AI BASED FITNESS BOT

A DESIGN PROJECT REPORT

Submitted by

NAME

INIGHO BENJAMIN L

LOGA PRASATH S S

VASIL HASSAN R

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE, 2025

AI BASED FITNESS BOT

A DESIGN PROJECT REPORT

Submitted by

INIGHO BENJAMIN L (811722001016)

LOGA PRASATH S S (811722001028)

VASIL HASSAN R (811722001054)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE, 2025

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this design project report titled **"AI BASED FITNESS BOT"** is the bonafide work of **INIGHO BENJAMIN L (8117220016), LOGA PRASATH S S (811722001028), VASIL HASSAN R (811722001054),** who carried out the design project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other design project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. T. Avudaiappan, M.E., Ph.D.,
HEAD OF THE DEPARTMENT

Department of Artificial Intelligence
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

SIGNATURE

Mrs. Sumathi, M.E.,
SUPERVISOR

ASSISTANT PROFESSOR
Department of Artificial Intelligence
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

Submitted for the viva-voce examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the design project report on **"AI BASED FITNESS BOT"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This design project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF TECHNOLOGY**.

Signature

_____

INIGHO BENJAMIN L

_____

LOGA PRASATH S S

_____

VASIL HASSAN R

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this design project.

We are glad to credit honourable chairman **Dr. K.RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our design project and offering adequate duration in completing our design project.

We would like to thank **Dr. N. VASUDEVAN, M.E., Ph.D.,** Principal, who gave opportunity to frame the design project the full satisfaction.

We whole heartily thank **Dr. T.AVUDAIAPPAN**, **M.E., Ph.D.,** Head of the department, **ARTIFICIAL INTELLIGENCE** for providing his encourage pursuing this design project.

We express our deep and sincere gratitude to our design project guide **Mrs. SUMATHI**, **M.E.**, Department of **ARTIFICIAL INTELLIGENCE,** for her incalculable suggestions, creativity, assistance and patience which motivated us to carry out this design project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

This project presents the design and development of an AI-Based Fitness Chat Bot that functions as a virtual personal fitness assistant. The system is capable of providing personalized workout and diet recommendations through a conversational interface. With the rising demand for intelligent and accessible health solutions, this chatbot aims to simplify the user's fitness journey by offering instant, AI-driven guidance tailored to individual preferences.The bot collects essential user inputs such as age, weight, region, dietary preference, and fitness level. Using this information, it generates structured and adaptive fitness plans. The intelligence behind the system is powered by the Mistral-7B Instruct model, which enables human-like understanding and response generation. The backend is developed using Python and Flask, while the frontend is a responsive web interface that facilitates seamless interaction between the user and the AI.Unlike static fitness applications, this system adapts to real-time user queries and updates recommendations based on ongoing feedback, simulating the experience of interacting with a knowledgeable and friendly human coach. The project demonstrates the application of large language models in real-world domains such as fitness and wellness, showcasing the potential of AI to support personalized healthcare and improve lifestyle outcomes through accessible technology.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
BMR	Basal Metabolic Rate
BMI	Body Mass Index
GPT	Generative Pre-Trained Model
JSON	JavaScript Object Notation
LLM	Large Language Model
NLP	Natural Language Processing
UX	User Experience

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The integration of artificial intelligence (AI) into health and wellness applications has significantly transformed how individuals' access and engage with fitness and nutritional support. Traditional fitness applications often rely on static rule-based systems or generic content, which limits their ability to adapt to diverse user needs. These systems may fail to offer the level of personalization and interactivity expected in modern user-centric platforms.

This project proposes the development of T-800 Fitness Coach, an AI-powered virtual assistant designed to offer real-time, personalized fitness and dietary guidance through natural language conversation. The system utilizes the Mistral-7B-Instruct-v0.2 model, a large language model (LLM) specifically fine-tuned for instruction-following tasks. The model is capable of generating coherent, contextually appropriate, and well-formatted responses based on user inputs.

The frontend of the application is implemented using HTML, CSS, and JavaScript, offering an intuitive and responsive user interface. The backend is built using the Flask web framework in Python, which handles user data processing and interaction with the language model. The system collects essential user details—such as age, weight, fitness goal, dietary preference, fitness level, and region—and incorporates these inputs into a system prompt. This prompt directs the Mistral model to produce responses tailored to the specific needs and preferences of each user.

By combining a structured user interface with intelligent, model-driven dialogue generation, T-800 Fitness Coach aims to simulate a human-like coaching experience that is both informative and motivational.

.

1.2 OBJECTIVE

The primary objective of this project is to develop a web-based virtual assistant that provides users with customized fitness and nutrition support through natural language interaction. At the core of this system lies the Mistral-7B-Instruct-v0.2 model, a powerful instruction-tuned language model capable of interpreting user queries and generating relevant, structured, and personalized responses. The application is designed to collect user-specific data—such as age, weight, dietary preference, fitness goal, fitness level, and region—and incorporate these inputs into a carefully engineered prompt. This allows the language model to tailor its recommendations to the unique needs of each user, offering context-sensitive advice on workout routines, meal plans, and general health practices.

Beyond basic functionality, the project also aims to evaluate the real-world utility of instruction-based large language models in delivering expert-like guidance within niche domains such as fitness and wellness. A key focus is placed on creating a smooth and engaging user experience through a clean, responsive frontend interface and a robust backend architecture. The system is designed to simulate a conversational, motivational coach, not only answering questions but also encouraging users to stay consistent with their health goals. Furthermore, the project serves as a demonstration of how general-purpose models like Mistral-7B-Instruct can be effectively adapted for specialized tasks without domain-specific training, thereby showcasing the growing potential of language models in personalized AI solutions.

