ISM6218 Advanced Database Management
FINAL PROJECT REPORT

Group 2	May 3, 2024

FNU Ashima, Nandini Malviya, Sai Sruthi Subraveti, Shanmukha Sree Sai Teja Damineni, Shrimanth Kavuluru

The Bulls Fiction Factory

MONGODB:

A robust, open-source NoSQL database, MongoDB provides an adaptable and scalable approach to data management and storage. MongoDB employs a document-oriented data format as opposed to conventional relational databases, storing data in adaptable documents that resemble JSON rather than rows and columns. Because of this, MongoDB is a great tool for managing unstructured or semi-structured data and enables quicker application development and iteration.

Flask & integrating Flask with MongoDB:

Flask is a versatile Python web framework, empowering developers to swiftly craft online applications with its suite of tools and patterns. It is simple, user-friendly, and adaptable, so a preferred choice for projects ranging from personal endeavors to large-scale enterprise systems.

Integration of MongoDB with Flask involves linking a Flask web application to a MongoDB database, facilitating seamless utilization of MongoDB as the backend storage solution.
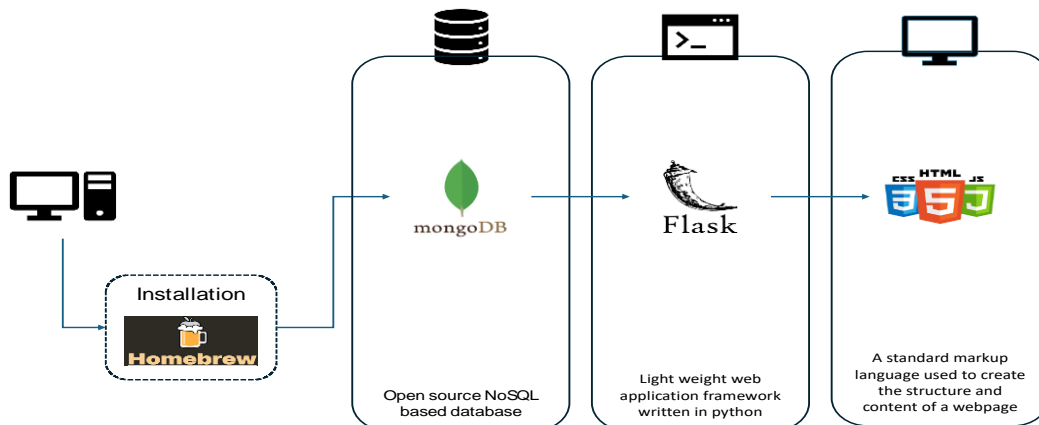
HTML Front end:

The user-facing interface of a web application is created with HTML, CSS, and JavaScript front end that is UI-based. With elements like <div>, <header>, and <footer> and HTML organizes content to define the UI's structure and layout. The user interface's design, layout, and responsiveness on various screens and devices are all controlled via CSS. JavaScript provides interactivity by enabling client-side operations, event handling, and dynamic behavior. When combined, these technologies give programmers the ability to design intuitive, dynamic, and aesthetically pleasing user interfaces that improve the user experience for users accessing web applications via a web browser.

Package Manager:

Homebrew is a package manager designed specifically for Linux and macOS systems. It provides users with a command-line interface that makes it simple to install and manage a variety of software packages, libraries, and utilities that are not installed by default on the operating system. By keeping software packages current and handling dependencies, this utility streamlines the installation process and improves user ease. In addition, it offers a centralized repository called the Homebrew Formula Repository, which makes user collaboration and package definition exchange easy.

From Installation to Development: The Homebrew Website Stack

Installation

Homebrew

Open source NoSQL based database

Light weight web application framework written in python

A standard markup language used to create the structure and content of a webpage

Database & Collection:

