Precision Fitness Instruction System

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science Engineering

by

MELISSA MARIA AUGUSTINE

20BCE2543

Under the guidance of

Prof. Sivakumar V

SCOPE

VIT, Vellore.

June, 2024

DECLARATION

I here-by declare that the thesis entitled “Precision Fitness Instruction System” submitted by us, for the award of the degrees of Bachelor of Technology in Computer Science and Engineering in Vellore Institute of Technology, Vellore is a record of bonafide work carried out by us under the supervision of Prof. Sivakumar V.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

CERTIFICATE

This is to certify that the thesis entitled “Precision Fitness Instruction System” submitted by **Melissa Maria Augustine 20BCE2543**, School of Computer Science and Engineering, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by them under my supervision during the period, 05. 01. 2024 to 19. 06. 2024, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and meets the necessary standards for submission.

Place: Vellore

Date: 19/06/2024 **Signature of the Guide**

Internal Examiner **External Examiner**

ACKNOWLEDGEMENTS

It is our pleasure to express our extreme gratitude to Prof. Sivakumar V., our guide, the School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, for his constant guidance, encouragement, and understanding; we obtained a lot of knowledge from him through this endeavor. Our association with him on this thesis has given us an excellent opportunity to work with an experienced professor in the field of Artificial Intelligence.

We would also like to extend our thanks to Dr. G. Viswanathan, our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G V Selvam, our esteemed Vice President, and Dr. Sandhya Pentareddy, our Executive Director for providing an exceptional working environment and inspiring all of us during the tenure of the course. We would also like to express our gratitude towards our Dean, Dr. Ramesh Babu K, and our Head of Department, Dr. Umadevi K S for their valuable support.

Place: Vellore

Date: 19/06/2024

Student Name:

Melissa Maria Augustine

EXECUTIVE SUMMARY

My project involves developing a software system that leverages a large language model, such as GPT-3.5, to evaluate personal attributes and generate customized workout routines. The software considers various user characteristics, including age, gender, fitness level (beginner, intermediate, advanced), strengths, focus areas, favorite exercises, least favorite exercises, and weaknesses. By analyzing these attributes, the language model assesses the user's profile and matches it to a vector database containing various exercise videos. The system then curates and stitches together these videos to create a workout routine tailored to the user’s specific needs and goals. This personalized approach ensures that users receive a workout plan that aligns with their preferences and fitness objectives, making the exercise experience more effective and enjoyable.

List of Figures

1. INTRODUCTION

1.1 OBJECTIVES

1. **Personalized Fitness Solutions:** Create a system that designs workout routines just for you, taking into account your age, gender, fitness level, strengths, weaknesses, and exercise preferences.

2. **Integration with a Vector Database:** Build a way to match your fitness details to a database of exercise videos, so you get a variety of workouts that fit your needs perfectly.

3. **Adaptive and Dynamic Results:** Make sure the software can adjust to your progress and change fitness goals, keeping your workouts fresh and challenging.

4. **Promoting accessible and Regular Exercise**: Encourage you to stick with your fitness routine by giving you workouts that are enjoyable and tailored to your likes and fitness level.

1.2 MOTIVATION

The motivation behind this project stems from the need for more personalized and engaging fitness solutions. Many people struggle to stick to generic workout plans that don't cater to their individual needs and preferences. By leveraging advanced technology, such as AI large language models and vector databases, to analyze personal attributes, this project aims to create workout routines that are tailored to each user's unique characteristics, including their age, fitness level, strengths, and preferences. The goal is to make fitness more accessible and enjoyable, promoting regular exercise and helping users achieve their goals more effectively. Additionally, the project seeks to demonstrate the practical application of large language models in enhancing personal health and wellness, showcasing how AI can be used to create meaningful, customized experiences. This approach not only improves user satisfaction but also contributes to the broader field of personalized health and fitness technology.

1.3 BACKGROUND

The background of this project lies in the growing demand for personalized fitness solutions and the advancements in artificial intelligence. Traditional workout plans often fail to address individual needs, leading to poor adherence and less effective results. With the rise of AI and vector databases, there's an opportunity to revolutionize how fitness routines are created. These models can analyze detailed personal information to generate tailored workout plans that consider age, gender, fitness level, strengths, and preferences. Additionally, the integration of AI with vector databases of exercise videos offers a dynamic and comprehensive approach to fitness. This project aims to bridge the gap between generic fitness advice and highly personalized workout experiences, leveraging AI to make fitness more engaging, effective, and accessible for everyone. By combining advanced technology with user-specific data, the project aspires to set a new standard in personalized fitness solutions, promoting better health outcomes and enhancing the overall exercise experience.

2. LITERATURE SURVEY

2.1 SURVEY ON EXISTING SYSTEM

The paper "Virtual Fitness Trainer using Artificial Intelligence" explores the development of an AI-driven fitness application utilizing Python, OpenCV, and MediaPipe to offer personalized workout experiences. The application tracks user movements via computer vision, providing real-time feedback on form and technique. Machine learning algorithms tailor fitness recommendations based on individual goals and abilities, ensuring a customized fitness journey. The system enhances user engagement by offering immediate feedback, progress tracking, and reducing injury risk, thereby improving overall workout effectiveness and user satisfaction

The paper "PersonalFit: Fitness App with Intelligent Plan Generator" by Tiago Oliveira, Diogo Leite, and Goreti Marreiros discusses the development of an intelligent fitness application aimed at providing dynamic and personalized workout plans. The app comprises three main modules: an intelligent plan generator, an Android application, and a communication API. The intelligent plan generator uses clustering techniques to adapt training plans based on user feedback and historical data from similar users. By leveraging data mining, the system dynamically updates exercise routines to optimize results. The Android app is designed to be user-friendly, allowing users to easily register, access, and provide feedback on their training plans. The communication module, built on a RESTful architecture and hosted on Microsoft Azure, facilitates the interaction between the app and the server. The paper highlights the system's ability to personalize fitness plans effectively, but also notes the need for further improvements and real-world testing to enhance its functionality and user experience

The paper "Enhancing Fitness Training with AI" discusses the development of an AI-based virtual trainer using computer vision technology. This system employs a camera to capture users' movements during exercises and provides real-time feedback to improve their form and technique. The AI model analyzes the captured data to detect deviations from correct form and suggests corrective actions. Additionally, the system offers personalized workout plans and tracks users' progress over time, promoting healthy habits and enhancing workout effectiveness. The use of OpenCV for image processing ensures high accuracy and execution

speed. The system is designed to be scalable and accessible, suitable for both gym newcomers and home trainers. It integrates machine learning libraries like TensorFlow and PyTorch, enhancing its computer vision capabilities. The paper concludes that this AI virtual trainer can significantly improve the fitness experience by providing personalized, interactive, and effective guidance during workout