CHAPTER 2

LITERATURE SURVEY

2.1 ARTIFICIAL INTELLIGENCE IN HEALTHCARE

A. Bohr, K. Memarzadeh

This book presents an in-depth exploration of real-world applications of artificial intelligence within the healthcare industry. It covers essential domains such as patient monitoring, clinical diagnosis, and the development of personalized treatment plans using data-driven insights. The authors also discuss the integration of machine learning and natural language processing in building intelligent medical systems, which directly aligns with the use of AI in personalized fitness coaching.

Merits

- Offers comprehensive coverage of AI applications in clinical settings.
- Highlights the importance of user data in tailoring recommendations.
- Provides insights into real-time monitoring and decision support systems.
- Reinforces the relevance of NLP in healthcare advisory tools.

Demerits

- Primarily focused on traditional healthcare systems rather than fitness applications.
- Less technical depth on NLP model training or deployment strategies.
- Limited discussion of transformer-based architectures.

2.2 TRANSFORMERS FOR NATURAL LANGUAGE PROCESSING

Rothman, D

Rothman's work focuses on the transformer architecture that revolutionized modern natural language processing. It introduces the principles of self-attention, encoder-decoder mechanisms, and multi-head attention, all critical to models like Mistral-7B. The book provides practical guides to building transformer-based applications such as chatbots and question-answering systems—core components of the T-800 Fitness Coach.

Merits

- Deep explanation of transformer models with practical examples.
- Directly applicable to chatbot and virtual assistant development.
- Supports fine-tuning, customization, and conversational flow modeling.
- Bridges the gap between theoretical concepts and coding practices.

Demerits

- Less focus on integrating NLP models with personalized data input.
- Does not extensively cover deployment or performance tuning in low-resource settings.
- Health and fitness use cases are not explicitly discussed.

2.3 HEALTH INFORMATICS: PRACTICAL GUIDE FOR HEALTHCARE AND IT PROFESSIONALS

Hoyt, R. E., & Yoshihashi, A. K.

This guidebook provides a comprehensive overview of the fusion between healthcare delivery and digital information systems. It emphasizes user-centered design, electronic health records, and system interoperability—concepts that are mirrored in the T-800 system's collection and use of user input to generate personalized fitness responses.

Merits

- Promotes user-centric design for healthcare applications.
- Highlights data-driven approaches in developing smart health tools.
- Offers guidance on building systems that align with individual user profiles.
- Encourages ethical and secure handling of user data.

Demerits

- Focuses more on institutional healthcare environments than personal fitness.
- Lacks detailed discussion on machine learning or AI model selection.
- Not tailored to real-time chatbot or NLP-based interaction design.

2.4 LANGUAGE MODELS ARE FEW-SHOT LEARNERS

Brown, T. B., et al.

This seminal paper introduces GPT-3 and the few-shot learning paradigm, showing how large pretrained language models can perform a variety of tasks with minimal training data. This foundation supports the use of models like Mistral-7B in applications such as the T-800, where task-specific fine-tuning is avoided in favour of well-structured prompting.

Merits

- Validates the capability of large language models for general-purpose use.
- Demonstrates practical success in generating human-like responses.
- Justifies the use of Mistral-7B-Instruct with structured system prompts.
- Eliminates the need for fine-tuning, speeding up development.

Demerits

- High computational cost if retraining or scaling is needed.
- Theoretical focus with limited discussion on domain-specific personalization.
- Less emphasis on end-user experience or interface integration.

2.5 LLAMA: OPEN AND EFFICIENT FOUNDATION LANGUAGE MODELS

Touvron, H., et al.

This paper introduces the LLaMA family of open-source language models, optimized for instruction following with significantly reduced computational requirements. It presents a strong case for the use of efficient, open-weight models in domain-specific applications like the T-800 Fitness Coach, particularly when accessibility and lightweight deployment are priorities.

Merits

- Advocates for open-access, efficient models like Mistral.
- Encourages instruction-tuning for better user interaction.
- Aligns with current trends in accessible, smaller LLM deployment.
- Supports the use of LLMs in educational, healthcare, and personal assistant domains.

Demerits

- Heavy on model architecture, light on application integration.
- Evaluation mostly benchmark-driven, not focused on real-world UX.
- Does not provide deployment strategies for web-based systems.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The field of digital fitness assistance is currently populated by a variety of tools and applications ranging from mobile fitness trackers, web-based platforms, static video-based coaching, to human personal trainers. These solutions, while popular, largely follow a one-size-fits-all approach that lacks true personalization. Users typically input basic details and receive generic plans that do not adapt over time or respond to nuanced health queries.

Most applications offer a list of preloaded exercise routines and meal suggestions. Some allow goal selection (e.g., weight loss or muscle gain), but these systems do not adapt based on feedback or conversation. The systems are driven by conditional logic rather than actual understanding and lack any meaningful interaction loop with the user.

Furthermore, existing chatbot implementations (if present) are often rule-based, responding with fixed templates rather than understanding user intent or generating customized responses. These bots lack memory, reasoning, or the ability to handle follow-up questions. Additionally, human trainers—while more capable—are often costly, limited by location, and unable to provide 24/7 support.

Finally, data collected in traditional systems (such as weight, age, or fitness level) is typically underutilized. These systems are rarely equipped to integrate AI techniques such as machine learning or natural language processing to deliver truly intelligent insights or guidance.

3.1.1 Demerits

Lack of Contextual Personalization

The existing systems often fail to tailor fitness and nutrition plans based on comprehensive user inputs such as age, region, dietary habits, fitness level, and lifestyle.

Static and Scripted Interactions

Most applications rely on rule-based or pre-scripted responses, which are not capable of understanding complex queries or offering dynamic, human-like interactions.

