Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to specify the software requirements for the Entity Management System,

which allows users to manage basic entities characterized by a name and an area attribute.

The system demonstrates basic object modeling, data persistence, and user interaction.

1.2 Scope

This system:

- Accepts entity details from the user.
- Stores entity records in a persistent file (data.json).
- Allows viewing of all stored records.

The core component of this system is the class Charan, which models each entity's data.

1.3 Definitions, Acronyms, and Abbreviations

- CLI: Command-Line Interface
- JSON: JavaScript Object Notation (used for data storage)
- Charan: Main class representing entity data

2. Overall Description

2.1 Product Perspective

The product is a standalone command-line tool. It does not depend on external APIs or databases.

It uses JSON files for persistent storage.

2.2 Product Functions

- Add a new entity (Charan object)
- View all entities
- Save data to data.json
- Load data from data.json

2.3 User Characteristics

Users are expected to be familiar with basic command-line operations.

2.4 Constraints

- Data is stored locally in a file called data.json.
- No GUI is provided.
- No authentication is required.

3. Specific Requirements

3.1 Functional Requirements

1. Add Entity
   - Input: Name, Area
   - Action: Create a Charan object and store it.
   - Output: Confirmation message.

2. View Entities
   - Display all Charan records.

3. Save Data

- Serialize all Charan objects and write to data.json.

4. Load Data

- Read from data.json and reconstruct Charan objects.

4. Class Specification

4.1 Class: Charan

Attributes:

- name: string

- area: float

Methods:

- to_dict(): Serializes the object to dictionary format.

- from_dict(data): Static method to deserialize a dictionary into a Charan object.

5. External Interface Requirements

5.1 User Interfaces

- CLI-based menu:

  - 1. Add Entity

  - 2. View Entities

  - 3. Exit

5.2 Hardware Interfaces

- Runs on standard personal computers.

5.3 Software Interfaces

- Python 3 interpreter

- File system access to read/write data.json

6. Non-Functional Requirements

- Usability: Simple CLI interface

- Reliability: Handles missing files gracefully

- Maintainability: Code is modular and readable

- Portability: Runs on any OS with Python installed

7. Appendix

- main.py: Contains CLI and flow logic

- data.json: File to persist records