The paper "AI Gym Trainer Using Artificial Intelligence" discusses the development of an AI-based personal trainer designed to monitor and guide users through fitness exercises. Utilizing MediaPipe's "BlazePose" module and OpenCV for real-time human pose estimation, the system captures and analyzes users' movements to provide immediate feedback on exercise form and technique. It aims to improve user engagement, reduce the risk of injury, and ensure correct exercise execution. The system calculates body angles and compares them with standard postures, giving corrective guidance as needed. The AI trainer supports a variety of exercises and tracks metrics such as repetitions, speed, and completion percentage. This approach addresses the limitations of human trainers by offering cost-effective, personalized, and scalable fitness training solutions that can be used in both professional and amateur settings

2.2: EXISTING GAPS

I. Data Requirements and Generalization:

Many systems, such as those using Convolutional Neural Networks (CNNs), require large amounts of labeled training data, which can be time-consuming and expensive to collect. This limits the model’s effectiveness and generalizability to real-world scenarios where such extensive datasets might not be available.

II. Complexity and Usability:

The complexity of AI models, especially CNNs, makes them difficult to understand and optimize, particularly for non-experts. This complexity can also hinder troubleshooting and refining the models.

III. Real-Time Processing and Performance:

While some systems like OpenPose are effective for real-time human pose estimation, they often require high computational resources, such as GPUs, to function efficiently. This can make them less accessible for everyday users who may not have high-end hardware.

IV. Feedback and Interaction Quality:

Current AI fitness trainers provide feedback on exercise form and technique but may lack the personalized encouragement and motivational support that human trainers offer. This can impact user engagement and adherence to workout routines.

V. Accuracy and Reliability:

Ensuring the accuracy and reliability of the AI systems is crucial, especially in providing real-time feedback. Any inaccuracies in pose detection or form correction could potentially lead to injuries or ineffective workouts. Rigorous testing and validation are necessary to address these concerns.

2.3 PROBLEM STATEMENT

Existing solutions face several critical challenges, including high data requirements, complexity, real-time processing limitations, and a lack of comprehensive user engagement and personalization. To address these gaps, we propose developing an AI-driven fitness training system that leverages vector databases for faster data retrieval and provides dynamic video outputs for enhanced user engagement. This system will integrate advanced machine learning algorithms with real-time feedback mechanisms to offer highly personalized workout plans and corrective guidance. By utilizing vector databases, the system will efficiently manage and query vast amounts of exercise data, ensuring rapid and accurate responses. Additionally, incorporating comprehensive user data from wearable devices and other inputs will enhance the accuracy and reliability of exercise monitoring and feedback.

3. TECHNICAL SPECIFICATION

3.1: Requirements

3.1.1: Functional requirements

1. User Input Handling: The system collects and verifies user details crucial for personalized fitness planning, like age, gender, and fitness level. It also gathers information on strengths, weaknesses, focus areas, and preferences for exercises. Validating these inputs ensures accurate workout recommendations tailored to each user's needs.

2. Exercise Video Database Interaction: To recommend personalized workouts, the system interacts with an exercise video database. It queries the database to fetch exercise videos based on user attributes and preferences. The database filters and sorts videos by muscle groups, difficulty levels, and available equipment, aiding users in finding suitable exercises.

3. Personalization and Recommendation: The system customizes workout routines based on user inputs. It uses gathered data to suggest exercises that match the user's strengths, weaknesses, and fitness goals. It offers varied routines to adapt to changes in user fitness levels and preferences over time.

4. Workout Plan Generation: The system automatically creates detailed workout plans tailored to each user. It arranges exercise sequences, sets repetitions, and defines rest intervals. Users can modify these plans by selecting preferred exercises or adjusting routines to better suit their fitness preferences and goals.

5. Testing and Validation: Thorough testing ensures the system accurately recommends exercises based on user inputs. It verifies robustness across various scenarios and user profiles. Continuous user feedback informs ongoing improvements to enhance system accuracy and user satisfaction

3.1.2: Non-functional requirements

1. Performance: The system ensures quick response times for user inputs and maintains performance as user and database interactions scale up.

2. Reliability: It maintains high availability with minimal downtime and robust error handling to recover from unexpected issues promptly.

3. Security: User data is securely stored, and all communications are encrypted to protect privacy and ensure data integrity.

4. Usability: The interface is intuitive for easy user input, workout plan viewing, and exercise video access, meeting accessibility standards.

5. Performance: It maintains swift response times and scalability to handle increased user interactions and database queries effectively.

3.2: Feasibility

3.2.1: Technical Feasibility

Developing the software involves ensuring we have skilled developers and the right technology stack to handle tasks like video stitching, database integration, and user interface design. It's crucial to guarantee the system can scale with increasing users and maintain fast response times. Security measures must also be robust to protect user data and system integrity.

3.2.2: Economic feasibility

From a financial standpoint, we need to consider initial development costs, ongoing operational expenses like hosting and maintenance, and infrastructure costs for data management. Revenue could come from subscriptions, advertising, or partnerships with fitness brands. Assessing market demand and potential returns is essential to ensure

profitability and justify investment.

3.3.3: Social feasibility

Socially, the software's success hinges on user adoption and engagement. It must cater to current fitness trends, offer a user-friendly interface, and be accessible to diverse user demographics. Promoting health and wellness through personalized fitness plans and ensuring ethical data privacy practices will enhance user trust and adoption.