Limited Scalability and Learning

Current platforms do not learn from user behavior over time and cannot adapt to changing user goals or preferences, leading to stagnation in engagement.

Low User Motivation and Retention

The absence of interactive feedback and motivational dialogue reduces user enthusiasm, often resulting in discontinued usage after initial interest.

Restricted Accessibility to Expert-Level Guidance

Human coaching services are either expensive or geographically limited, while automated systems cannot match the intelligence or empathy of expert trainers.

No Integration of Advanced NLP Technologies

The majority of existing systems do not incorporate modern AI models such as transformers, making them incapable of nuanced conversation or real-time language understanding.

3.2 PROPOSED SYSTEM

The proposed system, T-800 Fitness Coach, introduces a new paradigm in virtual fitness guidance by leveraging transformer-based artificial intelligence to provide real-time, intelligent, and context-aware responses. The system is built on a combination of technologies, including a Flask-based backend, a responsive HTML/CSS frontend, and a large language model (LLM)—specifically the Mistral-7B-Instruct—through the Together API.

The system begins by collecting user input across several dimensions including age, weight, region, dietary preference, fitness level, and goal. This information is stored temporarily to construct a contextual prompt, which is passed to the language model to generate personalized recommendations. The user then engages with the assistant through a chat-based interface that mimics a human conversation. The assistant can handle questions, generate personalized workout or meal plans, and offer motivational dialogue—something traditional fitness apps cannot do.

The solution is designed to be lightweight and browser-accessible, ensuring wide compatibility across devices. Additionally, the use of instruction-tuned LLMs allows the assistant to adapt to follow-up questions, maintain conversational flow, and offer structured, logically formatted answers using markdown-like formatting (e.g., bullet points, headings).

As a modern, AI-driven fitness platform, T-800 distinguishes itself by delivering an adaptive, evolving user experience that closely replicates the dynamic interaction of a real-life fitness coach. Unlike conventional systems that rely on static content or rule-based responses, T-800 engages users through intelligent, conversational dialogue powered by a transformer-based language model. This allows it to not only understand diverse user inputs but also respond with personalized, context-aware guidance that adjusts in real-time as the user’s goals, fitness levels, or queries evolve.

3.2.1 Merits

Enhanced Personalization

The assistant generates responses that are tailored to the user's demographic and physical profile, such as age, weight, dietary preference, regional context, and fitness goals.

Conversational Intelligence via NLP

The use of a transformer-based large language model allows for sophisticated understanding of user queries and generation of human-like, grammatically rich, and logically structured responses.

Interactive and Motivational User Experience

The system simulates a personal coach by maintaining a motivational and friendly tone, thereby improving user engagement and consistency in following health routines.

Web-Based Accessibility

The application is platform-independent and requires no installation, making it easily accessible from any device with an internet connection.

Future-Ready Architecture

Designed with modularity in mind, the system can be extended to support additional features such as voice interaction, progress tracking, wearable integration, or multilingual support.

Open Model Integration for Cost-Effective AI Use

By leveraging openly available models like Mistral-7B, the system reduces dependency on commercial APIs, ensuring both cost efficiency and deployment flexibility.

CHAPTER 4

SYSTEM SPECIFICATIONS

4.1 HARDWARE SPECIFICATIONS

Computer/Server (Local or Cloud-Based)

A system with a minimum of 8GB RAM is recommended to support backend processing and interface rendering. GPU support is optional.

User Device:

The frontend is browser-based and responsive, allowing access from any modern device, including desktops, laptops, tablets, and smartphones.

Internet Connectivity:

A stable internet connection is essential to send and receive requests from the Together AI API, which powers the AI-based chat functionality.

Power Supply:

Standard AC power is sufficient to operate the host device, whether for development or live deployment.

4.2 SOFTWARE SPECIFICATIONS

Operating System: Windows, macOS, or Linux.
Backend: Python with Flask; Requests for API communication.
Frontend: HTML, CSS, JavaScript for responsive chat UI.
AI Integration: Together AI API with Mistral-7B-Instruct model.
NLP: Generates personalized, context-aware responses.
Logic: Converts user input into prompts; formats AI replies.
Session: Maintains chat history for contextual flow.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECHTURE

The system design of the T-800 Fitness Coach application is based on a modular client-server architecture that enables efficient interaction between the user and the AI-powered virtual assistant. The design adheres to the principles of separation of concerns, maintainability, and scalability, making it adaptable for future upgrades or integrations.

The presentation layer forms the frontend of the application and is responsible for facilitating user interaction. This layer is implemented using standard web technologies, including HTML5, CSS3, and JavaScript. Upon launching the application, users are greeted with a personalized data collection form. This form captures essential details such as the user's age, weight, fitness level, fitness goals, geographical region, and dietary preferences. Once this information is submitted, the interface transitions into a dynamic chat window. This chat interface allows users to communicate with the AI assistant using natural language inputs. To enhance the user experience, the frontend includes features such as real-time message rendering, markdown-style response formatting (e.g., bold text and bullet points), and a typing indicator that simulates AI activity.

The application logic layer is implemented using the Flask micro web framework in Python. This layer serves multiple roles: it hosts the static files of the frontend, exposes RESTful endpoints for communication, and manages the overall logic flow of user interactions. When a user sends a message, the complete conversation history, including the user's initial profile and prior exchanges, is formatted into a JSON payload. This data is then submitted to the backend via a POST request to the /chat endpoint.

