A PROJECT REPORT

On

FITNESS TRACKING APPLICATION

Submitted in partial fulfillment of the requirements to

JWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA

MASTER OF COMPUTER APPLICATIONS

Submitted By

VASIREDDY LIKHITHA

(22JK1F0033)

Under the Esteemed Guidance of

Mrs. V. MOUNIKA, MCA

Assistant Professor

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

GUNTUR ENGINEERING COLLEGE, YANAMADALA

GUNTUR – 522019, ANDHRA PRADESH, INDIA

(November- 2022 to May -2024)

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
GUNTUR ENGINEERING COLLEGE, YANAMADALA
GUNTUR – 522019, ANDHRA PRADESH, INDIA
(November- 2022 to May -2024)

CERTIFICATE

This is to certify that this project report entitled “**FITNESS TRACKING APPLICATION USING PYTHON**” is a bonafide record of work carried out by VASIREDDY LIKHITHA (22JK1F0033) under the guidance and supervision of **Mrs. V. MOUNIKA** in partial fulfillment of the requirements for the award of the degree of Master Of Computer Applications in GUNTUR ENGINEERING COLLAGE (GEC) FOR during the year 2022- 2024.

Internal Guide **Head of the Department**

External Guide

ACKNOWLEDGEMENT

I would like to express our profound gratitude towards "Mrs. V. MOUNIKA", Assistant Professor Dept. Of MCA, who played a supervisory role to utmost perfection, enabled us to seek through our MCA main project and for guiding as an internal guide methodically and meticulously.

I am highly indebted to MR. ESWAR RAO, M.Tech, Assistant Professor,Dept of MCA, Head of the Department, Computer Science and Engineering for providing us all the necessary support.

I am very much thankful to the college management for their continuous support and facilities provided.

I render our deep sense of gratitude to Dr. M. RAMA KOTAIAH, Principal, for permitting us to carry out our main project works. We would like to express our sincere thanks to Computer Science and Engineering staff for lending us their time to help us complete the work successfully.

I would also like to thank our staff, parents, and friends for their enduring encouragement and assistance whenever required.

Vasireddy Likhitha
(22JK1F0033)

DECLARATION

I hereby inform that this main project entitled **“FITNESS TRACKING APPLICATION USING PYTHON”** has been Carried out and submitted in partial fulfillment for the award to the degree of **Master of Computer Applications to Jawaharlal Nehru Technological University Kakinada** under the guidance of **Mrs. V. MOUNIKA MCA., Assistant Professor Dept. of MCA,** The work embodied in this project work is original and has not been submitted in part or full for any degree of this or any degree of any other university.

Vasireddy Likhitha
(22JK1F0033)

INDEX

LIST OF FIGURES

INTRODUCTION

INTRODUCTION

The Python-based fitness monitoring program is an all-inclusive resource for keeping tabs on and bettering one's health and level of physical activity. This application uses the Python programming language to track a variety of fitness measures, including steps taken, calories burnt, distance walked, and workout time. It has an easy-to-use interface and comprehensive capabilities.

This fitness monitoring program can easily connect with sensors, wearable devices, and other sources of data thanks to Python's speed and flexibility, allowing for real-time updates and individualized insights. Users may stay inspired and dedicated to a healthy lifestyle by setting fitness goals, creating training schedules, and tracking their progress over time. This guide will go over the Python fitness tracking app's architecture, main features, data movement, and implementation details. By delving into the application's design concepts and functioning, developers may enhance their grasp of how to tailor and expand its features to suit individual user requirements.

Fitness monitoring solutions that are both efficient and effective are in great demand due to the growing importance of health and fitness in today's fast-paced society. People are keen to track their workouts, evaluate their development, and reach their fitness objectives precisely. We present a novel fitness tracking app built in Python, a robust and flexible programming language, in reaction to this increasing need. To give customers a complete platform for tracking and controlling their fitness journey, our fitness tracking app uses Python's capabilities. Our software provides a smooth user experience with powerful capabilities by taking use of Python's ease of use, scalability, and large library support.

