**Software Requirements Specification (SRS) Document**

**Product Name**: Schema Verification and Data Warehouse Table Generation Tool  
**Version**: 1.0  
**Prepared By**: [Your Name/Organization]  
**Date Created**: April 13, 2025

**Table of Contents**

1. Introduction
   * 1.1 Purpose
   * 1.2 Intended Audience and Reading Suggestions
   * 1.3 Product Scope
   * 1.4 References
2. Overall Description
   * 2.1 Product Perspective
   * 2.2 Product Functions
   * 2.3 User Classes and Characteristics
   * 2.4 Operating Environment
   * 2.5 Design and Implementation Constraints
   * 2.6 Assumptions and Dependencies
3. System Features and Requirements
   * 3.1 Functional Requirements
   * 3.2 Nonfunctional Requirements
4. External Interface Requirements
5. Other Requirements

**1. Introduction**

**1.1 Purpose**

This document specifies the requirements for a web-based application that assists data engineers in verifying database schemas, generating data warehouse tables, and creating migration scripts for offline use with future cloud scalability.

**1.2 Intended Audience and Reading Suggestions**

The intended audience includes:

* **Developers**: For understanding implementation details.
* **Data Engineers**: For using schema verification and table generation features.
* **Project Managers**: For tracking progress.
* **Testers**: For validating functionality.

**1.3 Product Scope**

The product will:

* Allow users to configure databases (SQL Server, PostgreSQL, MySQL, etc.).
* Fetch database schemas and auto-populate tables.
* Enable schema analysis using an offline LLM.
* Generate data warehouse tables and migration scripts.
* Provide integration options for Whareescape or other vendors in the future.

**1.4 References**

* IEEE SRS Template[1](https://dspmuranchi.ac.in/pdf/Blog/srs_template-ieee.pdf).
* Python libraries documentation for pyodbc, psycopg2, and mysql.connector.

**2. Overall Description**

**2.1 Product Perspective**

This product is a standalone tool designed for small organizations with an offline-first approach, scalable to cloud environments later.

**2.2 Product Functions**

* Database configuration and connection.
* Schema fetching and table selection.
* Schema analysis using an LLM.
* Data warehouse table generation.
* Migration script creation.

**2.3 User Classes and Characteristics**

* **Data Engineers**: Primary users configuring databases and generating scripts.
* **Administrators**: Managing user access and configurations.

**2.4 Operating Environment**

* Backend: Python (FastAPI/Flask).
* Frontend: React with Vite.
* Deployment: Docker containers (ports starting from 7000).

**2.5 Design and Implementation Constraints**

* Use only open-source libraries.
* Offline LLM integration (no external API calls).
* Modular plug-and-play architecture.

**2.6 Assumptions and Dependencies**

* Users will have access to database credentials.
* Docker will be used for deployment.

**3. System Features and Requirements**

**3.1 Functional Requirements**

1. Allow users to register/login securely (using JWT authentication).
2. Support database configuration for SQL Server, PostgreSQL, MySQL, etc., without requiring users to specify drivers explicitly.
3. Fetch databases under the configured connection automatically.
4. Auto-populate tables from the selected database.
5. Analyze schemas using an offline LLM based on user prompts.
6. Generate data warehouse tables based on schema analysis.
7. Create migration scripts without reading actual data for privacy reasons.
8. Provide error handling and logging mechanisms for debugging.

**3.2 Nonfunctional Requirements**

1. **Performance**:
   * Response time for schema fetching should be <5 seconds for typical databases (<100 tables).
   * LLM-based analysis should complete within <10 seconds for standard schemas.
2. **Security**:
   * Encrypted storage of sensitive credentials (e.g., passwords).
   * Secure user authentication using industry standards.
3. **Scalability**:
   * Modular architecture to support future integrations with Whareescape or other tools.
4. **Usability**:
   * Intuitive UI/UX design with clear workflows for database configuration, schema selection, and script generation.

**4. External Interface Requirements**

**User Interfaces:**

The web-based UI will allow users to:

* Configure databases via forms (host, username, password, etc.).
* View fetched schemas in a tabular format.
* Input prompts for schema analysis.

**Hardware Interfaces:**

No specific hardware requirements; the tool will run on any system supporting Docker.

**Software Interfaces:**

Supports integration with SQL Server (pyodbc), PostgreSQL (psycopg2), MySQL (mysql.connector).

**5. Other Requirements**

**Logging:**

Implement robust logging using Python's logging or structlog libraries to capture all errors and user actions.

**Error Handling:**

Provide detailed error messages with actionable suggestions during failures (e.g., invalid credentials or unsupported database types).

This SRS document provides a comprehensive roadmap for building the schema verification and data warehouse table generation tool while adhering to open-source principles, modularity, and scalability goals suitable for small organizations transitioning to cloud environments in the future.

**Citations:**

1. <https://dspmuranchi.ac.in/pdf/Blog/srs_template-ieee.pdf>
2. <https://github.com/jam01/SRS-Template>
3. <https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document>
4. <https://exinfm.com/training/M2C3/srs_template.doc>
5. <https://asana.com/resources/software-requirement-document-template>
6. <https://www.overleaf.com/latex/templates/software-requirements-specification/ryktphvctsxg>
7. <https://bit.ai/templates/software-requirements-document-template>
8. <https://nicsi.com/pbd/files/NICSI-PBD-SRS%20Template%20830-1998%20-13082019-V1.0.pdf>

Answer from Perplexity: <pplx.ai/share>