Database: A database in MongoDB is a logical container that stores groups of documents. It functions as a high-level data management and storage organizational unit. Multiple databases, each with its own collections and documents, can be hosted on a single server by MongoDB.
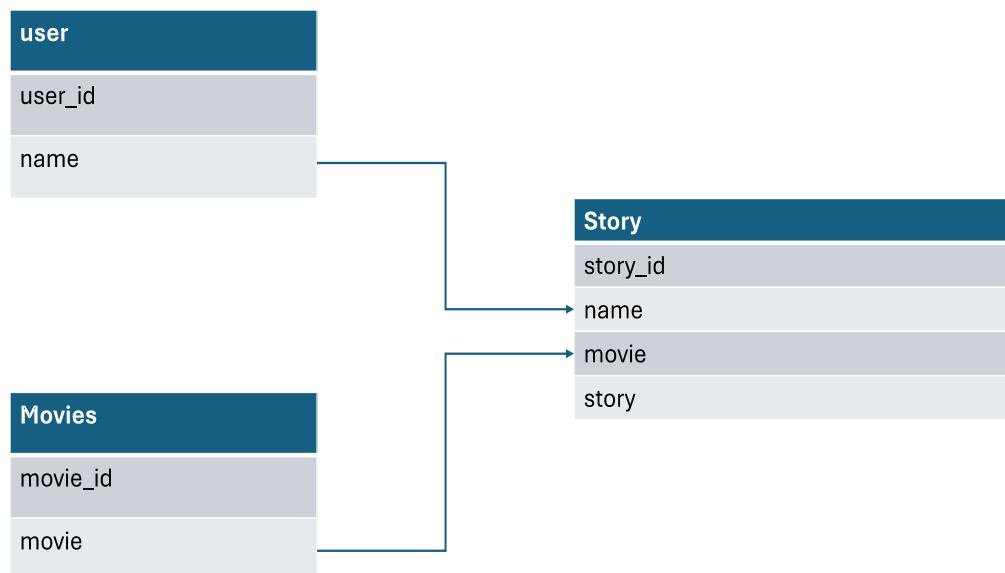
Collection:A MongoDB collection serves as a repository for a variety of documents, resembling a traditional database table but with a twist. Unlike rigid relational schemas, MongoDB collections embrace flexibility, accommodating diverse document structures and data types within a single entity.

Database & Collections used:

Database: bull_fitting

Collections:

- Movie
- User
- Story

user
user_id
name

Story
story_id
name
movie
story

Movies
movie_id
movie

Tables description:

User Table: This table stores username and an ID autogenerated by the server.

```
In [12]: import pymongo
         client = pymongo.MongoClient("mongodb://localhost:27017/")
         db = client["bull_fitting"]
         user_collection = db["user"]
         print(user_collection.full_name)
         print(user_collection.count_documents({}))
         documents = user_collection.find({})

         #print each documents
         for document in documents:
             print(document)

         bull_fitting.user
         2
         {'_id': ObjectId('663408aae8a9b320ccea97a4'), 'name': 'test 4'}
         {'_id': ObjectId('66340aa8e47cc4e5e4bce8d6'), 'name': 'Shanmukha'}
```

Movie Table: This table stores the name of the movie and an autogenerated ID.

Viewing the data in movie_collection

```
In [21]: import pymongo
         client = pymongo.MongoClient("mongodb://localhost:27017/")
         db = client["bull_fitting"]
         movie_collection = db["movies"]
         print(movie_collection.full_name)
         print(movie_collection.count_documents({}))
         documents = movie_collection.find({})

         #print each documents
         for document in documents:
             print(document)

         bull_fitting.movies
         1
         {'_id': ObjectId('66340aa8e47cc4e5e4bce8d7'), 'movie': 'Dune Part 1'}
```