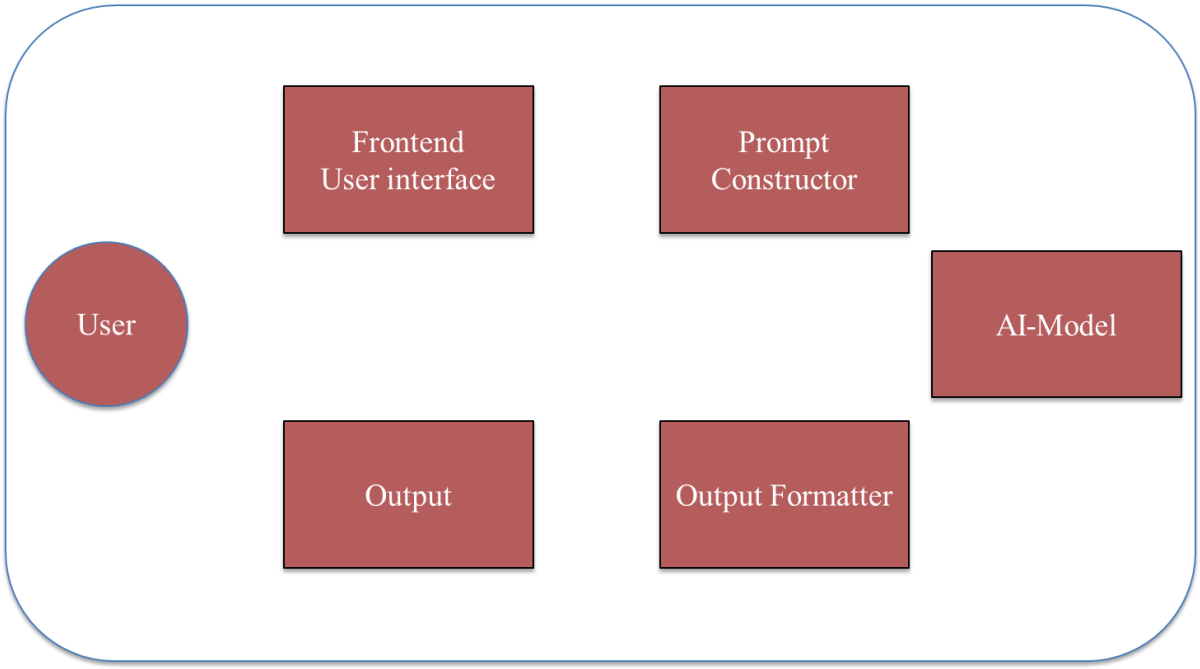

Fig. 5.1 System Architecture

The typical workflow begins when a user accesses the application and enters their fitness-related information. This information is incorporated into a system prompt that defines the assistant's role and the scope of conversation. As the user submits queries through the chat interface, each new message is appended to the shared message history and sent to the Flask backend. The server reformats this history and communicates with the Together.ai API. Once the AI model processes the conversation, it returns a text-based response. This response is then relayed back to the frontend, where it is dynamically rendered using custom logic that converts markdown-like syntax into visually structured HTML elements.

The overall architecture of the T-800 Fitness Coach supports a clear delineation of responsibilities. The frontend focuses solely on user experience and visual rendering, the backend manages the flow of data and server-side logic, and the intelligence layer provides the core natural language understanding and response generation. Utilizing an external API for AI processing decouples the model execution from server infrastructure, thereby simplifying updates, scaling, and maintenance.

Furthermore, by maintaining a persistent message history and embedding user context into every request, the system ensures continuity in conversation and delivers responses that are context-aware and user-specific. This layered, modular design allows the T-800 Fitness Coach to operate as a seamless, intelligent, and personalized virtual fitness assistant, blending the strengths of web development and modern AI technologies.

CHAPTER 6

MODULES DESCRIPTION

6.1 USER INTERFACE MODULE

The User Interface Module is the primary access point for users interacting with the T-800 Fitness Coach system. It presents a visually appealing, responsive web interface where users input their fitness information and engage in natural language conversation with the AI assistant. Developed using standard frontend technologies, this module ensures accessibility, usability, and dynamic interaction on both desktop and mobile devices

Components

1. HTML/CSS Interface

- Structures the layout of the web application, including the data input form and chat window.

2. JavaScript Logic

- Handles user events such as form submission, message sending, and response rendering.

3. Responsive Design

- CSS media queries and flexible layouts ensure compatibility across devices and screen sizes.

4. Validation Logic

- Verifies that the uploaded image is not empty, corrupted, or of unsupported format.

5. Chat Interface Elements

- Includes message bubbles, sender tags ("You", "T-800"), typing indicator animation, and input field.

Working Principle

When the user accesses the web application, they are first presented with a fitness profiling form. Once the form is filled and submitted, the UI switches to the chat interface using JavaScript-based state control. The user's inputs are not only stored in memory but also embedded in the first system message to guide the AI.

The chat interface captures each user query, visually appends it to the message container, and sends it to the backend for processing. Once the AI responds, the UI renders the assistant's message using HTML converted from markdown-style formatting. The typing indicator is shown during the waiting period and hidden upon receiving the response.

1. **User Input**

   When a user opens the app, they are presented with a form to enter their age, weight, goal, fitness level, region, and diet preference.

2. **Interface Transition**

   Upon submission, the system hides the form and reveals a chat interface.

3. **Chat Interaction**

   Users can type queries into the input box. JavaScript appends messages to the chat window and triggers backend communication .

4. **Dynamic Updates**

   The chat interface displays bot responses in real time, formatted with markdown-style enhancements like bold text and bullet points.

Function

- **User Engagement**: Encourages users to input accurate fitness data through a clean and user-friendly form interface.
- **Responsiveness**: Adapts to various device screens to offer consistent usability, including mobile phones, tablets, and desktops.
- **Real-Time Interactivity**: Facilitates seamless two-way communication between the user and the AI assistant.