4. DESIGN APPROACH

4.1 System Architecture

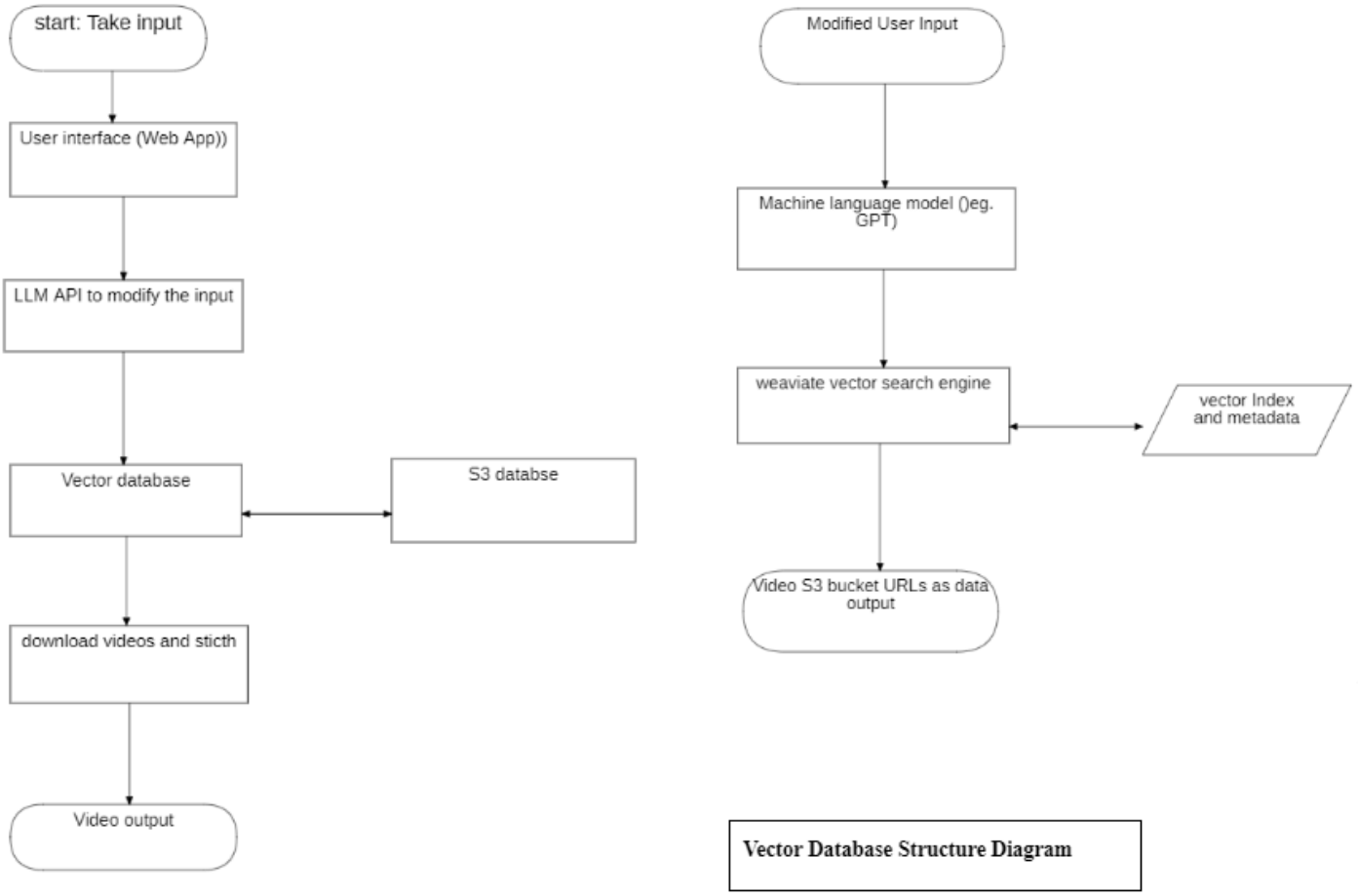

Fig 1: System Architecture of Precision Fitness Instruction System

The system architecture involves several key components working together to deliver personalized fitness recommendations. At its core is Weaviate, a vector database designed to store and retrieve complex data representations known as vectors. These vectors represent various attributes of exercises, user preferences, and fitness goals. Weaviate's ability to efficiently handle vector-based queries makes it well-suited for storing and retrieving exercise data based on user attributes.

Integrating with Weaviate is LangChain, powered by the OpenAI API (GPT-3.5). LangChain

enhances the system by modifying and optimizing the output generated by GPT-3.5 to align with Weaviate's vector database format. This integration ensures that the natural language descriptions or queries provided by users are transformed into structured vectors that Weaviate can effectively process.

Overall, the architecture enables seamless interaction between user inputs, the natural language processing capabilities of GPT-3.5 through LangChain, and the efficient storage and retrieval of exercise data in Weaviate. This approach not only facilitates personalized workout recommendations based on user attributes and preferences but also enhances the scalability and responsiveness of the system to meet varying user needs.