The demand for efficient tools to track and oversee exercise routines is greater than ever before in this age of growing health awareness. Step right up to the cutting edge of fitness technology with our state-of-the-art Fitness Tracking Application, built with the utmost care and precision using the robust and versatile Python programming language. Our tracking fitness app has the potential to change the game when it comes to people's dedication to their fitness and health objectives. With Python's help, we were able to build a flexible and adaptable platform that meets the demands of users all around the world.

Technology has evolved into an essential resource in the modern day for anyone pursuing improved physical fitness. Step right up to our state-of-the-art Fitness Tracker Application, a game-changer that will set you on the path to peak health. Python provides a solid basis for our program, which provides a user-friendly platform for monitoring, analyzing, and improving your fitness activities.

Maintaining a healthy lifestyle should be a top priority as we face the challenges of contemporary living. With pinpoint accuracy and maximum efficiency, our fitness tracker app leads users through each stage of their path to better health. We have developed a flexible and adaptable application that meets the varied demands and goals of people all around the globe by utilizing Python's capabilities.

Fig 1 : Fitness tracking application tracker

What is Fitness tracking application?

An individual's level of physical activity, exercise regimen, and general health and wellness may be better managed with the use of a fitness monitoring program. Smartphones, tablets, and a variety of wearable fitness trackers are common platforms for these kinds of apps.

User Interface (UI): When interacting with the Command Line Interface (CLI), the built-in input() method can be used for a basic text-based user interface (UI).

Use libraries like Tkinter, PyQt, or Kivy to create a more sophisticated GUI.

Data collection:

Involves users manually entering several pieces of information, such as their daily step count, exercise kind, duration, and intensity.

The ability to automatically track utilizing application programming interfaces (APIs) from fitness trackers or cellphones (this requires knowledge of the API of the platform or device you're targeting).

Data Archiving:

Simple files (such as text files, CSV files, or JSON files) or databases can hold data.

Analyzing Data:

Gain valuable insights like time series patterns, goal comparisons, etc., by utilizing Python's robust data analysis packages, such as Pandas for data processing and Matplotlib or Seaborn for visualization.

Requests for Information and Suggestions:

Set up logic to inform users of their progress; this may be in the form of encouraging messages when they reach milestones, prompts to keep moving, or recommendations on how to enhance their exercise program according to their statistics.

Connectivity to Third-Party APIs:

If you want more sophisticated features, you can connect with third-party APIs for things like weather (so you can get exercise time recommendations), maps (so you can keep tabs on your runs or bike rides), and social networking (so you can share your accomplishments).

How do you design a Fitness tracking application ?

The process of conceptualization organizing, creation, evaluation, deployment, and maintenance are the several stages that comprise the design of a fitness monitoring program. An all-inclusive manual for developing a fitness monitoring app is available here:

1. Define Objectives and Scope

Identify Goals: Find out which tasks you would like the app to do, whether it's encouraging exercise, keeping tabs on health indicators, or keeping track of your food intake.

Target Audience: Make sure the design and functionalities are tailored to the consumers by understanding their needs, preferences, and demographics.

Scope of Features: Pick the most important features. A few examples of what may fall under this category include health monitoring, social integration, dietary tracking, activity tracking, and tailored insights.

2. Research and User Feedback

Market Analysis: Research the industry by looking at comparable applications and your rivals to see what features users like and where you may be able to fill a need.

User Surveys and Interviews: In order to learn what features are most important to users and where they are experiencing problems with existing solutions, it is recommended to conduct user surveys and interviews.

Technology Feasibility: Check the viability of the technologies like GPS, motion detectors, health trackers, and third-party APIs that will be required to implement the intended functionalities.

3. Design User Experience (UX)

User Personas and Journey Maps: To better understand how users will engage with the app, it is helpful to develop comprehensive user profiles and plot out their journeys.

Wireframes: Create wireframes that outline the structure and flow of the app's various screens and interactions.

Prototype: Create a working model of the UI and user interactions by creating a clickable prototype.

4. User Interface (UI) Design

Style Guide: Design a style guide that incorporates the app's branding into the app's visual aspects, such as hues, typography, and icons.

Creating a High-Fidelity Design: Take the schematics and turn them into a complete, interactive design.

Accessibility: Everyone, especially those with mobility issues, should be able to easily use the design.

5. Technical Architecture and Development

Consider Your Options for Development: Build your app with your intended users and their needs in mind before deciding among native, hybrids, or web app development.

