System Requirements Specification Index

For

Youth Centre Management System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1 Project Abstract

BrightFuture Foundation requires a simple management system for their youth center. The system will track different types of staff members, activities, and resources to streamline daily operations. By implementing abstract classes and interfaces, the system will demonstrate fundamental OOP principles while providing a practical tool for center administrators.

2 Business Requirements:

Screen Name	Console input screen
Problem Statement	1. System must track different types of personnel (counselors, educators, volunteers) 2. System must manage various youth activities and resources 3. Console implementation must demonstrate Abstract classes, Interfaces, Basic inheritance

3 Constraints

3.1 Class Requirements

1. \`Person\` Abstract Class:
   - Abstract attributes: id, name, role
   - Class Variables: person_count
   - Abstract methods: display_info(), perform_duty()
   - Example: Cannot be instantiated directly

2. `ISchedulable` Interface:
   - Methods: schedule(date, time), is_available(date, time)
3. `ICertified` Interface:
   - Methods: verify_certification(), get_certification_details()
4. `Counselor` Class (extends `Person`, implements `ISchedulable`, `ICertified`):
   - Additional attributes: specialization, case_load
   - Override methods: All required methods
   - Example: `Counselor("C001", "Emma Smith", "behavioral")`
5. `Educator` Class (extends `Person`, implements `ISchedulable`, `ICertified`):
   - Additional attributes: subject, education_level
   - Override methods: All required methods
   - Example: `Educator("E001", "John Davis", "mathematics")`
6. `Volunteer` Class (extends `Person`, implements `ISchedulable`):
   - Additional attributes: hours_completed, availability
   - Override methods: All required methods
   - Example: `Volunteer("V001", "Sara Johnson", "weekends")`
7. `YouthCenter` Class:
   - Attributes: name, personnel, activities
   - Methods: add_person(), remove_person(), find_person_by_id(), create_activity()
   - Example: `YouthCenter("BrightFuture Center")`

3.2 Operation Constraints

1. Basic Interactions:
   - Staff members perform duties based on their role
   - Counselors and educators must be certified
   - Volunteers track hours completed
2. Input Validation:
   - All persons IDs must be unique
   - Case load must be positive integer (1-20)

   - Hours completed must be non-negative

3. Exception Handling:
   - Must handle PersonNotFoundException
   - Must handle ScheduleConflictException
   - Must handle CertificationException

3.3 OUTPUT CONSTRAINTS

1. Display Format:
   - Person info must show: ID, name, role, type-specific attributes
   - YouthCenter info must show: name, personnel count by type, active activities

4. TEMPLATE CODE STRUCTURE:

1. Person Classes:
   - `Person` (abstract base class)
   - `ISchedulable` (interface)
   - `ICertified` (interface)
   - `Counselor` (concrete class)
   - `Educator` (concrete class)
   - `Volunteer` (concrete class)
2. Center Classes:
   - `YouthCenter`
3. Exception Classes:
   - `PersonNotFoundException`
   - `ScheduleConflictException`
   - `CertificationException`
4. Program Control:
   - `main()` - main program function

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. View the main menu
3. Select operations:

   - Option 1: Add Person
   - Option 2: Schedule Activity
   - Option 3: Display All Personnel
   - Option 4: Verify Certifications
   - Option 0: Exit
4. View results after each operation
5. Exit program when finished