4.2 Design

4.2.1: Data Flow Diagram

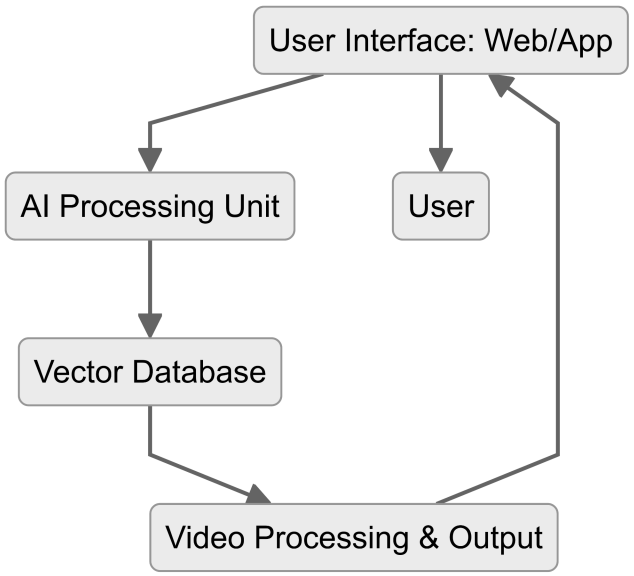

Fig 2: Data Flow Diagram

4.2.2: Sequence Diagram

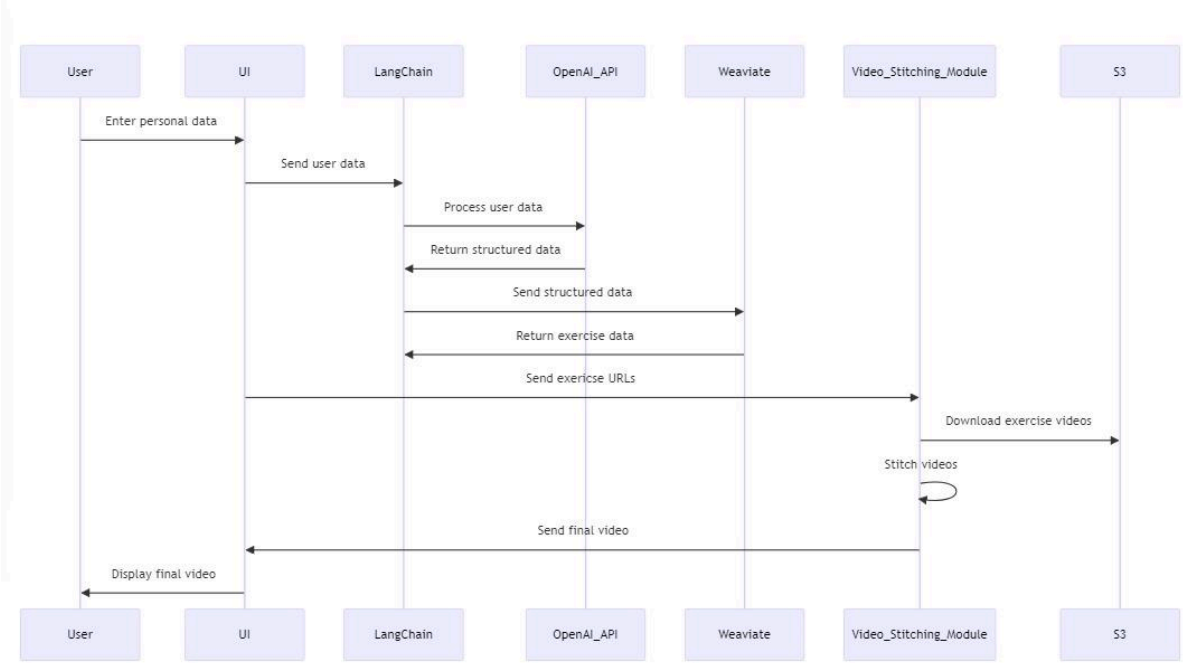

Fig 3: Sequence Diagram

4.2.3: Use Case Diagram

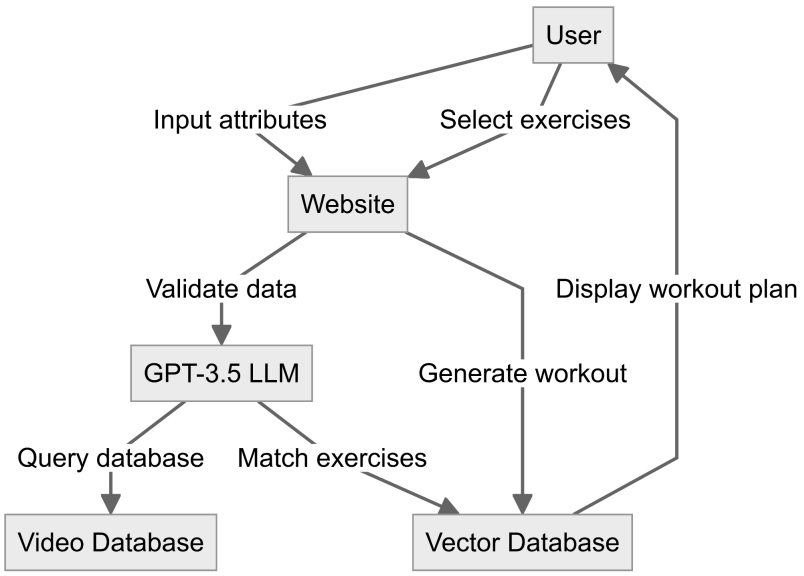

Fig 4: Use Case Diagram

4.2.4: Class Diagram of user and databases involved

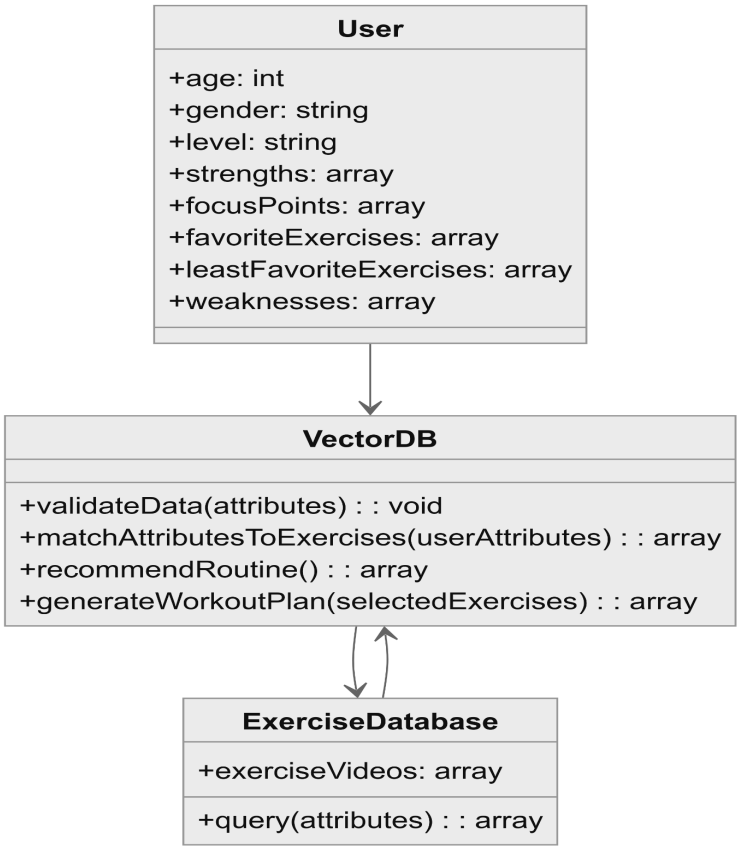

Fig 5: Class Diagram

4.3: Constraints, Alternatives and Tradeoffs

Constraints:

The project faces technical challenges in integrating Weaviate and LangChain effectively, requiring expertise in aligning data formats and ensuring seamless communication. It must achieve fast response times and scalability while handling large datasets and user requests, with a focus on robust security measures to safeguard user data throughout Weaviate and LangChain interactions. Economically, the project must manage costs for software development, hosting, and maintenance within budget constraints.

Alternatives:

Exploring alternatives includes considering different vector databases or storage solutions if

Weaviate's integration or performance poses challenges. Evaluating alternative natural language processing APIs or models beyond GPT-3.5 could offer cost-effective or more suitable solutions for processing user queries and generating personalized recommendations. These alternatives provide flexibility in adapting to evolving technical requirements and optimizing system performance based on user feedback and emerging technologies.