- **Visual Feedback**: Enhances experience with animations, feedback cues, and smooth transitions.
- **Frontend State Control**: Manages the visibility of sections (e.g., hiding the form after submission and showing the chat window).

6.2 USER DATA AND CONTEXT

The User Data and Context Module is a critical intermediary that links the user's personal information to the AI system's reasoning capabilities. Its primary goal is to personalize the interaction by embedding user data into the conversation's foundational system prompt. The module captures user input fields—age, weight, fitness goal, fitness level, region, and diet preference—and organizes them into a structured JSON object.

This module also plays a central role in managing the **conversation context**. Every message exchanged is preserved in a session-level message history array, which is used to maintain continuity and relevance in responses. By contextualizing each conversation turn, the AI assistant can refer to previous user messages and adjust its replies, accordingly, simulating a natural and informed conversation.

Components

1. **User Profile JSON Object**
   - Stores structured data like age, weight, fitness level, region, and diet preferences.
2. **Prompt Constructor**
   - Dynamically generates the initial system prompt by embedding user data into a predefined template.
3. **Message History Manager:**
   - Maintains an array of messages (user and assistant) for persistent context during multi-turn conversations.
4. **Session Context Handler**
   - Stores and updates session-specific information, ensuring each chat is consistent and personalized.

Working Principle

- Once the user completes the form, all inputs are captured and stored in a JSON object. This data is used to form a custom system message that sets the tone, style, and expertise level for the AI.
- Every user query and AI response is appended to a shared conversation history array. This array is sent to the AI on each request, ensuring the assistant has access to past exchanges and can respond coherently.

Function

- **Personalization Engine**: Injects user-specific variables into the AI system prompt to generate tailored responses.
- **Contextual Memory**: Maintains an ongoing log of messages to support multi-turn, coherent conversations.
- **Consistency Across Sessions**: Ensures that the AI doesn't reset or forget user goals during a single session.
- **Input Normalization**: Converts raw user inputs into machine-readable and prompt-compatible formats.
- **Scalability**: The context module can be extended in the future to include more parameters (e.g., fitness history, health metrics).

6.3 BACKEND CONTROLLER MODULE

The Backend Controller Module is the central processing hub that manages server-side logic and orchestrates the communication between the frontend interface and the AI model. Developed using the Flask micro web framework, it exposes RESTful APIs to handle chat interactions and serves static assets required for the frontend display.

This module processes incoming HTTP POST requests containing the user's message history, system prompt, and other metadata. It validates the JSON structure, logs errors when needed, and relays the request to the AI Integration Module. Upon receiving a response from the AI, it parses the assistant's message and packages it into

a JSON format suitable for frontend rendering.

Components

- **Flask Web Server**:

  Listens for incoming HTTP requests and routes them to appropriate handlers.
- **Static File Routing**:

  Serves frontend assets like HTML, CSS, JS, and image files from the static/ directory.
- **/chat Endpoint**:

  Accepts POST requests from the frontend and handles the message history payload.
- **Request & Response Parser**:

  Validates incoming messages, processes them, and extracts the response from the AI before returning it to the UI.
- **Error Handling Mechanism**:

  Captures exceptions (e.g., timeout, API failure) and returns user-friendly error messages.

Working Principle of the Indicator Light Module

After receiving a response from the backend, the raw text is passed to the formatter function. It processes markdown cues and replaces them with HTML equivalents. The formatted string is injected into the chat interface inside a newly created message block.

Simultaneously, the typing indicator is toggled off, and the chat scrolls to the latest message. If an error is encountered, a warning message is displayed, and the issue is logged in the console.

Function

- **API Gateway**: Acts as a centralized endpoint for all chat communication between the frontend and the AI service.
- **Request Routing**: Directs incoming user data to appropriate processing functions.
- **Security & Validation**: Validates payloads and protects against malformed or malicious input.
- **Error Resilience**: Ensures graceful failure by catching exceptions and returning human-readable fallback messages.
- **Infrastructure Backbone**: Supports server deployment both locally and on cloud platforms like Render, Heroku, or AWS.

.

6.4 AI INTERGRATION MODULE

The AI Integration Module is responsible for connecting the system to the Mistral-7B-Instruct model hosted on the Together.ai platform. This module encapsulates all interactions with the AI API, including prompt construction, request submission, response handling, and error recovery.

It constructs a payload that includes the complete message history, system prompt, model name, temperature settings, and token limits. Once the response is received, it extracts the relevant content and passes it to the backend for relay to the frontend. This module effectively decouples the AI reasoning layer from the rest of the system, enabling future updates (e.g., switching models or providers) without requiring frontend or backend architectural changes.

Components

- **Markdown-Like Formatter**:

  Converts AI's markdown-styled text into HTML tags (`*text*` → `<b>text</b>`, `- item` → `<li>item</li>`).

- **Dynamic DOM Updater**:

  Injects new response blocks into the chat window, maintaining flow and scroll behaviour.
- **Typing Animation Controller**:

  Displays and hides the typing indicator during API processing.
- **Error Display Handler**:

  Shows fallback text (e.g., “Sorry, something went wrong”) in case of connection or response issues.
- **List and Paragraph Parser**:

  Detects bullet points, numbered lists, and structured paragraphs for readable formatting.

Working Principle

After receiving a response from the backend, the raw text is passed to the formatter function. It processes markdown cues and replaces them with HTML equivalents. The formatted string is injected into the chat interface inside a newly created message block.

Simultaneously, the typing indicator is toggled off, and the chat scrolls to the latest message. If an error is encountered, a warning message is displayed, and the issue is logged in the console.