How is the Tech Stack Defined? Choose reliable technology for the application's front and back ends, databases, and any third-party services.

API Integration: Create and maintain application programming interfaces (APIs) to manage data transfer between the app, any external services, and servers.

Methodology: Use an agile development approach that embraces feedback and iterative testing at every stage.

6. Testing and Quality Assurance

Functional Testing: To make sure the software performs as intended, it must undergo functional testing.

Usability Testing: Find out what users think by testing the product with actual people.

Performance Testing: Make that the app works properly in a variety of settings, such as on multiple devices and in different network configurations, by doing performance testing.

Security Testing: Implement and test safety measures that safeguard user information as well as guarantee compliance with the appropriate regulations and laws.

7. Deployment and Launch

To find any last-minute problems and get early feedback, launch the app to only a few people as a beta.

Marketing, advertising, and reaching out to prospective consumers are all part of a well-thought-out launch plan.

Be sure to follow each app store's criteria before submitting your app to them.

8. Post-Launch Monitoring and Updates

User Input: Keep an eye on app performance data and user input to find places to make improvements.

Constantly improve the app by incorporating user input and staying up-to-date with technical advancements.

Keep the app up-to-date with security patches, compliance checks, and updates for new systems and gadgets as part of routine maintenance.

A well-designed, intuitive, and functional fitness monitoring software that caters to its customers' individual demands in this area may be yours by according to these guidelines.

SYSTEM ANALYSIS

2.1 Existing System

Registering users, monitoring their activities, establishing goals, measuring their progress, and maybe even social sharing capabilities are all likely components of the existing system. All aspects of the system, including its architecture, UI, data storage techniques, security measures, and interactions with third parties, should be documented. Additionally, it has to explain how the program will handle errors, user processes, and component functionality, as well as any planned upgrades. Software developers, testing personnel, and clients should all find useful instructions for the program in the documentation.

Types of Existing Fitness Tracking Systems

Smartphone Apps:

Applications such as MyFitnessPal, Google Fit, and Apple Health

Particulars: These applications frequently make use of the sensors integrated into the smartphone to monitor the user's activity levels and the number of steps they take. In addition, they provide the option to let users manually input data such as exercise, food consumption, and water usage.

Methodology: Tracks motion with the use of the phone's sensors and global positioning system.

Wearable Devices:

Brands like Apple Watch, Fitbit, and Garmin

Characteristics: These gadgets keep tabs on your heart rate, the quality of your sleep, and even your stress levels in addition to your physical activity.

Modern technology includes programs and sensors that track and analyze vital signs in real time.

Specialized Fitness Apps:

Some examples include Peloton, Strava, and Nike Training Club.

Highlights: Designed with a particular kind of workout or activity in mind. Many of these applications have built-in social features that allow users to join challenges, communicate with friends, and share their accomplishments.

Technology: It offers comprehensive sport-specific data and frequently connects with other devices (such as bicycling monitors or cardiac monitors).

Web-based Platforms:

Using Fitbit Dashboard and Training Peaks as examples

Benefits: These systems usually provide you a better look at all the data that applications and gadgets have acquired. They could provide in-depth training programs, individual coaching, and sophisticated analytics.

The technology behind these systems allows users to examine their health and fitness in its entirety by integrating data from many sources and making them available through web browsers.

Common Features in Fitness Tracking Applications

Activity Monitoring: Steps, mileage, kilojoules expended, moving minutes.

Tracking vitals: respiration rate, sleep duration, hydration status, and food consumption.

A user's step count, weight reduction, or activity level may be customized using the goal-setting feature.

Connectivity: A lot of applications may connect to other services and devices, such as smart scales, nutrition monitors, and online fitness groups.

Involvement of Users: Tools to keep users engaged, such as badges, challenges, and alerts. Graphs and statistics used in data visualization let people see how they've progressed over time.

Technical Factors to Think About

Data Security and Privacy: In the healthcare industry, privacy regulations and laws like HIPAA and General Data Protection Regulation (GDPR) demand compliance while handling sensitive health information.

These systems need to be dependable enough to give feedback and monitoring in real-time and scalable enough to manage massive amounts of data.

The goal of the user interface as well as the design is to create a dynamic and easy-to-navigate platform that people of all fitness levels and experience levels can enjoy using to keep tabs on their health.

