**Product Scraper and Database Handler**

**Overview**

This project involves scraping product data from various online retailers and storing it in a MongoDB database. It includes scraping for different product categories, such as graphic cards and processors, from multiple websites. The data is saved and managed using the MongoDBHandler class, which handles database operations efficiently.

## Prerequisites

* Python
* MongoDB
* Required Python libraries:
  + requests
  + beautifulsoup4
  + pymongo
  + selenium

## Installation

1. **Clone the repository.**
2. **Install the required libraries:**

pip install -r requirements.txt

1. **Ensure MongoDB is running on your local machine.**

## Usage

1. **Run all scrapers:**

To execute all the scrapers and save the data into MongoDB, run the following command:

python main.py

This command will execute all individual scrapers (zestro, xtreme, rbtech, czone, jtech, and paklap) and save the data into the MongoDB database.

1. **Customize MongoDB settings:**

Update the MongoDB connection settings in the MongoDBHandler class if your MongoDB instance is hosted on a different server or port.

## Scripts Overview

### Scrapers

1. **zestro.py**
   * **Functionality:** Scrapes graphic cards and processors data from the Zestro website.
   * **Functions:**
     + getGraphicCards(db\_handler): Scrapes graphic card data.
     + getProcessors(db\_handler): Scrapes processor data.
     + zestroRunAll(): Runs the scraping functions and saves data to the database.
2. **xtreme.py**
   * **Functionality:** Scrapes graphic cards and processors data from the Xtreme website.
   * **Functions:**
     + getGraphicCards(db\_handler): Scrapes graphic card data.
     + getProcessors(db\_handler): Scrapes processor data.
     + xtremeRunAll(): Runs the scraping functions and saves data to the database.
3. **rbtech.py**
   * **Functionality:** Scrapes graphic cards and processors data from the RBTech website.
   * **Functions:**
     + getGraphicCards(db\_handler): Scrapes graphic card data.
     + getProcessors(db\_handler): Scrapes processor data.
     + rbtechRunAll(): Runs the scraping functions and saves data to the database.
4. **czone.py**
   * **Functionality:** Scrapes graphic cards and processors data from the CZone website.
   * **Functions:**
     + getGraphicCards(db\_handler): Scrapes graphic card data.
     + getProcessors(db\_handler): Scrapes processor data.
     + czoneRunAll(): Runs the scraping functions and saves data to the database.
5. **jtech.py**
   * **Functionality:** Scrapes graphic cards and processors data from the JTech website.
   * **Functions:**
     + getGraphicCards(db\_handler): Scrapes graphic card data.
     + getProcessors(db\_handler): Scrapes processor data.
     + jtechRunAll(): Runs the scraping functions and saves data to the database.
6. **paklap.py**
   * **Functionality:** Scrapes graphic cards data from the PakLap website.
   * **Functions:**
     + getGraphicCards(db\_handler): Scrapes graphic card data.
     + paklapRunAll(): Runs the scraping function and saves data to the database.

### Database Handler

**mongodb.py**

* **Functionality:** Handles all interactions with the MongoDB database.
* **Class:**
  + MongoDBHandler
    - **Methods:**
      * \_\_init\_\_(self, uri, db\_name, collection\_name): Initializes the MongoDB connection.
      * saveToDB(self, name, category, images, titles, prices, details, links): Saves or updates product data in the database.
      * close\_connection(self): Closes the MongoDB connection.

### Main Execution

**main.py**

* **Functionality:** Runs all the scrapers sequentially.
* **Function:**
  + run\_all\_scrapers(): Executes all individual scrapers and handles exceptions.