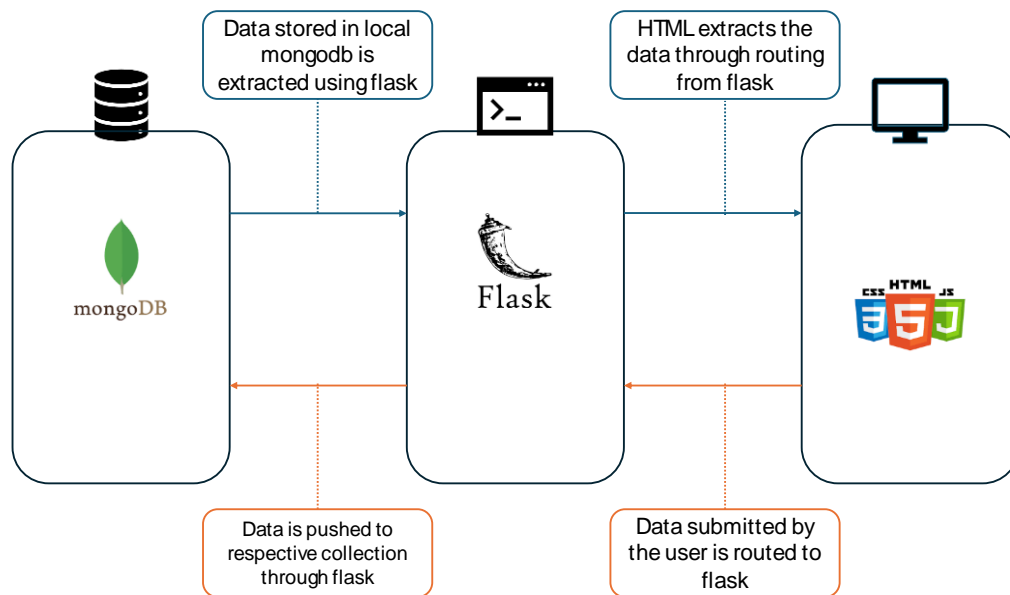Story Table: This table stores the username, movie name and their fictional story.

Viewing the data in story_collection

```python
import pymongo
client = pymongo.MongoClient("mongodb://localhost:27017/")
db = client["bull_fitting"]
story_collection = db["story"]
print(story_collection.full_name)
print(story_collection.count_documents({}))
documents = story_collection.find({})

#print each documents
for document in documents:
    print(document)
```

```
bull_fitting.story
1
{'_id': ObjectId('66340aa8e47cc4e5e4bce8d5'), 'name': 'Shanmukha', 'movie': 'Dune Part 1', 'story': "Dune: The Lost
Sister\r\nIn the midst of the intricate political landscape of Arrakis, a forgotten sibling emerged, shrouded in my
stery. Kaileen, the younger sister of Paul Atreides, had been secretly sent to the Bene Gesserit sisterhood on the
planet Wallach IX, hidden from the prying eyes of their enemies.\r\nTrained in the ancient arts of the Bene Gesseri
t, Kaileen possessed a unique combination of abilities, inherited from her mother, the Lady Jessica. Her presence w
as unknown to Paul, who believed she had perished in the escape from House Harkonnen's treachery.\r\nAs Paul naviga
ted the treacherous landscape of Arrakis, leading the Fremen rebellion against the Empire, Kaileen began to experie
nce strange visions, hinting at a connection to her long-lost brother. The Bene Gesserit Mother Superior, sensing t
he significance of Kaileen's lineage, sent her on a perilous journey to Arrakis, to reunite with Paul and aid in th
e fight against the Emperor.\r\nUpon arrival, Kaileen encountered Stilgar, the Fremen leader, who was initially war
y of her intentions. However, her remarkable abilities and knowledge of the Bene Gesserit ways quickly earned his r
espect. As she navigated the complexities of Fremen culture and politics, Kaileen discovered a deep connection to t
he native people and their struggle.\r\nAs Paul and Kaileen finally reunited, their combined strength and prescienc
e proved formidable. Together, they forged a powerful alliance, leveraging the Bene Gesserit's secrets and the Frem
en's knowledge of the desert to challenge the Empire's grip on Arrakis.\r\nThe story concludes with a tantalizing h
int of the upcoming battle, as the siblings prepare to face their greatest challenge yet: the confrontation with th
e ruthless Emperor Shaddam IV. The fate of Arrakis and the future of the Atreides dynasty hung in the balance, as t
he lost sister emerged as a powerful force in the struggle for control of the spice.", 'created_on': datetime.datet
ime(2024, 5, 2, 17, 50, 32, 372000)}
```

Data Extraction with Flask:

Data stored in local mongodb is extracted using flask

HTML extracts the data through routing from flask

Data is pushed to respective collection through flask

Data submitted by the user is routed to flask

How The Bulls Fiction Factory Works:

Web Page: This website allows for the creation and sharing of fan fiction works. Users can contribute anonymously or using a specified username. The portal allows users to pick the movie they are writing a fan fiction narrative for before submitting their work.

Name:

Movie:

Fan POV: Tell us your story?

Submit

Movie Search

Search Movie: _____ [Search] [View All] [Delete Entries]

Name:
Sree

Movie:
Life

Fan POV: Tell us your story?