There has been a tremendous amount of development in fitness monitoring gadgets and apps. A quick rundown of some of the earlier fitness monitoring devices:

Pedometers: These basic gadgets count the steps a person takes and were one of the first types of fitness trackers. Mechanical ones came initially, and then digital ones. The main idea was to track users' steps and distance traveled in an effort to promote physical activity.

Heart Rate Monitors: Heart rate monitors gained popularity as a means for athletes to streamline their training regimens by keeping tabs on their heart rates as they worked out. A wristwatch-style receiver and a chest strap were the components of the first versions. More recent models typically connect to other fitness trackers and mobile phones.

Body Bugg: One of the earliest wearable body monitors to assess calorie expenditure using sensors that detected numerous bodily factors was the Body Bugg, which was released in the mid-2000s. The usage of the program "The Biggest Loser" helped bring it to a wider audience.

Nike+ Running Sensor: Launched in 2006 as a joint venture between Nike and Apple, the Nike+ Running Sensors was an in-shoe device that synced with an iPod. The device was useful for monitoring the speed and distance covered by runners.

Fitbit Tracker: The 2007-founded fitness monitoring company Fitbit introduced a wearable gadget that did more than just count steps; it also monitored sleep and, later on, heart rate. With the addition of GPS tracking, push alerts to your smartphone, and the ability to connect with social networks for fitness challenges, Fitbit has become a very feature-rich fitness tracker.

Garmin Sports Watches: For many years, Garmin has served as a leading provider of fitness trackers, with models designed for jogging, swimming, cycling, and other activities. In addition to other high-tech functions, these gadgets usually have GPS and the ability to track your heart rate.

EndomOndo: One of the first GPS-based fitness monitoring applications, EndomOndo allowed users to record their steps, runs, and other activities with only their smartphone when it was released in 2007. Users may also connect with others, challenge them other, and share their workouts using the app's social capabilities. Users were encouraged to transition to Under Armour's MapMyRun platform in 2021, when Endom Ondo was terminated.

Fig 2 : Endomondo app

MyFitnessPal: MyFitnessPal emphasizes the nutritional component of fitness monitoring and was launched in 2005. An extensive database of food nutritional information is provided, which helps users monitor their calorie consumption and activity. It offers a comprehensive perspective on health and fitness by integrating with other fitness applications and gadgets.

Launched in 2009, Strava caters mostly to bikers and runners. Most notably, it has a segment function that lets users compare their performance on different sections of road or trail. Strava is now a powerful athletic social network where users can post and track their exercises, take part in difficulties, and even organize their own virtual races.

Fitocracy: Launched in 2011, this software gamified fitness by making users' exercises into a game with point systems, levels, and missions that were directly tied to their physical activity. It used gamification and social media to inspire its users, but it lost steam as time went on, changed its emphasis, and then died out.

MapMyFitness: Founded in 2005, MapMyFitness (which encompasses MapMyRun, MapMyRide, etc.) provides the ability to monitor more than 600 different kinds of exercises. It offers comprehensive exercise logs and GPS-based route monitoring. It was bought by Under Armour in 2013, which led to its integration with their wider range of applications, including MyFitnessPal.

These gadgets laid the groundwork for today's smartwatches and fitness trackers, which use a combination of software and hardware to provide users a complete picture of their health and fitness levels. A wider trend towards complete health monitoring beyond basic fitness measures is shown in the additional features included in many modern devices. These features include stress tracking, ECG (electrocardiogram) tracking, and even arterial blood oxygen concentrations.

2.2 Problems of the existing system

Fitness tracker apps of yesteryear have helped a great deal with individual health monitoring and inspiration, but they have also been the target of many complaints and setbacks. The following are examples of typical issues encountered with these applications:

Initial fitness devices and their accompanying applications had accuracy issues, especially when it came to counting steps, calculating calories burnt, and evaluating the quality of sleep. The accuracy of the data captured was also hampered by GPS errors, which hindered the effectiveness of apps made for running or cycling.

Data Security and Privacy Concerns: Fitness trackers gather a lot of personal information, including where you are, your health, and even your biometric data. Concerns around data usage, sharing, and security are common among users, particularly in light of the fact that even large corporations have been hit by data breaches.

