**AUTO-DASH: Automatic Dashboard Generator**

**High-Level Design Document**

**1. Introduction**

**1.1 Purpose**

This document outlines the high-level design for AUTO-DASH, an automatic dashboard generator system that creates customized Dash applications based on user-provided CSV data.

**1.2 Scope**

AUTO-DASH aims to streamline the process of creating interactive dashboards by leveraging AI to generate code based on data insights.

**1.3 Definitions, Acronyms, and Abbreviations**

- CSV: Comma-Separated Values

- AI: Artificial Intelligence

- UI: User Interface

- API: Application Programming Interface

**2. System Overview**

AUTO-DASH is a web-based application that allows users to upload CSV files, analyzes the data, and generates a custom Dash dashboard using AI-powered code generation.

**3. System Architecture**

**3.1 Architectural Diagram**

[Insert a high-level architectural diagram here]

**3.2 Component Description**

**1. User Interface (UI)**

- Technology: Streamlit

- Purpose: Provides a web interface for users to interact with the system

**2. Data Loader**

2.1 Should read data from user input from streamlit.

**3. Data Analyzer**

- Input: Preprocessed DataFrame

- Output: Data insights dictionary

- Purpose: Extracts key insights and statistics from the data

**4. Prompt Builder**

- Input: Data insights

- Output: Formatted prompt string

- Purpose: Constructs a prompt for the AI model based on data insights

**5. Dashboard Generator**

- Input: Formatted prompt

- Output: Python code for Dash application

- Purpose: Interfaces with AI API to generate dashboard code

**6. Code Executor**

- Input: Generated Python code

- Output: Running Dash application

- Purpose: Saves and executes the generated Dash application code

**4. Data Flow**

1. User uploads CSV file through the Streamlit UI

2. Data Loader processes the CSV file

3. Data Analyzer extracts insights from processed data

4. Prompt Builder creates AI prompt using data insights

5. Dashboard Generator sends prompt to AI and receives Dash code

6. Code Executor saves and runs the generated Dash application

7. User interacts with the generated dashboard

**5. External Interfaces**

**5.1 User Interfaces**

- Streamlit web interface for file upload and system interaction

- Generated Dash dashboard for data visualization

**5.2 APIs**

- Claude 3.5 Sonnet API for code generation

**6. Technologies and Frameworks**

- Python 3.9+

- Pandas for data processing

- Streamlit for main user interface

- Dash and Plotly for dashboard creation and visualization

**7. Security Considerations**

- Data privacy: Ensure user data is not stored or transmitted unnecessarily

- API security: Secure handling of AI API keys

- Input validation: Sanitize and validate all user inputs

**Version-1 Release Notes**

**How to bulid using Auto-Dash for your organisation.**

**(The purpose of this is to create internal dashboards for lightning-fast insights.)**

1. Improve the reference.py file. This file may need to be customized for each company to align with their preferred style of reporting. Some organizations prefer tables over charts, or vice versa. This will give you flexiblity is creating a better example for your Organisation.