANCE)
   - Final check: machine_eligible == "Eligible" AND first_aid AND has_good_attendance
   - Return "Eligible" or "Not Eligible"
4. Night Shift Eligibility:
   - Must use AND, OR operator
   - Must validate basic_eligible is "Eligible" or "Not Eligible"
   - Store intermediate result in is_qualified = (team_leader OR experience >= 5)
   - Final check: basic_eligible == "Eligible" AND night_approved AND is_qualified
   - Return "Eligible" or "Not Eligible"

- Store in can_supervise_safety

5. Trainer Eligibility:
    - Must use AND, OR operator
    - Must validate supervisor_eligible is "Eligible" or "Not Eligible"
    - Store intermediate result in has_trainer_qualification = (incidents == 0 OR team_leader)
    - Final check: supervisor_eligible == "Eligible" AND has_trainer_qualification
    - Return "Eligible" or "Not Eligible"

2.3 OUTPUT CONSTRAINTS

1. Display Format:
    - Each result on new line
    - Show role name followed by eligibility status
    - Use simple headers with hyphen separators

2. Required Output Format:
    - Show "Eligibility Results:"
    - Show "-" * 30
    - Show "Basic Work: Eligible/Not Eligible"
    - Show "Machine Operation: Eligible/Not Eligible"
    - Show "Safety Supervisor: Eligible/Not Eligible"
    - Show "Night Shift: Eligible/Not Eligible"
    - Show "Trainer: Eligible/Not Eligible"

4. TEMPLATE CODE STRUCTURE:

1. Constants Definition:
    - Define all threshold constants at module level
    - MIN_EXPERIENCE
    - MIN_SAFETY_SCORE
    - MAX_INCIDENTS
    - MIN_TRAINING_SCORE
    - MIN_ATTENDANCE

2. Function Definition:
    - check_basic_eligibility(safety_training, safety_score, experience, incidents)
    - check_machine_operator(basic_eligible, certification, training_score)

        - check_safety_supervisor(machine_eligible, first_aid, attendance)
        - check_night_shift(basic_eligible, night_approved, team_leader, experience)
        - check_trainer(supervisor_eligible, incidents, team_leader)

3. Main Program Structure
    - Input Section
        - Get boolean inputs (yes/no responses)
        - Get numeric inputs with ranges
        - Convert all inputs to appropriate data types
    - Eligibility Calculation
        - Calculate each eligibility in sequence
        - Store results in appropriately named variables
        - Pass results to next level of checks
    - Output Section
        - Display header with separator
        - Show each eligibility result
        - Use consistent formatting

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. Enter all yes/no responses
3. Enter all numeric values when prompted
4. View eligibility results for all roles