Battery Life: A lot of fitness trackers have a limited battery life, particularly the more feature-packed ones with GPS and pulse monitoring. Not everyone wanted to deal with the inconvenience of regular recharges, or wanted to track activities that lasted for lengthy periods of time.

The use of certain shoes equipped with sensors (such as the Nike+ system) or chest bands for heart rate tracking were commonplace in earlier applications. Because of this dependence, user comfort and accessibility may be compromised, making seamless activity tracking more of a hassle for consumers.

Put Too Much Stock in Quantitative Data: Many fitness applications place too much stock in numerical data, such as the number of steps a user takes or the number of calories burnt, which may not be indicative of their true health or fitness level. This excessive focus on metrics at the expense of consumers' well-being as a whole or their ability to recover from mental health issues is a real risk.

Some fitness applications have come under fire for having too complicated user interfaces that made them difficult to use, particularly for those who aren't tech-savvy. Apps with poor design may discourage consumers from using the tool frequently, which in turn reduces its usefulness.

While features like leaderboards, challenges, and badges were effective in getting people to use fitness apps at first, maintaining that motivation over the long haul was a major difficulty. Users' involvement tends to decline as the initial excitement wears off, particularly if they don't observe immediate benefits or find the app's rewards to be uninteresting.

Problems with Integration: Many of the first fitness applications were quite isolated and couldn't connect to other services or devices. Disjointed user experiences and fragmented health data might result from people utilizing numerous health and fitness platforms without integration.

As a result of these difficulties, the design of modern fitness tracker software has shifted to prioritize more precision, better protection of user data, longer battery life, and a more comprehensive and integrated method of monitoring health and fitness.

Disadvantages of Existing System

Issues with Accuracy: Data Accuracy was a common problem with early fitness monitoring applications. In instance, improper wear of the device might cause the step counter to under-or over-count the user's steps. Inaccurate health data was often produced because calorie burn estimates were based on broad algorithms that did not take into consideration individual differences in metabolism or the kinds of activities that people engaged in.

Many people were worried about the storage, usage, and possible sharing of the critical health and geographic information collected by these applications. Customers were apprehensive about privacy rules that let businesses to share their data with outside parties, especially after security breaches exposed their personal information.

A lot of fitness applications needed supplementary hardware, like external sensors or certain smartphone models, which may be a turnoff for people who didn't want to buy the extra gear. The app's functioning was also affected by device malfunctions or incompatibilities due to this reliance.

Lack of Personalization: Many fitness applications' early releases were missing in action when it came to personalization. They took into account each user's individual fitness level, health status, and objectives while making suggestions and setting targets. Disillusionment and unattainable objectives may result from this blanket strategy.

Constant usage of GPS and various other sensors by apps can rapidly deplete a phone's battery life. People who wish to use the application for a long time, like while they're riding their bikes or going for a run, found this annoying.

The focus was mostly on numerical statistics like the number of steps taken or calories burnt, rather than on more nuanced measures of health like flexibility, strength, or mental well-being. Because of this, we risk losing sight of the big picture when it comes to health.

Some fitness applications' user interfaces were overly sophisticated, making them hard to use, particularly for those without strong technical skills. This may discourage users from making full use of the software or lead to irritation and eventual disuse.

While gamification and social elements helped attract users in the beginning, it was difficult for applications to keep them interested over the long haul. If users don't see improvement right away or consistently, they may lose interest as the novelty wears off.

Many fitness applications have poor interoperability with other devices and apps, making it difficult for consumers to centralize their health data. Users may not be able to obtain a comprehensive picture of their fitness and health due to this lack of compatibility.

Newer versions of fitness monitoring apps have attempted to fix these issues by putting an emphasis on privacy protections, user customisation, greater accuracy, and an improved user experience.

PROPOSED SYSTEM :

When you're designing a new fitness monitoring software, it's important to take into account user expectations, current technological advancements, and the limitations of previous systems. With an emphasis on user involvement, technical innovation, and holistic health management, here is a suggestion for an updated fitness monitoring application system:

Improved Precision and the Integration of Sensors

Improve the precision of activity tracking by fusing data from many sensors, such as GPS, gyroscopes, and accelerometers and heart rate monitors.
Personalized activity detection and health insights may be achieved through the use of machine learning algorithms, which utilize sophisticated algorithms to interpret sensor data more precisely.