Trade-offs:

Balancing performance with cost optimization is crucial, especially in improving video stitching and database update times, which may increase expenses. Simplifying integration processes for faster deployment versus maintaining extensive capabilities for future scalability involves tradeoffs between complexity and flexibility. Balancing data privacy with personalized recommendations requires careful consideration. Strict privacy measures may restrict data collection, affecting personalization depth and user satisfaction. Achieving the right balance ensures the system meets both functional and non-functional requirements effectively.

5. SCHEDULE, MODULES AND TESTING

5.1: Gantt Chart

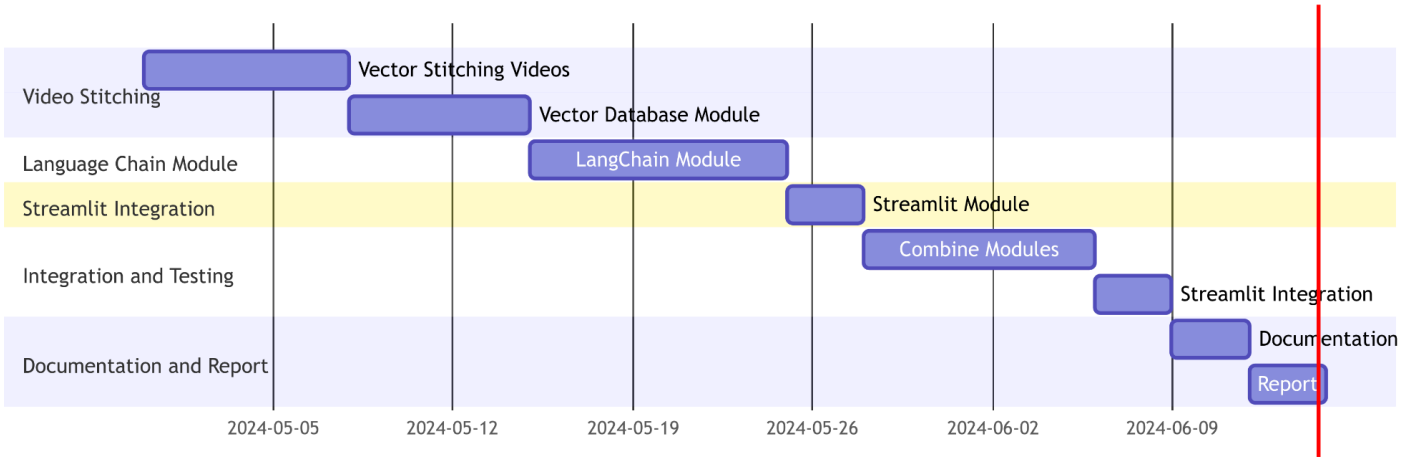

Fig 6: Gantt Chart

Module 1: User Interface (UI):

The User Interface (UI) module is where users interact with the system. It allows users to input their age, gender, fitness level, and exercise preferences. The UI displays personalized workout recommendations and exercise videos based on these inputs. It provides an easy-to-use interface where users can navigate through different options, select preferred exercises, and view detailed workout plans. The UI plays a crucial role in ensuring that users can easily access and benefit from the personalized fitness recommendations generated by the system.

Module 2: LangChain (OpenAI API, GPT-3.5):

LangChain utilizes the OpenAI API, specifically GPT-3.5, for natural language processing tasks. It transforms user inputs, such as queries or descriptions of fitness goals, into structured data that can be understood and processed by the system. This module enhances the system's ability to interpret and respond to user interactions effectively. By converting natural language into vector representations compatible with Weaviate, LangChain ensures that user preferences and attributes are accurately reflected in the system's recommendations and searches.

Module 3: Weaviate (Vector Database):

Weaviate serves as the core database for storing and retrieving vector-based data representations. It stores vectors that encode information about exercises, user profiles, and fitness preferences. Weaviate's strength lies in its ability to efficiently handle complex queries and provide rapid access to relevant data. By utilizing vector-based storage, it optimizes the retrieval of personalized workout recommendations based on user attributes. Weaviate ensures that the system can deliver accurate and tailored fitness plans by leveraging its robust indexing and querying capabilities.

Loading json file of exercises and descriptions in to weaviate code:

```
!pip install "weaviate-client==3.*"
import weaviate
client = weaviate.Client(
    url="https://capstone20bce2543-jxnrz4sa.weaviate.network",  # 
Replace with your Weaviate endpoint

auth_client_secret=weaviate.auth.AuthApiKey(api_key="LE9JY76cgiiSgNBC45
1tt7lZCXdCF2DaalNd"),  # Replace with your Weaviate instance API key
    additional_headers={
        'X-OpenAI-Api-key': 
"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"# Replace with 
your third party API key and identifying header
    },
    timeout_config=(5, 15)
)

class_obj = {
    "class": "Sciii",
    "vectorizer": "text2vec-openai",  # If set to "none" you must 
always provide vectors yourself. Could be any other "text2vec-*" also.
    "moduleConfig": {
        "text2vec-openai": {},
        "generative-openai": {}  # Ensure the `generative-openai` 
module is used for generative queries
    }
}

client.schema.create_class(class_obj)
import requests
```

```
import json

# Path to the JSON file
file_path = '/content/exercisesssf.json'

# Open the file and load the data
with open(file_path, 'r') as file:
    data = json.load(file)['exercises']  # Load data and directly
access the 'exercises' list

# Assuming 'client.batch' and 'batch.add_data_object' are part of a
defined class and context
client.batch.configure(batch_size=100)  # Configure batch
with client.batch as batch:  # Initialize a batch process
    for i, d in enumerate(data):  # Batch import data from the
'exercises' list
        print(f"importing question: {i+1}")
        properties = {
            "title": d["Title"],
            "description": d["Desc"],
            "type": d["Type"],
            "bodyPart": d["BodyPart"],
            "equipment": d["Equipment"],
            "level": d["Level"],
            "rest_time": d["rest_time"],
            "video_s3_url": d["video_s3_url"]
        }
        batch.add_data_object(
            data_object=properties,
            class_name="Sciii"
        )
```

Module 4: Exercise Video Database(S3):

The S3 Database bucket module houses a collection of exercise videos categorized by muscle groups, difficulty levels, and required equipment. It integrates seamlessly with the system to provide visual demonstrations of recommended exercises. Users can view these videos to understand proper exercise techniques and forms. This module enriches the user experience by combining personalized workout plans with visual guides, enhancing user engagement and effectiveness in following the recommended fitness routines.

