A white car charging at a charging station

Description automatically generated

Electric Vehicle Population

Data Management System

Design Document

Prepared by: Babu

Last Updated: 11/10/2024

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author/s** | |
| 11/10/2024 | 0.1 | Initial Version | Babu |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[1 Introduction 2](#_Toc182395741)

[2 Requirements 2](#_Toc182395742)

[3 Solution Architecture 3](#_Toc182395743)

[3.1 High-level Services 3](#_Toc182395744)

[3.1.1 On the Cloud 3](#_Toc182395745)

[4 Technology Stack 3](#_Toc182395746)

[5 Project Layout 3](#_Toc182395747)

[5.1 Structure 3](#_Toc182395748)

[5.2 DTO Classes 4](#_Toc182395749)

[5.3 Repository Layer 4](#_Toc182395750)

[5.4 Services Layer 4](#_Toc182395751)

[5.5 Controllers Layer 4](#_Toc182395752)

[5.6 Exception Handling 4](#_Toc182395753)

[5.7 Unit and Integration Testing 4](#_Toc182395754)

[5.8 Logging and Observability 4](#_Toc182395755)

[5.9 Helm Chart for Deployment 4](#_Toc182395756)

[5.10 Documentation 4](#_Toc182395757)

[6 Solution Deployment 4](#_Toc182395758)

[7 Appendix 5](#_Toc182395759)

[7.1 Data Overview 5](#_Toc182395760)

[8 FAQs 5](#_Toc182395761)

[9 Confidentiality of Contents 5](#_Toc182395762)

# Introduction

Java Spring Boot application with design principles applied to build a scalable, maintainable REST API for managing Electric Vehicle Population Data.

*Note: This is just a place holder for now, check developer notes.*

# Requirements

This section captures requirements for each sub-system of Electric Vehicle Population Data System.

REQ-1: REST API to expose CRUD

REQ-2: Use RDS

Note: Rest of the requirements omitted from pdf document as stated in the project guide.

# Solution Architecture

In this section we will define solution architecture.

// TBD

## High-level Services

First, let’s *identify cloud services* that will be leveraging in implementation.

// TBD

### On the Cloud

// TBD

# Technology Stack

|  |  |  |
| --- | --- | --- |
| **#** | **Technology** | **Description** |
| 1 | Core Framework | Spring Boot2 |
| 2 | Persistent Layer Framework | Spring Data JPA |
| 3 | Database | PgSQL |
| 4 |  |  |

// TBD

# Project Layout

// TBD

## Structure

Following the principles of Separation of Concerns and Dependency Injection, here’s an organized project structure:

## DTO Classes

Use the DTO pattern to separate database representation (entity) from what the API exposes (DTO). Annotate entities with JPA annotations and DTOs with validation constraints.

## Repository Layer

Use Spring Data JPA repository interfaces for database operations.

## Services Layer

The Services layer encapsulates business logic and coordinates between the Controller and Repository layers.

## Controllers Layer

Controllers handle REST Calls from clients, forwarding them to services and transforming service responses for the client.

## Exception Handling

Implementing global exception handling with error messes provides clear cause of error.

## Unit and Integration Testing

Using JUnit and MockMVC for unit tests on controllers.

## Logging and Observability

Integrating SLF4J with a logging framework for logging with OpenTelemetry for metrics and traces.

## Helm Chart for Deployment

Helm charts are used to deploy services, and config-map for external db connection info.

## Documentation

Design documentation & REST API’s are included as part of solution distribution.

# Solution Deployment

TBD – This section will be updated soon.

# Appendix

TBD – This section will be updated soon.

## Data Overview

TBD – This section will be updated soon.

# FAQs

TBD – This section will be updated soon.

# Confidentiality of Contents

The contents of this document are shared only under NDA and should not be reattributed.