Efficient Monitoring of Health Status

For a more complete picture of your health, keep tabs on your rest, stress levels, and how quickly you bounce back from workouts. To give a more comprehensive picture of health, add sensors that can track skin temperature and blood oxygen levels.
Adding a nutritional component that allows users to record their meals and get information about their eating, such as calories counts, protein distribution, and nutritional deficits, would be a great addition to the app. One possible solution is to use image recognition software to analyze food images and streamline the logging process.

Data Security and Privacy

Provide consumers with complete agency over their data sharing selections and make it crystal clear how their data will be utilized via transparent privacy policies.
End-to-End Encryption: To safeguard user privacy and avoid data breaches, use strong encryption mechanisms for both data transit and storage.

Personalization of the User Interface

Customized Objectives and Suggestions: By analyzing user data, we can create fitness and health objectives that are specific to each person and take into account their current activity level, health profile, and previous achievements.

Make the application more accessible to people with disabilities and other impairments by designing an adaptable user interface that changes to fit the user's needs and streamlines interaction.

Engaging Users in a Sustainable Way

Incorporate elements like leaderboards, badges, and challenges to inspire users with a feeling of accomplishment and rivalry.

Community Features: Create a welcoming community by integrating social networking features that let users find and join groups, share accomplishments, and connect with friends.

Healthcare Service Integration

To facilitate remote monitoring and individualized healthcare services, telehealth connectivity enables patients to transmit their medical records directly to their doctors.

Applications for Other Parties: Provide application programming interfaces (APIs) so that the fitness tracker may be integrated with other health applications and gadgets. This will provide a more unified environment for health tracking.

Premium Functions

An AI-powered virtual coach can monitor your progress in real-time as you exercise, provide suggestions to improve your results, and keep you motivated.

Exercises using Augmented Reality: Make home exercises more interesting by adding digital information or beautiful settings onto the user's perspective.

Optimal Management of Batteries

Increased battery life for smartwatches is possible via energy-efficient design, which aims to optimize software algorithms and sensor utilization to decrease power consumption without sacrificing on functionality.

Actionable Strategy

Build some basic app prototypes and put them through rigorous user testing to see what people think so you can improve the design.

In order to mitigate development risks and incorporate user input, it is recommended to launch the app in stages, beginning with its essential functionality and then adding advanced features.

Make sure the app stays up-to-date and relevant by setting up systems for ongoing feedback and updates. This will help it adapt to user demands and new technologies.

By improving upon these crucial areas, the fitness monitoring app may provide users with a more trustworthy, interesting, and all-encompassing resource for managing their physical well-being, therefore raising the bar for what fitness applications can do.

LITERATURE SURVEY

LITERATURE SURVEY

The development, efficacy, technical progress, user engagement tactics, and possible problems with fitness monitoring apps would all be part of a literature review on the subject. To give you a sense of the breadth and depth of the current research on fitness monitoring apps, here is an overview:

The Developing State of Fitness Monitoring Devices: The historical context of this part covers the evolution of health monitoring technology, from early pedometers and cardiac trackers to modern, high-tech wearables and smartphone apps.

A look at recent technical advancements, including the usage of accelerometers, the advent of global positioning systems (GPS) monitoring, and the incorporation of machine learning to provide individualized health insights.

Analyze research that evaluate the effects of fitness monitoring apps on various physical health outcomes, including levels of physical activity, weight reduction, cardiovascular health, and general physical well-being.

Welfare of the Mind and Spirit: Examine studies that have examined the effects of activity tracking on the mind, including how it affects motivation, psychological involvement with health, and stress associated with continuous health monitoring, such as quantified-self stress.

User Engagement and Interaction

Research on the user's perspective on the design of the app's interface and overall experience has shed light on the factors that influence the convenience and uptake of fitness tracking software. Strategies for Engagement: Analyzing the efficacy of methods used to enhance user engagement and retention, such as gamification, social features, and customisation.

Privacy and the Veracity of Data

Data Accuracy: Read up on the effects of inaccurate fitness tracker data as well as the accuracy and dependability of such data.