After the defeat of the alien entity, Calvin and the remaining crew members returned to Earth as heroes. However, their relief was short-lived, as they soon discovered a strange, glowing pod on the International Space Station. Inside, they found an infant alien, the last of its kind. As they raised the creature, they realized it possessed advanced intelligence and abilities. The team named it "Echo" and worked to understand its powers, hoping to use them for humanity's benefit. But as Echo grew, so did concerns about its intentions, leaving the crew wondering if they had made a new friend or a new threat.

Submit

What happens after submitting the fan fiction:

The corresponding data is systematically categorized into distinct collections using the flask routing mechanism.

CRUD Operations:

Insert:

Code:

The insert_one() function inserts the data into the database, and the flask code returns the data in the collection to be presented on the website in decreasing order (last in, first displayed).

```python
@test_fan_fic_v3.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        # If the form is submitted, handle the form data as before
        name = request.form.get('name')
        movie = request.form.get('movie')
        story = request.form.get('story')
        current_datetime = datetime.now()
        created_on = int(current_datetime.timestamp())

        story_document = {
            "name": name,
            "movie": movie,
            "story": story,
            "created_on": current_datetime
        }
        user_document = {
            "name": name
        }
        movie_document = {
            "movie": movie
        }

        story_collection.insert_one(story_document)
        user_collection.insert_one(user_document)
        movie_collection.insert_one(movie_document)

        return redirect(url_for('index'))
    else:
        # If the request method is GET, retrieve all data from MongoDB
        data = list(story_collection.find().sort("created_on", -1))

        return render_template('Fan_Fic_main.html', data=data)
```

Search Results:

Name	Movie	Story
Sree	Life	After the defeat of the alien entity, Calvin and the remaining crew members returned to Earth as heroes. However, their relief was short-lived, as they soon discovered a strange, glowing pod on the International Space Station. Inside, they found an infant alien, the last of its kind. As they raised the creature, they realized it possessed advanced intelligence and abilities. The team named it "Echo" and worked to understand its powers, hoping to use them for humanity's benefit. But as Echo grew, so did concerns about its intentions, leaving the crew wondering if they had made a new friend or a new threat.
Shanmukha	Dune Part 1	Dune: The Lost Sister In the midst of the intricate political landscape of Arrakis, a forgotten sibling emerged, shrouded in mystery. Kaileen, the younger sister of Paul Atreides, had been secretly sent to the Bene Gesserit sisterhood on the planet Wallach IX, hidden from the prying eyes of their enemies. Trained in the ancient arts of the Bene Gesserit, Kaileen possessed a unique combination of abilities, inherited from her mother, the Lady Jessica. Her presence was unknown to Paul, who believed she had perished in the escape from House Harkonnen's treachery. As Paul navigated the treacherous landscape of Arrakis, leading the Fremen rebellion against the Empire, Kaileen began to experience strange visions, hinting at a connection to her long-lost brother. The Bene Gesserit Mother Superior, sensing the significance of Kaileen's lineage, sent her on a perilous journey to Arrakis, to reunite with Paul and aid in the fight against the Emperor. Upon arrival, Kaileen encountered Stilgar, the Fremen leader, who was initially wary of her intentions. However, her remarkable abilities and knowledge of the Bene Gesserit ways quickly earned his respect. As she navigated the complexities of Fremen culture and politics, Kaileen discovered a deep connection to the native people and their struggle. As Paul and Kaileen finally reunited, their combined strength and prescience proved formidable. Together, they forged a powerful alliance, leveraging the Bene Gesserit's secrets and the Fremen's knowledge of the desert to challenge the Empire's grip on Arrakis. The story concludes with a tantalizing hint of the upcoming battle, as the siblings prepare to face their greatest challenge yet: the confrontation with the ruthless Emperor Shaddam IV. The fate of Arrakis and the future of the Atreides dynasty hung in the balance, as the lost sister emerged as a powerful force in the struggle for control of the spice.

Search: The user can search the data by using the movie name.

