High-Level Design (HLD)

1. High-Level Design (HLD)

1.1 Introduction

- **Objective:** To develop a machine learning system capable of classifying mushrooms as either edible or poisonous based on their physical characteristics and to expose this system's functionality through a scalable web API.
- **Problem Statement:** The misidentification of wild mushrooms poses a significant health risk, including poisoning and death. An automated and accurate classification system is necessary to assist in identifying mushrooms, thereby minimizing the potential for human error.
- **Scope:** The system will function as a backend REST API. It will accept a mushroom's features in a JSON format and return a prediction (edible or poisonous) along with a confidence score. The project scope does not include a graphical user interface (GUI).

1.2 System Architecture

The system is designed with a decoupled architecture, separating the offline model training from the online prediction service.

1. **Client:** Any application or user (e.g., Postman, a mobile app, a web front-end) that consumes the API by sending POST requests.
2. **Cloud Platform (Google Cloud Run):** A fully managed, serverless platform that automatically runs and scales the stateless prediction service container based on incoming traffic. It handles HTTPS termination, load balancing, and instance management.
3. **Prediction Service (Docker Container):** A self-contained Docker image that packages the application and all its dependencies. It consists of:
    - **Gunicorn Server:** A production-ready WSGI HTTP server to run the Flask application.
    - **Flask Application:** A Python-based micro-framework that defines the /predict API endpoint and handles request/response logic.
    - **ML Model:** A pre-trained Scikit-learn model loaded into memory for fast predictions.
4. **Offline Training Pipeline:** A separate process (typically a Python script) responsible for:
    - Loading the raw mushrooms.csv dataset.
    - Performing data cleaning and preprocessing.
    - Training the machine learning model.
    - Serializing the trained model and required artifacts (e.g., model.joblib, model_columns.joblib).

1.3 Technology Stack

- **Programming Language:** Python 3.9
- **Machine Learning:** Scikit-learn, Pandas
- **API Framework:** Flask
- **WSGI Server:** Gunicorn
- **Containerization:** Docker
- **Cloud Platform:** Google Cloud Platform (Cloud Run, Artifact Registry)