Data storage, usage, and possible sharing are all aspects of data privacy that need discussion, as are ways to safeguard user privacy.

Connectivity to Healthcare Infrastructure

Healthcare Integration: Investigate the clinical applications of fitness monitoring data, including its use in telemedicine, chronic illness management, and preventative healthcare. Examine the current state of interoperability and the obstacles that have been overcome in order to make fitness tracking data compatible with many different healthcare systems and devices.

Studies of Markets and Demographics: Rates of Adoption: Look at the rates of adoption across various populations and the obstacles to adoption.

In this market analysis, we will take a look at the fitness tracking industry's leading companies, their products, and how they plan to compete.

Where to Go From Here: Trends in Technology: Forecasts on upcoming developments in fitness tracking technology, including the incorporation of augmented and virtual reality, and the creation of more sophisticated biometric sensors.

Discussion of the moral concerns around data sharing, constant monitoring, and the possible abuse of personal health information.

Ways of Conducting Research

Methods Revealed: The survey would also take a look at the techniques used to gather and analyze data in research that measure fitness, including randomized controlled trials, user feedback analysis, and longitudinal studies.

The purpose of this literature review is to offer a synopsis of the present and future of fitness monitoring apps, including a discussion of the advantages and disadvantages of the various approaches. It may lay the groundwork for future studies or developments that aim to make these apps even more useful and easy to use.

SYSTEM REQUIREMENTS

4.1 Hardware Requirements

Overview:
Our fitness monitoring software can't be launched or run without the devices listed in this document. Optimal performance, interoperability, and thorough data gathering across all devices and situations are the goals.

Purpose:
Developers, other stakeholders, and quality assurance teams can use this document as a reference and checklist to ensure the fitness monitoring application works properly and is compatible with the hardware that has been specified.

Scope: In order for the fitness monitoring app to work properly, it is required that certain mobile devices, smartwatches, and related peripherals be compatible with it.

Hardware Requirements

1. Mobile Devices

Any version of Android 8.0 (Oreo) or later is required.

All versions of iOS 12.

2. Minimum Hardware Specifications:

Computer Processor: 1.4 GHz quad-core
Required Memory: 2 GB
Storage: A minimum of 100 MB available
Monitoring Devices: Global Positioning System, Accelerometer, Gyroscope
Wireless networking: Bluetooth 4.0, 802.11 b/g/n

3. Recommended Hardware Specifications:

Central Processing Unit: Eight-core 2.0 GHz or newer

Memory: 4 GB Available storage space: 1 GB

Locator, acceleration, gyroscope, and heart rate monitor

Near Field Communication (NFC), Wi-Fi (802.11 ac), and Bluetooth 5.0

B. Wearable Devices

Compatibility:

Intended for use only with the aforementioned Android and iOS gadgets.

Bluetooth connection directly to smartphone app

Sensors:

Heart rate monitor

SpO2 sensor

Accelerometer

Gyroscope

Ambient light sensor

Runtime: At least twenty-four hours of continuous use

Battery life of up to seven days

Water Resistance:

Minimum IP67 rating for dust and water resistance

C. Peripheral Devices

Fitness Equipment:

Compatibility with smart gym equipment via Bluetooth or Wi-Fi

Capability to transmit workout data (e.g., treadmill speed, cycling cadence)

Additional Sensors:

External heart rate straps

Smart scales for weight and body composition analysis

4. Testing and Validation

Extensive testing should be carried out to guarantee that the data retrieved via sensor is accurate and dependable.
Wearables should be tested in a variety of environmental situations to ensure their operation and endurance. This includes testing for resistance to water and dust, among other features.