Function

- **AI Abstraction Layer**

  Simplifies interaction with third-party AI models via standard API calls.
- **Dynamic Prompting**

  Builds prompts in real-time using structured user data and conversation history.
- **Configurability**

  Allows easy tuning of parameters such as temperature (creativity), model selection, and token limits.

- **Scalability**

  Facilitates replacement or extension of the AI model in the future without affecting other system components.

- **Reliability**

  Handles rate limits, timeout conditions, and fallback scenarios through structured exception handling.

6.5 RESPONSE HANDLING AND RENDERING MODULE

The Response Handling and Rendering Module is a client-side utility responsible for converting raw text responses from the AI into visually appealing, well-formatted HTML content. It interprets markdown-like syntax in AI messages, such as bullet points, bold text, and headings, and renders them accordingly.

In addition, this module manages the visibility of the typing animation and ensures that the chat interface scrolls to display the most recent message. It also handles error messages gracefully, displaying fallback responses if the backend or AI service encounters an issue. This module plays a vital role in delivering clarity, readability, and real-time feedback, thereby enriching the user's experience.

Components

- **Markdown-Like Formatter**

  Converts AI's markdown-styled text into HTML tags (*text* → text, - item → item).

- **Dynamic DOM Updater**

  Injects new response blocks into the chat window, maintaining flow and scroll behavior.

- **Typing Animation Controller**

  Displays and hides the typing indicator during API processing.

- **Error Display Handler**

  Shows fallback text (e.g., “Sorry, something went wrong”) in case of connection or response issues.

- **List and Paragraph Parser**

  Detects bullet points, numbered lists, and structured paragraphs for readable formatting.

Working Principle

After receiving a response from the backend, the raw text is passed to the formatter function. It processes markdown cues and replaces them with HTML equivalents. The formatted string is injected into the chat interface inside a newly created message block.

Simultaneously, the typing indicator is toggled off, and the chat scrolls to the latest message. If an error is encountered, a warning message is displayed, and the issue is logged in the console.

Function

- **AI Abstraction Layer**: Simplifies interaction with third-party AI models via standard API calls.
- **Dynamic Prompting**: Builds prompts in real-time using structured user data and conversation history.
- **Configurability**: Allows easy tuning of parameters such as temperature (creativity), model selection, and token limits.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The T-800 Fitness Coach project successfully demonstrates the integration of modern web technologies with advanced natural language processing to deliver a personalized virtual fitness assistant. Built on a modular, client-server architecture, the system provides a seamless user experience from data input to AI-powered conversation. By leveraging HTML5, CSS3, and JavaScript for the frontend, the application ensures a responsive and intuitive interface. The Flask-based backend effectively manages API requests, data flow, and interaction with the Together.ai platform, where the Mistral-7B-Instruct model processes natural language queries and generates human-like responses. Throughout the development, particular focus was placed on personalization and contextual awareness. By embedding user-specific information into the AI's prompt, the system offers tailored diet advice, fitness guidance, and motivational interaction, creating a sense of intelligent companionship for users on their fitness journey.The modular approach in system design enabled logical separation of responsibilities, simplifying both implementation and future scalability. The AI model's integration allows for flexible language understanding and output generation without the need for local GPU resources, making it deployable on lightweight infrastructure. In conclusion, the T-800 Fitness Coach achieves its objective of delivering an interactive, context-aware, and user-friendly fitness coaching experience using a combination of modern web development and AI capabilities.

7.2 FUTURE ENHANCEMENT

While the current implementation of the T-800 Fitness Coach fulfills its primary goal of providing personalized, AI-driven fitness assistance, there remains significant potential for enhancement and expansion in future iterations of the system.One of the most impactful improvements would be the addition of user authentication and profile

management. By allowing users to register and log in, the system could store individual fitness profiles, preferences, and historical data, thereby enabling continuity and deeper personalization. Alongside this, integrating a backend database such as Firebase, SQLite, or PostgreSQL would allow persistent storage of user conversations, progress tracking data, and long-term fitness plans.

Another promising enhancement lies in enabling real-time streaming of AI responses. Currently, the AI replies only after the entire message is generated. Future versions could use token-based streaming to deliver responses word-by-word, simulating a more natural and interactive conversation flow. Similarly, packaging the application as a mobile app using frameworks like React Native or Flutter could significantly boost accessibility, allowing users to engage with the assistant on the go.The system could also evolve through voice command integration, allowing users to interact via speech rather than typing. This would enhance accessibility for a wider audience, particularly for users with visual impairments or limited mobility. In addition, integrating the application with third-party fitness APIs such as Google Fit or Fitbit would allow the assistant to offer more dynamic, data-driven recommendations based on the user's real-world physical activity, heart rate, or sleep patterns.

Expanding the application's reach by implementing multi-language support would further improve usability, especially within a linguistically diverse country like India. This would allow users to interact in their native language, making the experience more inclusive and relatable

APPENDIX A
SOURCE CODE

Backend

```
from flask import Flask, request, jsonify, send_from_directory
import requests
import os

app = Flask(__name__, static_folder='static')

# Configuration
TOGETHER_API_KEY = "api_key"  # 🔐 Replace with your actual key
MODEL_NAME = "mistralai/Mistral-7B-Instruct-v0.2"

# Serve the frontend
@app.route('/')
def serve_frontend():
    return send_from_directory('.', 'index.html')

# Chat API endpoint
@app.route('/chat', methods=['POST'])
def chat():
    try:
        data = request.json
        messages = data.get('messages', [])

        payload = {
            "model": MODEL_NAME,
            "messages": messages,
```

```
            "temperature": 0.7,
            "max_tokens": 800
        }

        headers = {
            "Authorization": f"Bearer {TOGETHER_API_KEY}",
            "Content-Type": "application/json"
        }

        response = requests.post(
            "https://api.together.xyz/v1/chat/completions",
            json=payload,
            headers=headers
        )
        response.raise_for_status()

        ai_message = response.json()["choices"][0]["message"]["content"]
        return jsonify({"response": ai_message})

    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

Frontend