Module 5: Vector Stitching Module:

The Vector Stitching Module processes and combines vectors that represent user attributes and exercise characteristics. It ensures that the vectors accurately capture relevant information needed for personalized workout planning. By stitching together these vectors, the module facilitates the generation of comprehensive workout plans tailored to each user's specific needs and preferences. This process enables the system to provide customized recommendations that align closely with user-defined fitness goals and capabilities, enhancing the overall effectiveness and relevance of the workout plans generated.

Module 6: Streamlit Integration Module:

The Streamlit Integration Module integrates various system components into a cohesive web application interface using Streamlit, a Python library. It provides a user-friendly environment where personalized workout plans and an exercise video are presented in an accessible and interactive format. This module ensures seamless navigation and interaction within the system, enabling users to easily access and benefit from the personalized fitness recommendations generated by the system.

5.3: Unit and Integration Testing

Unit Testing Implementation:

Unit testing ensures that each module of the system works correctly on its own. For example, the User Interface (UI) module's unit tests check if input forms and navigation buttons function as expected. LangChain’s unit tests verify that it properly transforms user queries into structured data. Weaviate's unit tests validate its ability to store and retrieve exercise data accurately. Each module, like the Vector Stitching Module, is tested independently to ensure its algorithms produce correct outputs.

Integration Testing Implementation:

Integration testing combines all system modules into cohesive functions to check how they work together. It tests interactions between modules, such as UI and LangChain, to verify that user inputs are processed correctly. For instance, testing LangChain and Weaviate integration ensures data transformations align with database requirements. Integration tests also validate

overall system functionality by simulating user interactions from inputting data to receiving personalized workout plans. This approach ensures that all components collaborate seamlessly to deliver a reliable and cohesive user experience.

6. PROJECT DEMONSTRATION

The project is aimed at developing a personalized fitness software that integrates advanced technologies to tailor workout routines based on individual user profiles and preferences. This software harnesses the power of AI to analyze user inputs and generate customized exercise plans, enhancing user experience and effectiveness in achieving fitness goals. Here's a breakdown of the project's key components:

1. User Interface (UI) Module

At the heart of user interaction, the User Interface (UI) Module serves as the primary point of contact for users. It is designed to be straightforward and user-friendly, allowing users to easily input personal details such as age, gender, fitness level, and exercise preferences. The UI module is responsible for displaying personalized workout recommendations and exercise videos, facilitating easy navigation through various features and options. This ensures that users can effortlessly access and utilize the fitness plans tailored to their needs.

2. LangChain (OpenAI API, GPT-3.5) Module

The LangChain module utilizes the OpenAI API, specifically GPT-3.5, to process natural language inputs. This involves transforming user queries or descriptions of fitness goals into structured data that the system can understand and act upon. By leveraging advanced natural language processing (NLP) techniques, this module enhances the system's ability to interpret user inputs accurately and provide relevant responses and recommendations, making the interaction as natural and intuitive as possible.

3. Weaviate (Vector Database) Module

Central to the system's data handling capabilities, the Weaviate module serves as a vector database that stores and manages data in vector form. This includes information about exercises, user profiles, and fitness preferences. Its powerful indexing and querying capabilities allow for efficient handling of complex searches, enabling the system to quickly retrieve personalized workout recommendations that are most relevant to the user's input. The use of vector storage ensures that data retrieval is both fast and accurate, supporting the delivery of tailored fitness plans.

4. Exercise Video Database (S3) Module

This module consists of a comprehensive database stored on S3, containing a wide variety of exercise videos. These videos are categorized by factors such as muscle groups targeted, difficulty level, and equipment needed. The Exercise Video Database is integrated into the system to provide users with visual demonstrations of recommended exercises. This not only helps users perform exercises correctly but also enriches the overall learning experience by showing proper techniques and postures.

5. Vector Stitching Module

The Vector Stitching Module plays a critical role in the personalization of workout plans. It processes and combines vectors that represent user attributes and exercise characteristics, ensuring that the generated vectors accurately encapsulate the necessary information for tailored recommendations. This sophisticated processing allows for the creation of comprehensive and customized workout plans that align with each user's specific fitness goals and preferences.

6. Streamlit Integration Module

Finally, the Streamlit Integration Module brings all other components together into a web application. Utilizing Streamlit(Python library), this module provides a streamlined and interactive interface where users can access their personalized workout plans and download the stitched exercise video. It ensures that all system components work seamlessly together, providing a smooth and engaging user experience that encourages regular use and helps users stay committed to their fitness journeys.

7. CONCLUSION AND FUTURE WORK

7. 1: Conclusion

In conclusion, the development of our personalized fitness software project has integrated advanced technologies like natural language processing with LangChain (OpenAI API, GPT-3.5) and the vector database Weaviate to deliver tailored workout recommendations. Through rigorous unit testing of each module and comprehensive integration testing across the system, we have ensured its reliability and functionality.

7.2: Future Work

Looking ahead, deploying the system would involve utilizing frameworks like Django for backend development and React for frontend interfaces, ensuring efficient data processing and seamless user interaction. Future efforts will focus on optimizing the speed and efficiency of video stitching processes to enhance user experience and responsiveness. Continued refinement and enhancement of these components will be crucial to maintaining the system's performance and meeting evolving user expectations in personalized fitness planning.

8. REFERENCES

1. Oliveira, T., Leite, D., & Marreiros, G. (2016). PersonalFit: Fitness App with intelligent plan generator. Presented at C3S2E '16, July 20-22, 2016, Porto, Portugal. DOI: 10.1145/2948992.2949014.

2. Maletha, V., Zafar, M. Z., Mohit, & Sachin (2023). AI Gym Trainer Using Artificial Intelligent. International Research Journal of Modernization in Engineering Technology and Science, 5(12)

3. Kannan, R. G., Mohan, M., Gokul, R., & Kumar, G. P. (2023). Enhancing Fitness Training with AI. International Journal of Research Publication and Reviews, 4(4), 5418-5423. DOI: 10.55248/gengpi.234.4.38631

4. Lamba, Anuj, Anand Kumar Nayak, Pranay Pimple, Vaibhav Patil, Pawan Bhaldare, and Ram Kumar Solanki. "AI-Based Fitness Trainer." International Journal of Innovative Research in Technology, vol. 10, no. 1, June 2023, pp. 614. IJIRT, ISSN: 2349-6002..

5. V. Singh, A. Patade, G. Pawar and D. Hadsul, "trAIner - An AI Fitness Coach Solution," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 2022, pp. 1-4, doi: 10.1109/I2CT54291.2022.9824511.

