**Devices Price Classification System Documentation**

**Introduction:**

This documentation provides an overview of the Devices Price Classification System developed using Python and Spring Boot. The system predicts the prices of devices based on their specifications using machine learning models and provides RESTful API endpoints for interaction.

**Contents:**

1. Overview
2. Project Structure
3. Python Project - Machine Learning Model
4. Running the Application
5. Interacting with API Endpoints

**1. Overview:**

1. The Devices Price Classification System predicts the price range of devices based on their specifications such as battery power, camera quality, and screen resolution. It consists of two main components:

* Python project: Develops machine learning models to predict device prices and provides a Flask application for serving RESTful API endpoints.
* Spring Boot project: Implements RESTful API endpoints to interact with the machine learning models.

**2. Project Structure:**

* Python Project:
  + model.ipynb: Jupyter Notebook containing the machine learning model development process, including data preprocessing, model selection, and evaluation.
  + pipeline.pkl: Serialized machine learning model saved for deployment.
  + app.py: Flask application serving as the RESTful API for interaction with the machine learning model.
* Spring Boot Project:
  + Device.java: Entity class representing a device with specifications.
  + DeviceController.java: REST controller defining API endpoints for device operations.
  + DeviceService.java: Service class implementing business logic for device operations.
  + DeviceRepository.java: Repository interface for database operations.

**3. Python Project - Machine Learning Model:**

* Data Preprocessing: The dataset is preprocessed to handle missing values and engineer new features such as total pixels and screen area.
* Model Selection: Grid search is performed over multiple machine learning algorithms, including Support Vector Machine (SVM), Random Forest, and Decision Tree, to select the best-performing model.
* Evaluation: Evaluation metrics such as accuracy, precision, recall, and F1-score are used to assess the model's performance.
* Pipeline Creation: A pipeline is created with StandardScaler for feature scaling and the selected SVM model for prediction.
* Serialization: The pipeline is serialized and saved as pipeline.pkl for deployment.

**4. Running the Application:**

* Python Project:
  1. Run the Jupyter Notebook model.ipynb to train the machine learning model and serialize it.
  2. Execute app.py to start the Flask application serving the RESTful API.

**5.Interacting with API Endpoints:**

**RESTful Endpoints:**

1. /predict:

Method: POST

Description: Predicts the price range of a device based on its specifications.

Request Body: JSON object containing device specifications.

Response: JSON object with predicted price range.

1. /api/devices/:

Method: POST

Description: Retrieves a list of all devices.

Response: JSON object containing a list of device details.

1. /api/devices/{id}:

Method: GET

Description: Retrieves details of a specific device by its ID.

Response: JSON object containing device details.

1. /api/devices:

Method: POST

Description: Adds a new device.

Request Body: JSON object containing device specifications.

Response: JSON object confirming the addition of the device