```
<body>
    <header>
        <div class="container">
```

```
        <h1>T-800 Fitness Coach</h1>
        <p class="tagline">Your AI-powered fitness and nutrition assistant</p>
      </div>
    </header>

    <div class="container">
      <div id="user-info-section" class="user-info-form">
        <h2>Let's Get Started!</h2>
            <p>Please provide some information to personalize your fitness journey:</p>
        <form id="user-info-form">
          <div class="form-group">
            <label for="age">Age</label>
            <input type="number" id="age" required min="10" max="100">
          </div>
          <div class="form-group">
            <label for="weight">Weight (kg)</label>
             <input type="number" id="weight" required min="30" max="200" step="0.1">
          </div>
          <div class="form-group">
            <label for="goal">Fitness Goal</label>
            <select id="goal" required>
              <option value="">Select your goal</option>
              <option value="muscle gain">Muscle Gain</option>
              <option value="weight loss">Weight Loss</option>
              <option value="maintenance">Maintenance</option>
              <option value="endurance">Endurance</option>
              <option value="strength">Strength</option>
```

```
            </select>
        </div>
        <div class="form-group">
            <label for="fitness-level">Fitness Level</label>
            <select id="fitness-level" required>
                <option value="">Select your level</option>
                <option value="beginner">Beginner</option>
                <option value="intermediate">Intermediate</option>
                <option value="advanced">Advanced</option>
            </select>
        </div>
        <div class="form-group">
            <label for="region">Region</label>
              <input type="text" id="region" required placeholder="e.g., South India, North India">
        </div>
        <div class="form-group">
            <label for="diet">Diet Preference</label>
            <select id="diet" required>
                <option value="">Select your diet</option>
                <option value="non-veg">Non-vegetarian</option>
                <option value="veg">Vegetarian</option>
                <option value="vegan">Vegan</option>
            </select>
        </div>
        <button type="submit" id="submit-info">Start Chat</button>
    </form>
</div>
```