Code:

```python
@test_fan_fic_v3.route('/search', methods=['POST'])
def search():
    query = request.form.get('query')

    # Query MongoDB collection based on the user's search query
    search_data = list(story_collection.find({'movie': {'$regex': query, '$options': 'i'}}).sort("created_on", -1))

  # Fetch all data from MongoDB if there are no search results
    if not search_data:
        data = list(story_collection.find().sort("created_on", -1))
        #return render_template('Fan_Fic_main.html', data=data)

    return render_template('Fan_Fic_main.html', data=search_data)
```


Movie Search

Search Movie: Dune   Search   View All   Delete Entries

Search Results:

Name	Movie	Story
Shanmukha	Dune Part 1	Dune: The Lost Sister In the midst of the intricate political landscape of Arrakis, a forgotten sibling emerged, shrouded in mystery. Kaileen, the younger sister of Paul Atreides, had been secretly sent to the Bene Gesserit sisterhood on the planet Wallach IX, hidden from the prying eyes of their enemies. Trained in the ancient arts of the Bene Gesserit, Kaileen possessed a unique combination of abilities, inherited from her mother, the Lady Jessica. Her presence was unknown to Paul, who believed she had perished in the escape from House Harkonnen's treachery. As Paul navigated the treacherous landscape of Arrakis, leading the Fremen rebellion against the Empire, Kaileen began to experience strange visions, hinting at a connection to her long-lost brother. The Bene Gesserit Mother Superior, sensing the significance of Kaileen's lineage, sent her on a perilous journey to Arrakis, to reunite with Paul and aid in the fight against the Emperor. Upon arrival, Kaileen encountered Stilgar, the Fremen leader, who was initially wary of her intentions. However, her remarkable abilities and knowledge of the Bene Gesserit ways quickly earned his respect. As she navigated the complexities of Fremen culture and politics, Kaileen discovered a deep connection to the native people and their struggle. As Paul and Kaileen finally reunited, their combined strength and prescience proved formidable. Together, they forged a powerful alliance, leveraging the Bene Gesserit's secrets and the Fremen's knowledge of the desert to challenge the Empire's grip on Arrakis. The story concludes with a tantalizing hint of the upcoming battle, as the siblings prepare to face their greatest challenge yet: the confrontation with the ruthless Emperor Shaddam IV. The fate of Arrakis and the future of the Atreides dynasty hung in the balance, as the lost sister emerged as a powerful force in the struggle for control of the spice.

- User Input: The user enters a movie title, which the Flask application accepts as input. This film's title is referred to as the "query" throughout the process.
- Matching Movies: Flask uses a lookup function (perhaps called find) to search for movies based on the user's input. This function looks for movies that match the user's criteria and produces a list of possible matches.
- Search Results: The lookup function's list of movies is then stored in a variable called "search_data."
- Displaying Results: The HTML code receives the "search_data" containing a list of potential movie matches. This enables HTML to display the search results on the user's webpage.

```html
<div class="searchResults">
    <!--Search results will be displayed here -->
    {% if data %}
    <h2 style="color: ■white;">Search Results:</h2>
    <table class="movie-table">
        <thead>
            <tr>
                <th>Name</th>
                <th>Movie</th>
                <th>Story</th>
            </tr>
        </thead>
        <tbody>
            {% for item in data %}
            <tr>
                <td>{{ item.name }}</td>
                <td>{{ item.movie }}</td>
                <td>{{ item.story }}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
    {% else %}
    <p>No results found.</p>
    {% endif %}
</div>
```

View: A functionality to display all the data after the required information is gathered from a search.


```
@test_fan_fic_v3.route('/refresh', methods=['POST'])
def refresh():
    data = list(story_collection.find().sort("created_on", -1))

    return render_template('Fan_Fic_main.html', data=data)
```

Delete: A functionality to delete the data from the collection. The button leads the user (in our case the admin) to a pop up where they can choose, the combination of movie and user to delete from the collection. The admin can choose the movie and user combo to delete the data. The admin can multiselect and delete the selected entries.

Before:

After: The data is only deleted from the 'story' collection in the database, the movie and the user details are still stored in the 'user' and 'movie' collection.

