# System Requirements Specification Index

For

# Military Training Camp Management System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

# TABLE OF CONTENTS

# Military Training Camp Management System

## System Requirements Specification

## 1 PROJECT ABSTRACT

The Military Training Camp Management System (MTCMS) will track personnel, training schedules, equipment, and performance evaluations while implementing robust encapsulation principles. The system emphasizes data protection through proper access modifiers, controlled attribute access, and role-based security to safeguard sensitive military information.

## 2 BUSINESS REQUIREMENTS:

| Screen Name | Console input screen |
|---|---|
| Problem Statement | 1. System must enforce strong encapsulation of sensitive military data<br>2. Different user roles must have appropriate access controls<br>3. All data modification must occur through validated property methods<br>4. Sensitive information must be protected using name mangling<br>5. System must demonstrate proper OOP principles for a military environment |

## 3 CONSTRAINTS

### 3.1 CLASS REQUIREMENTS

1. `MilitaryPersonnel` Abstract Base Class:
    - Private attributes: __security_clearance, __performance_records
    - Protected attributes: _id, _name, _rank, _unit
    - Property methods with validation

- Methods for both updating and retrieving security clearance

2. `Officer` Class (extends MilitaryPersonnel):
   - Private attributes: __command_code
   - Protected attributes: _specialization
   - Property methods with restricted information disclosure
   - Methods that demonstrate encapsulation of sensitive data
3. `Recruit` Class (extends MilitaryPersonnel):
   - Private attributes: __training_scores, __disciplinary_record
   - Protected attributes: _aptitude_ratings
   - Property methods with appropriate validation
   - Methods for receiving training scores from Officers
   - Override of rank setter to prevent rank changes
4. `TrainingProgram` Class:
   - Private attributes: __program_code, __performance_metrics
   - Protected attributes: _name, _duration, _requirements
   - Methods for updating training data based on user role
   - Property accessors that return copies of collections
5. `EquipmentInventory` Class:
   - Private attributes: __equipment (containing details like serial numbers, maintenance history)
   - Methods demonstrating role-based information disclosure
   - Different views of equipment based on requestor role

## 3.2 Encapsulation Constraints

1. Access Modifiers:
   - Private attributes must use double underscore prefix (__attribute)
   - Protected attributes must use single underscore prefix (_attribute)
   - Public methods must not expose sensitive data directly

2. Property Decorators:
   - All attribute access must be through @property decorators
   - All attribute modifications must use property setters with validation

- Example: `@property def rank(self): return self._rank`

3. Data Validation:
   - Setters must validate all inputs before assigning values
   - Security clearance must be checked before accessing sensitive data
   - Input validation must verify data types and value ranges
4. Information Hiding:
   - Direct access to private attributes should be prevented
   - Name mangling must be used for truly sensitive data
   - Classes should only expose necessary interfaces
5. Immutable Returns:
   - Methods returning collections must return copies, not references
   - This prevents unintended modification of internal state
   - Example: `return self.__performance_records.copy()`

### 3.3 Role-Based Access Requirements

1. Officer Role:
   - Access to personnel records
   - Ability to modify training programs
   - Access to all equipment details
   - Ability to update training scores for recruits
   - Authority to authorize security clearance changes
2. Recruit Role:
   - Access only to own basic records
   - No modification rights to training programs
   - Limited equipment details access (no serial numbers, maintenance history)
   - No ability to update training scores for others

### 3.4 Implementation Constraints

1. Class Structure:
   - Use inheritance for personnel hierarchy

- Implement abstract methods where appropriate
- Define clear interfaces between system components

2. Method Design:
   - Methods should check user role before performing operations
   - Methods returning sensitive data should provide only necessary information
   - Methods should raise appropriate exceptions for access violations:
     - AccessDeniedException: For unauthorized access attempts
     - InvalidDataException: For data validation failures

3. Test Compatibility:
   - Implement both `get_performance_records` and `get_performance_history` methods
   - Support both `update_security_clearance(level, officer)` method and `security_clearance = (level, officer)` setter

## 4. TEMPLATE CODE STRUCTURE:

**1.** Base Class:

- `MilitaryPersonnel` (abstract)

**2.** Personnel Classes:

- `Officer`
- `Recruit`

**3.** Program Management:

- `TrainingProgram`
- `EquipmentInventory`

**4.** System Control:

- `CampManagementSystem`

## 5. EXECUTION STEPS TO FOLLOW:

1. Create a new system instance and personnel
2. Attempt to access attributes directly (should fail for private/protected)
3. Demonstrate property access with different user roles
4. Show validation working for improper inputs
5. Demonstrate proper data hiding when displaying information

6. Show how encapsulation prevents data corruption through immutable returns
7. Demonstrate role-based access producing different views of the same data
8. Show authorization checks preventing unauthorized operations