System Requirements Specification Index

For

Worker Safety Eligibility Checker

Version 1.0

IIHT Pvt. Ltd.

fullstack@iiht.com

Table of Contents

Worker Safety Eligibility Checker

System Requirements Specification

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.

2 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

3 Constraints

3.1 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

3.2 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"

3.3 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. DETAILED IMPLEMENTATION GUIDE

5.1 Basic Eligibility Function

1. Implement basic work eligibility checking.

Function: `check_basic_eligibility(safety_training, safety_score, experience, incidents)`

- This function should determine if a worker meets basic requirements for any work role.
- A worker qualifies if they have completed safety training AND either have a good safety score OR sufficient experience AND have an acceptable number of incidents.
- Use the constants to check against minimum thresholds. The function should evaluate multiple conditions and return the appropriate eligibility status.

5.2 Machine Operator Function

2. Implement machine operator eligibility checking.

Function: `check_machine_operator(basic_eligible, certification, training_score)`

- This function checks if a worker can operate machinery.
- The worker must first be basically eligible, have the required certification, and meet the minimum training score requirement.
- All conditions must be satisfied for the worker to be eligible for machine operation.
- Check the eligibility string from the previous function and combine it with certification status and training performance.

5.3 Safety Supervisor Function

3. Implement safety supervisor eligibility checking.

Function: `check_safety_supervisor(machine_eligible, first_aid, attendance)`
This function determines if a worker can supervise safety operations.

The worker must be eligible for machine operation, have first aid certification, and maintain good attendance.

Verify the machine operator eligibility status and combine it with the additional safety supervisor requirements using the attendance threshold constant.

5.4 Night Shift Function

4. Implement night shift eligibility checking.

Function: `check_night_shift(basic_eligible, night_approved, team_leader, experience)`

- This function checks eligibility for night shift work.

- The worker needs basic eligibility and night shift approval, plus they must either be a team leader OR have sufficient experience (5+ years).
- This function provides alternative qualification paths - workers can qualify through leadership or through extensive experience.

5.5 Trainer Function

5. Implement trainer eligibility checking.

Function: `check_trainer(supervisor_eligible, incidents, team_leader)`

- This function determines if a worker can train others.
- The worker must be eligible as a safety supervisor and either have a perfect safety record (zero incidents) OR be a team leader.
- This function allows for alternative qualifications where leadership can compensate for incident history.

5.6 Main Program Structure

6. Complete the main program flow.

- **Input Collection:** Gather all required information from the user including boolean responses (yes/no questions) and numeric values (scores, years, percentages). Convert user inputs to appropriate data types for processing.
- **Sequential Checking:** Call each eligibility function in the proper order since some roles depend on eligibility for previous roles. Store each result appropriately.
- **Results Display:** Present all eligibility results in a clear, organized format showing each role and the worker's qualification status.

5.7 Logic Implementation Requirements

7. Use appropriate logical operations for combining criteria.

- **Condition Combining:** Different functions require different logical approaches - some need all conditions to be true (AND logic), while others accept alternative qualifications (OR logic). Some functions combine both approaches where certain requirements are mandatory while others offer alternatives.
- **String Comparison:** Functions that depend on previous eligibility results need to check the exact string values returned by earlier functions.
- **Threshold Checking:** Use comparison operators with the defined constants to determine if numeric values meet minimum requirements.

5.8 Return Value Format

8. Ensure consistent eligibility status reporting.

Standard Responses: All functions should return consistent string values indicating eligibility status. Use the same format across all functions for uniform results that can be easily processed and displayed.

6. 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

Execution Steps to Follow:

- All actions like build, compile, running application, running test cases will be through Command Terminal.

- To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal

- This editor Auto Saves the code

- If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B -command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.

- These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.

- To launch application: python3 filename.py

- To run Test cases: python3 -m unittest

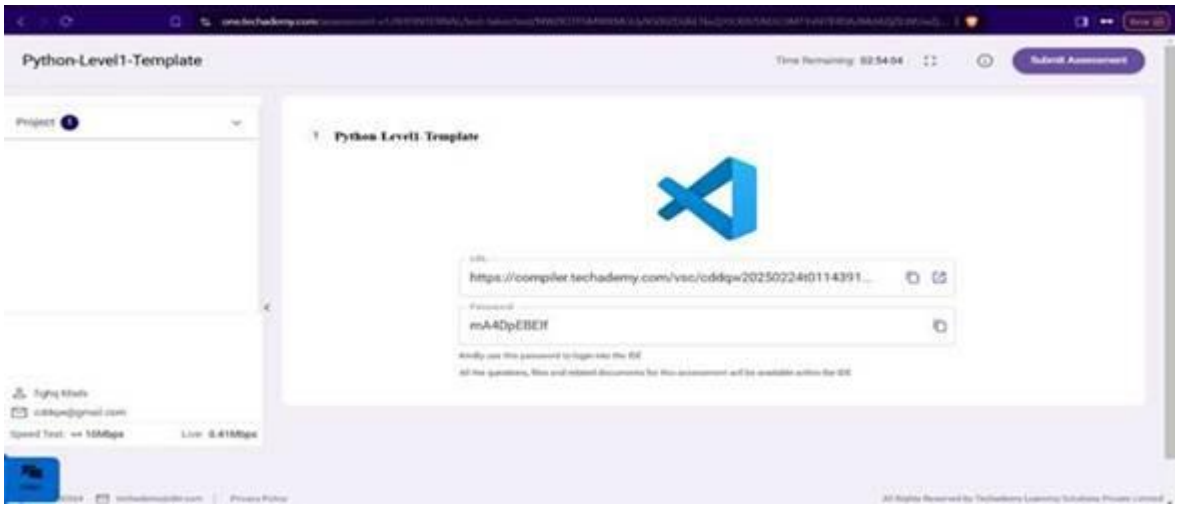
● Before Final Submission also, you need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

Screen shot to run the program

To run the application

Python3 filename.py

To run the testcase python -m unittest



● Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.