Search Results:

Name	Movie	Story
Sree	Life	After the defeat of the alien entity, Calvin and the remaining crew members returned to Earth as heroes. However, their relief was short-lived, as they soon discovered a strange, glowing pod on the International Space Station. Inside, they found an infant alien, the last of its kind. As they raised the creature, they realized it possessed advanced intelligence and abilities. The team named it "Echo" and worked to understand its powers, hoping to use them for humanity's benefit. But as Echo grew, so did concerns about its intentions, leaving the crew wondering if they had made a new friend or a new threat.
Shanmukha	Dune Part 1	Dune: The Lost Sister In the midst of the intricate political landscape of Arrakis, a forgotten sibling emerged, shrouded in mystery. Kaileen, the younger sister of Paul Atreides, had been secretly sent to the Bene Gesserit sisterhood on the planet Wallach IX, hidden from the prying eyes of their enemies. Trained in the ancient arts of the Bene Gesserit, Kaileen possessed a unique combination of abilities, inherited from her mother, the Lady Jessica. Her presence was unknown to Paul, who believed she had perished in the escape from House Harkonnen's treachery. As Paul navigated the treacherous landscape of Arrakis, leading the Fremen rebellion against the Empire, Kaileen began to experience strange visions, hinting at a connection to her long-lost brother. The Bene Gesserit Mother Superior, sensing the significance of Kaileen's lineage, sent her on a perilous journey to Arrakis, to reunite with Paul and aid in the fight against the Emperor. Upon arrival, Kaileen encountered Stilgar, the Fremen leader, who was initially wary of her intentions. However, her remarkable abilities and knowledge of the Bene Gesserit ways quickly earned his respect. As she navigated the complexities of Fremen culture and politics, Kaileen discovered a deep connection to the native people and their struggle. As Paul and Kaileen finally reunited, their combined strength and prescience proved formidable. Together, they forged a powerful alliance, leveraging the Bene Gesserit's secrets and the Fremen's knowledge of the desert to challenge the Empire's grip on Arrakis. The story concludes with a tantalizing hint of the upcoming battle, as the siblings prepare to face their greatest challenge yet: the confrontation with the ruthless Emperor Shaddam IV. The fate of Arrakis and the future of the Atreides dynasty hung in the balance, as the lost sister emerged as a powerful force in the struggle for control of the spice.

Viewing the data in movie_collection

```
In [24]: import pymongo
         client = pymongo.MongoClient("mongodb://localhost:27017/")
         db = client["bull_fitting"]
         movie_collection = db["movies"]
         print(movie_collection.full_name)
         print(movie_collection.count_documents({}))
         documents = movie_collection.find({})

         #print each documents
         for document in documents:
             print(document)

         bull_fitting.movies
         3
         {'_id': ObjectId('66340aa8e47cc4e5e4bce8d7'), 'movie': 'Dune Part 1'}
         {'_id': ObjectId('66341e02e47cc4e5e4bce8da'), 'movie': 'Life'}
         {'_id': ObjectId('66342422e47cc4e5e4bce8dd'), 'movie': 'John Wick 4'}
```

Why MongoDB:

Key Features of MongoDB:

- Document-Oriented: MongoDB uses data structures resembling JSON to store data in documents. Since each document can have its own schema, established schemas are not necessary and data modelling can be more flexible.

- Scalability: MongoDB is made to grow horizontally, which lets you split up data over several servers to manage heavy traffic loads and massive data volumes. It is compatible with replica sets, which offer fault tolerance and high availability, and sharding, which divides data among several servers.
- Excellent Performance: MongoDB provides excellent reading and writing performance. It supports a few query types, including range queries, geospatial queries, and text search, and optimizes query efficiency using in-memory caching and automatic indexing.
- Rich Query Language: CRUD (Create, Read, Update, Delete) operations, aggregation, and data manipulation is just a few of the many functions supported by MongoDB's robust query language. For effective data retrieval, it also enables full-text search and secondary indexes.
- Flexible Data architecture: Complex data structures can be stored and retrieved with ease thanks to MongoDB's flexible data architecture. Documents are perfect for applications with changing data requirements because they allow you to store sophisticated data types like arrays and nested objects.

Future iterations:

This fan fiction platform can be improved by:

- User Accounts: Allow logins for saving drafts, ownership, and future community features.
- OMDB API: Integrate automatic movie details and wider movie selection.
- Reviews/Comments: Enable user interaction with ratings, comments, and moderation.

These features can create a more engaging and dynamic user experience.