6. "Gym 2.0: How AI & Robotics are Redefining the Fitness Experience." Roars Inc., www.roarsinc.com.
7. Narayanan, Sridhar. "AI Can Coach You to Lose Weight. But a Human Touch Still Helps." Stanford Graduate School of Business, 9 July 2021, stanford.io/437JjGt.

8. "The AI Revolution: How to Use AI to Achieve Your Fitness Goals." Fitness Volt, fitnessvolt.com.
9. "The Top 4 Ways AI Has Impacted the Fitness Industry." ABC Fitness, abcfitness.com.
10. "Embrace Wellness with AI: Chatbots Boosting Exercise, Diet, and Sleep." Neuroscience News, neurosciencenews.com.

APPENDIX

APPENDIX A - CODE

Sample code:

1. Code 1 is Run on Google Colab for ease where we initiate the schema
2. Code 2 is run on VS where sample open API keys, aws keys, and weaviate secret keys are given.
3. Code 2 is run using streamlit run 'file_name.py'
4. Here "file_name.py" is "stremtrial.py"

Code1:

```
!pip install "weaviate-client==3.*"
import weaviate
client = weaviate.Client(
    url="https://capstone20bce2543-jxnrz4sa.weaviate.network",  # 
Replace with your Weaviate endpoint

auth_client_secret=weaviate.auth.AuthApiKey(api_key="LE9JY76cgiiSgNBC45
1tt7lZCXdCF2DaalNd"),  # Replace with your Weaviate instance API key
    additional_headers={
        'X-OpenAI-Api-key': 
"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"# Replace with 
your third party API key and identifying header
    },
    timeout_config=(5, 15)
)

class_obj = {
    "class": "Sciii",
    "vectorizer": "text2vec-openai",  # If set to "none" you must 
always provide vectors yourself. Could be any other "text2vec-*" also.
    "moduleConfig": {
        "text2vec-openai": {},
        "generative-openai": {}  # Ensure the `generative-openai` 
module is used for generative queries
    }
}

client.schema.create_class(class_obj)
import requests
import json
```

```
# Path to the JSON file
file_path = '/content/exercisesssf.json'

# Open the file and load the data
with open(file_path, 'r') as file:
    data = json.load(file)['exercises']  # Load data and directly
access the 'exercises' list

# Assuming 'client.batch' and 'batch.add_data_object' are part of a
defined class and context
client.batch.configure(batch_size=100)  # Configure batch
with client.batch as batch:  # Initialize a batch process
    for i, d in enumerate(data):  # Batch import data from the
'exercises' list
        print(f"importing question: {i+1}")
        properties = {
            "title": d["Title"],
            "description": d["Desc"],
            "type": d["Type"],
            "bodyPart": d["BodyPart"],
            "equipment": d["Equipment"],
            "level": d["Level"],
            "rest_time": d["rest_time"],
            "video_s3_url": d["video_s3_url"]
        }
        batch.add_data_object(
            data_object=properties,
            class_name="Sciii"
        )
```

Code 2:

```
import streamlit as st
import openai
import weaviate
import time
import boto3
from moviepy.editor import VideoFileClip, concatenate_videoclips
from requests.exceptions import ConnectionError
from langchain_community.llms import OpenAI
```

```
# AWS Configuration
aws_access_key_id = 'AKIAVRUVT65FCHIRF2H3'
aws_secret_access_key = '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'
bucket_name = 'mybucketcapstonemel'

# OpenAI API key configuration
openai_api_key = "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"
openai.api_key = openai_api_key

# Weaviate client setup
client = weaviate.Client(
    url="https://capstone20bce2543-jxnrz4sa.weaviate.network",

auth_client_secret=weaviate.auth.AuthApiKey(api_key="LE9JY76cgiiSgNBC45
1tt7lZCXdCF2DaalNd"),
    additional_headers={'X-OpenAI-Api-key': openai_api_key},
    timeout_config=(5, 15)
)

# Initialize the Langchain OpenAI LLM
llm = OpenAI(api_key=openai_api_key)

# Define the Streamlit app
def streamlit_app():
    st.title('Exercise Video Generator')

    # Input fields
    age = st.number_input('Age', min_value=0, max_value=100, value=25,
step=1)
    gender = st.selectbox('Gender', ('male', 'female', 'other'))
    level = st.selectbox('Fitness Level', ('beginner', 'intermediate',
'advanced'))
    focus_points = st.multiselect('Focus Points', ['chest', 'arms',
'legs', 'back', 'core'])
    favorite_exercises = st.multiselect('Favorite Exercises',
['walking', 'yoga', 'swimming', 'cycling', 'weightlifting', 'squats'])
    least_favorite_exercises = st.multiselect('Least Favorite
Exercises', ['running', 'squats', 'weight lifting', 'planks'])
    weaknesses = st.multiselect('Physical Weaknesses', ['knee pain',
'back pain', 'none'])

    if st.button('Generate Videos'):
        user_profile = {
```

```
            "age": age,
            "gender": gender,
            "level": level,
            "focus_points": focus_points,
            "favorite_exercises": favorite_exercises,
            "least_favorite_exercises": least_favorite_exercises,
            "weakness": weaknesses
        }

        concepts = generate_concepts(user_profile)
        st.write("Generated Concepts:\n", concepts)

        # Query the Weaviate client and print the video URLs
        video_urls = safe_query(client, concepts)
        st.write("Video URLs:", video_urls)

        # Download and stitch videos if available
        if video_urls:
            download_and_stitch_videos(video_urls)
        else:
            st.write("No video URLs available to download and stitch.")

def generate_concepts(user_profile):
    query_prompt = f"""
    Generate a list of exercise-related search concepts for a
{user_profile['age']}-year-old {user_profile['gender']} who is at an
{user_profile['level']} level of fitness. They focus on {',
'.join(user_profile['focus_points'])}, enjoy {',
'.join(user_profile['favorite_exercises'])}, dislike {',
'.join(user_profile['least_favorite_exercises'])}, and need exercises
that are suitable for {', '.join(user_profile['weakness'])}.
    """
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": query_prompt}],
        max_tokens=100,
        temperature=0.7
    )
    return response['choices'][0]['message']['content'].strip()

def safe_query(client, concepts, max_retries=5):
    video_urls = []
    retries = 0
```

```
    while retries < max_retries:
        try:
            response = client.query.get("Sciii", ["title",
"description", "type", "bodyPart", "equipment", "level", "rest_time",
"video_s3_url"]).with_near_text({"concepts":
concepts}).with_limit(5).do()
            for item in response['data']['Get']['Sciii']:
                video_urls.append(item['video_s3_url'])
            return video_urls
        except ConnectionError:
            retries += 1
            time.sleep(2**retries)
            print(f"Retry {retries}/{max_retries}")
    return video_urls

def download_and_stitch_videos(urls):
    clips = []
    s3 = boto3.client('s3', aws_access_key_id=aws_access_key_id,
aws_secret_access_key=aws_secret_access_key)
    for url in urls:
        file_name = url.split("/")[-1]
        try:
            s3.download_file(bucket_name, file_name, file_name)
            clip = VideoFileClip(file_name)
            clips.append(clip)
            print(f"Downloaded and loaded {file_name}")
        except Exception as e:
            print(f"Failed to download or load video {file_name}:
{str(e)}")

    if clips:
        # Use 'compose' method to handle clips with different sizes or
aspect ratios
        final_clip = concatenate_videoclips(clips, method='compose')
        final_clip.write_videofile("final_output.mp4")
        print("All videos have been stitched together into
'final_output.mp4'")
    else:
        print("No videos were downloaded or stitched.")

if __name__ == "__main__":
    streamlit_app()
```

