Project Documentation

1. Introduction

This project integrates MongoDB and Google Sheets, allowing you to perform CRUD operations on MongoDB and interact with Google Sheets. The application is built in Python and utilizes the ‘pymongo’ and ‘gspread’ libraries.

2. Setup and Requirements

a.Prerequisites

- Python 3 (Python 3.10.11 is used here)
- MongoDB installed and running (MongoDB Atlas used here)
- Google Sheets API credentials (creds.json should be generated)

b.Installation

- Clone the repository
- install as per requirement.txt

3. Project Structure

The project structure is organized as follows:

```
project-root/
|-- pymongo1.py
|-- sheets.py
|-- creds.json
|-- requirements.txt
|-- README.md
```

4. Connecting to MongoDB

```
conn_string = "mongodb+srv://<username>:<password>@<cluster-url>/<dbname>"
client = pymongo.MongoClient(conn_string)
myDB = client["test_db"]
```

To connect to MongoDB, replace <username>, <password>, <cluster-url>, and <dbname> with your MongoDB credentials in the conn_string variable in pymongo1.py

5. Google Sheets Integration

```
def authenticate_google_sheets():
    scope = ["https://spreadsheets.google.com/feeds",
'https://www.googleapis.com/auth/spreadsheets.readonly',
"https://www.googleapis.com/auth/drive.file",
"https://www.googleapis.com/auth/drive"]
    creds = ServiceAccountCredentials.from_json_keyfile_name("creds.json", scope)
    client = gspread.authorize(creds)
    return client.open("api_project").sheet1
```

Authentication
Ensure you have the creds.json file from Google Sheets API and replace it in the project directory. This file is used for authentication.

```
sheet = authenticate_google_sheets()
```

Usage
In pymongo.py, the authenticate_google_sheets function initializes the connection to Google Sheets. Replace "api_project" with your Google Sheets document name.

6. MongoDB CRUD Operations

Options:
1. View all documents
2. View a document by name
3. Insert a new document
4. Update a document by name
5. Delete a document by name
6. Delete all documents
7. Copy all data from sheet and store in MongoDB
8. Copy specific row from sheet and store in MongoDB
0. Exit
Enter your choice (0 to exit):

7. **Google Sheets Operations**

Options:
1. View all records
2. View a specific row
3. View a specific column
4. View a specific cell
5. Insert a new row
6. Update a row
7. Delete a row
0. Exit
Enter your choice (0 to exit):

Why Using MongoDB Atlas:

Managed Service:
Fully managed, no need for manual setup or maintenance.

Scalability:
Easily scales with automatic sharding for increased data load.

High Availability:
Built-in high availability with automated failover.

Backups and Recovery:
Regular backups and snapshots for data protection.

Security Features:
Network isolation, encryption at rest, and secure connections.

Global Deployment:
Allows deployment in various regions for lower-latency access.

Automated Updates:
Handles updates, patches, and maintenance tasks.

Cost-Effective:
Considered cost-effective for maintenance, scaling, and high availability.

Monitoring and Analytics:
Provides tools for monitoring and analytics dashboards.

Ease of Use:
User-friendly web interface for easy management.

Neural Network Model

1. The MNIST dataset is loaded using the mnist.load_data() method from TensorFlow.
2. Images are reshaped and normalized to values between 0 and 1.
3. A convolutional neural network (CNN) is constructed using TensorFlow's Keras API.
4. The model consists of convolutional layers with max-pooling, followed by fully connected layers.
5. The model is trained on the training dataset with a batch size of 64 for 5 epochs.

Configuration and Training

Neural Network Architecture:
A convolutional neural network (CNN) architecture is chosen for its effectiveness in image classification tasks.
Three convolutional layers with max-pooling are used, followed by two fully connected layers.

Normalization:
Input images are normalized to values between 0 and 1 to facilitate convergence during training.

Activation Functions:
ReLU (Rectified Linear Unit) is used as the activation function for convolutional layers, and softmax for the output layer.

Optimizer and Loss Function:
The Adam optimizer is chosen for its efficiency in handling sparse gradients.
Categorical cross entropy is selected as the loss function for multi-class classification.

Training Parameters:
The model is trained for 5 epochs with a batch size of 64.
A validation split of 20% is used to monitor the model's performance during training.