Limitations of the Current Fan Fiction Platform

The current iteration of the fan fiction platform has some limitations due to its dependency on a local server and a free MongoDB database.

- Limited Scalability: Local servers often have less processing power and storage capacity than cloud-based solutions. As the user base and number of stories increase, the local server may struggle to meet demand.
- Accessibility: The platform would only be available on the local machine or within a restricted local network. Users would be unable to access it from anywhere else.
- Maintenance: Keeping a local server running involves technical knowledge and constant upkeep, such as software upgrades and security patches.
- Storage Constraint: MongoDB's free tier only provides 500MB of storage. This can easily fill up, particularly if the stories include photographs or other huge media assets. To accommodate additional data, you will need to upgrade to a premium plan. Performance

and resource constraints are common among free levels. As the user base expands, the platform may face slowdowns or capacity limits.

- Limited Features: Free tiers may not have all the functionality found in paying plans. Essential functionality like user management and sophisticated search may be blocked.
- Cloud Hosting: Moving the platform to a cloud-based solution, such as Google Cloud Platform or Amazon Web Services, would improve scalability, accessibility, and potentially reduce maintenance costs.
- Upgrading to a premium MongoDB plan increases storage capacity and improves performance.
- Alternative Databases: Look at other NoSQL databases that may offer a free tier with more storage space or functionalities that are more suited to your needs.

References:

- **MongoDB - https://www.mongodb.com/docs/manual/**

- **Home Brew - https://brew.sh/**

- **Archive of our own (AO3) - https://archiveofourown.org/**

Appendix

Code (flask):

```
from flask import Flask, render_template, request, redirect, url_for, send_from_directory

from datetime import datetime

from pymongo import MongoClient

from bson import ObjectId

import os

test_fan_fic_v3 = Flask(__name__)

# Connect to MongoDB

client = MongoClient("mongodb://localhost:27017/")

db = client["bull_fitting"]
```

```python
user_collection = db["user"]

story_collection = db["story"]

movie_collection = db["movies"]

@test_fan_fic_v3.route('/', methods=['GET', 'POST'])

def index():

if request.method == 'POST':

# If the form is submitted, handle the form data as before

name = request.form.get('name')

movie = request.form.get('movie')

story = request.form.get('story')

current_datetime = datetime.now()

created_on = int(current_datetime.timestamp())

story_document = {

"name": name,

"movie": movie,

"story": story,

"created_on": current_datetime

}

user_document = {

"name": name

}

movie_document = {

"movie": movie
```

```python
        }

        story_collection.insert_one(story_document)

        user_collection.insert_one(user_document)

        movie_collection.insert_one(movie_document)

        return redirect(url_for('index'))

    else:

        # If the request method is GET, retrieve all data from MongoDB

        data = list(story_collection.find().sort("created_on", -1))

        return render_template('Fan_Fic_main.html', data=data)

@test_fan_fic_v3.route('/search', methods=['POST'])

def search():

    query = request.form.get('query')

    # Query MongoDB collection based on the user's search query

    search_data = list(story_collection.find({'movie': {'$regex': query, '$options':
    'i'}}).sort("created_on", -1))

    # Fetch all data from MongoDB if there are no search results

    if not search_data:

        data = list(story_collection.find().sort("created_on", -1))

        #return render_template('Fan_Fic_main.html', data=data)

    return render_template('Fan_Fic_main.html', data=search_data)

@test_fan_fic_v3.route('/refresh', methods=['POST'])

def refresh():

    data = list(story_collection.find().sort("created_on", -1))

    return render_template('Fan_Fic_main.html', data=data)
```

```python
@test_fan_fic_v3.route('/delete', methods=['POST'])

def delete():

# Get the IDs of documents to delete

ids_to_delete = request.form.getlist('document_id')

# Delete the selected documents

for document_id in ids_to_delete:

story_collection.delete_one({"_id": ObjectId(document_id)})

data = list(story_collection.find().sort("created_on", -1))

return render_template('Fan_Fic_main.html', data=data)

@test_fan_fic_v3.route('/image/<path:filename>')

def download_file(filename):

return
send_from_directory('/Users/tjdamineni/Desktop/Grad_Work/ADBMS/flask/hello_flask/static',
filename)

if __name__ == '__main__':

test_fan_fic_v3.run(debug=True)
```