APPENDIX B- SCREENSHOT

Result: UI Web Page

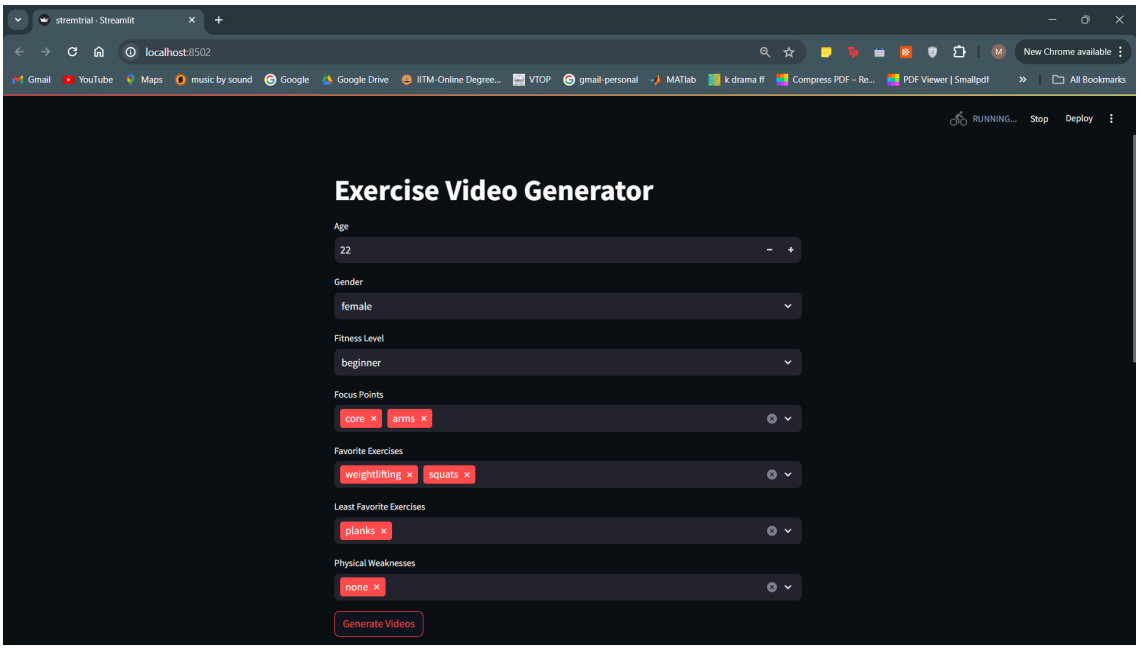

Fig 7.1: UI - Web page

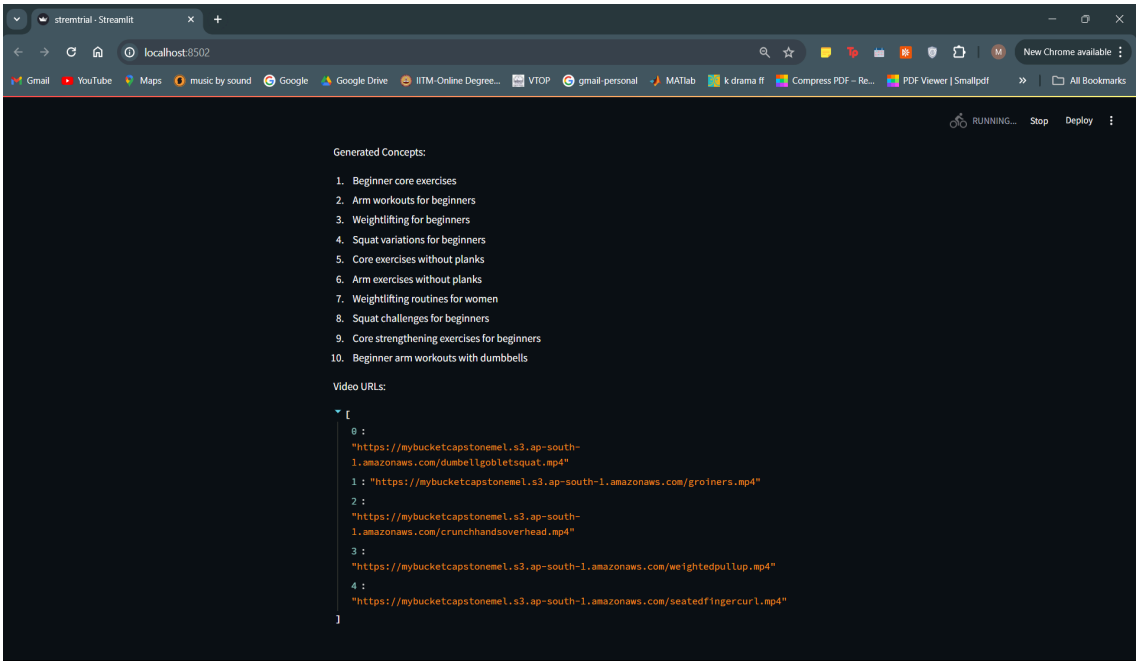

Fig 7.2: UI - Web page

Result: Stitched Video

Fig 8: Final output stitched video