```
    <div id="chat-section" class="chat-container" style="display: none;">
        <div class="chat-header">
            <img src="static\t-800.jpg" alt="T-800">
            <span>T-800 Fitness Coach</span>
        </div>
        <div class="chat-messages" id="chat-messages">
            <!-- Messages will appear here -->
        </div>
        <div class="typing-indicator" id="typing-indicator">
            <div class="typing-dots">
                <div class="typing-dot"></div>
                <div class="typing-dot"></div>
                <div class="typing-dot"></div>
            </div>
        </div>
        <div class="input-area">
             <input type="text" id="user-input" placeholder="Type your message
here...">
            <button id="send-btn">Send</button>
        </div>
    </div>
  </div>
```

APPENDIX B

SCREENSHOTS

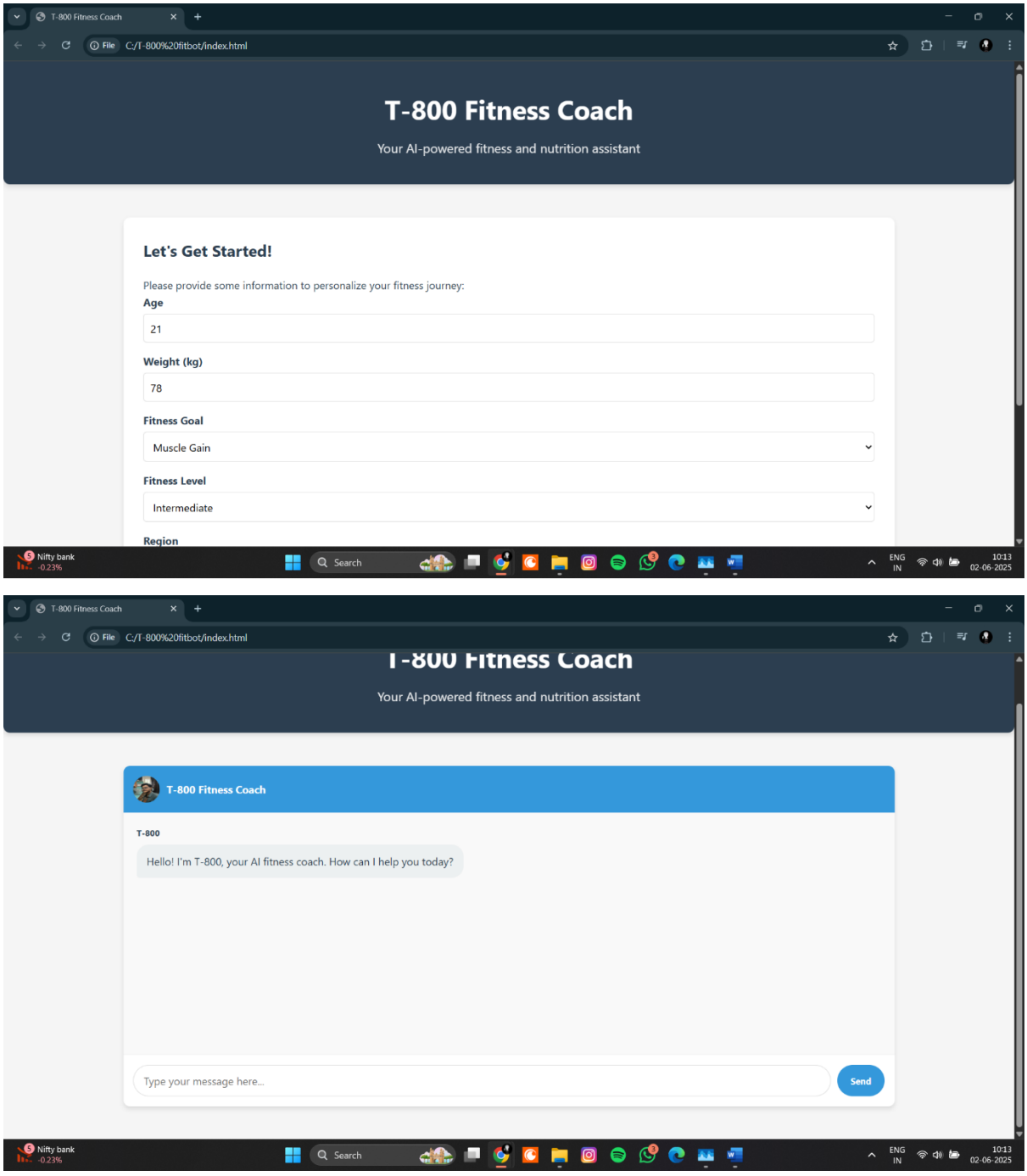

Fig. B.1 User Interface

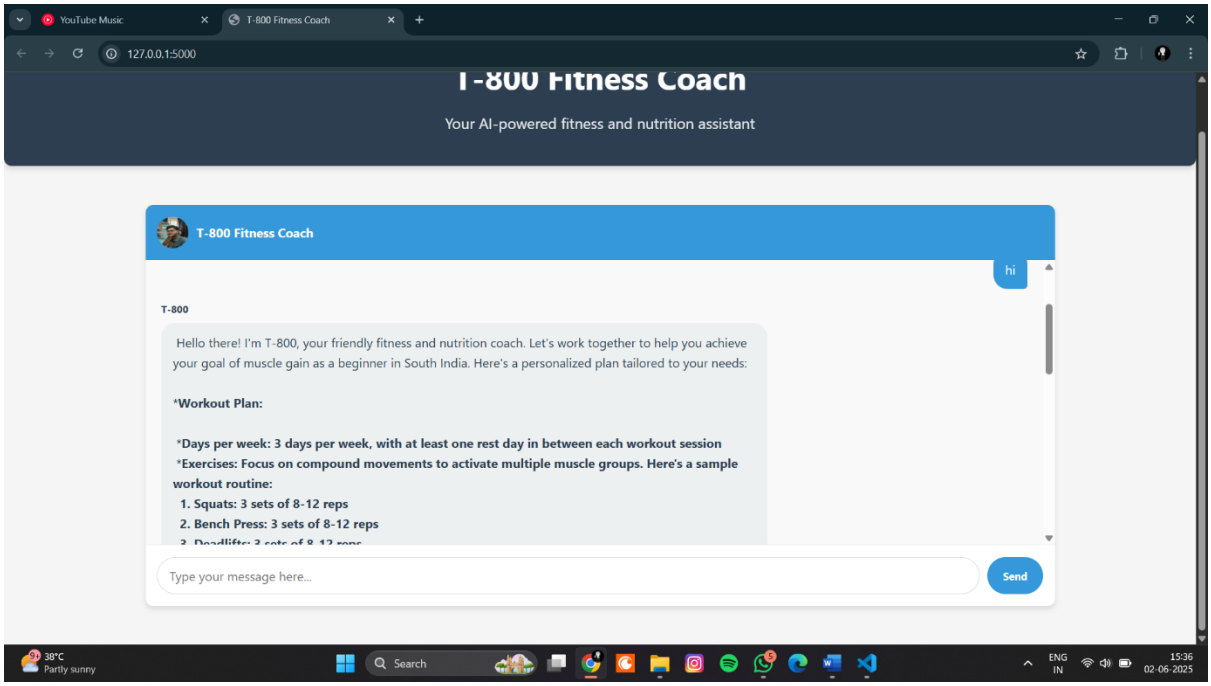

Fig B.2 AI Reply

REFERENCES

1. Flask, “Flask Documentation,” Pallets Projects [Online]. Available: https://flask.palletsprojects.com/,2025.
2. Together AI, “Together Inference API Documentation,” Together.ai [Online]. Available: https://docs.together.ai/docs/inference , 2024
3. Hugging Face, “Mistral-7B-Instruct v0.2,” Hugging Face [Online]. Available: https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2 , 2023.
4. R. Rao, J. D. Baker, and N. Moore, “ChatGPT and AI chatbots in medicine: A new frontier for digital health,” npj Digital Medicine, vol. 6, no. 1, pp. 1–3, [Online]. Available: https://www.nature.com/articles/s41746-023-00763-4, 2023.
5. H. B. McMillan, P. Brown, and N. Chater, “An artificial intelligence–based conversational agent for personalized human nutrition advice,” Nature Machine Intelligence, vol. 4, pp. 132–139, [Online].Available: https://www.nature.com/articles/s42256-022-00467-w,2022.
6. S. Howard and K. Yang, "Artificial Intelligence for Health and Fitness: An Emerging Frontier," *IEEE Access*, vol. 9, pp. 98760–98775, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9506923
7. J. F. Allen, "Natural Language Understanding," *Communications of the ACM*, vol. 45, no. 4, pp. 54–60, 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/504267
8. M. H. Al-Shamri, “Intelligent Fitness Training Assistant using AI Techniques,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 10, no. 6, pp. 144–150, 2019. [Online]. Available: https://thesai.org/Downloads/Volume10No6/Paper_20-Intelligent_Fitness_Training_Assistant.pdf.