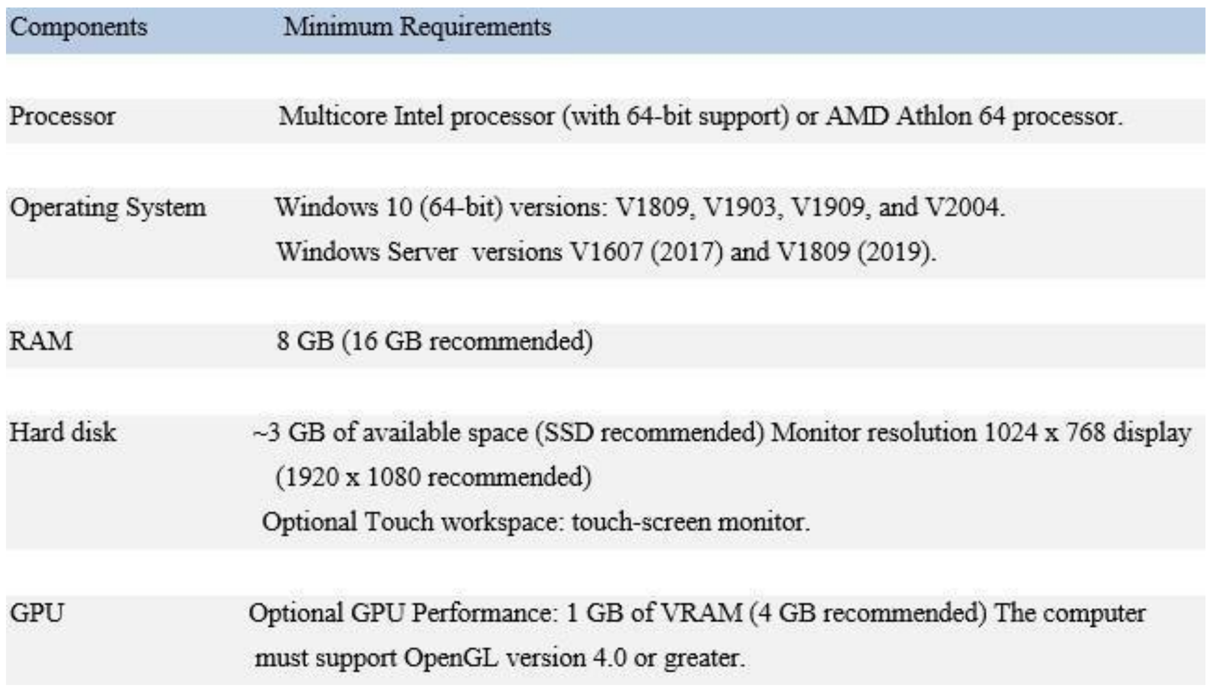
5. Conclusion

In order to build and test the fitness monitoring app, this paper is used as a reference. To make sure the app is consistent, dependable, and easy to use, make sure to follow these hardware requirements.

6. Approval

Document Prepared by: [Your Name/Team]
Reviewed by: [Reviewer's Name]
Approval Date: [Date]

Windows – Illustrator Minimum System Requirements



Components	Minimum Requirements
Processor	Multicore Intel processor (with 64-bit support) or AMD Athlon 64 processor.
Operating System	Windows 10 (64-bit) versions: V1809, V1903, V1909, and V2004. Windows Server versions V1607 (2017) and V1809 (2019).
RAM	8 GB (16 GB recommended)
Hard disk	~3 GB of available space (SSD recommended) Monitor resolution 1024 x 768 display (1920 x 1080 recommended) Optional Touch workspace: touch-screen monitor.
GPU	Optional GPU Performance: 1 GB of VRAM (4 GB recommended) The computer must support OpenGL version 4.0 or greater.

4.2 Software Requirements

Software specifications for a fitness monitoring app with the ability to measure, evaluate, and improve users' fitness levels via personalized and interactive features are outlined in this paper. To make sure the software works well, is reliable, and easy to use, it lists all the criteria, both functional and non-functional.

Operating System: The user's OS, such Microsoft Windows, needs to be compatible with the app that tracks software for fitness.

Core Development

Python

Virtual Environment

Data Collection

APIs

Smartphone Sensors

DATABASE MANAGEMENT SYSTEM

MYSQL

User Interface (UI)

Flask

Django

Tkinter

Visualization

Plotly

Matplotlib.

SYSTEM DESIGN

5.1 Use case Diagram

A user's interactions with a fitness tracking app are depicted in the diagram as an actor.

The many use scenarios illustrate the system's capabilities, including app launch and shutdown, workout creation and viewing, goal configuration, and statistics viewing.

You may further dissect each use case into scenarios that illustrate the user's interactions with the system in order to complete a certain activity. As a foundation for the system's design and implementation, these scenarios can aid in clarifying the system's needs and functionality.

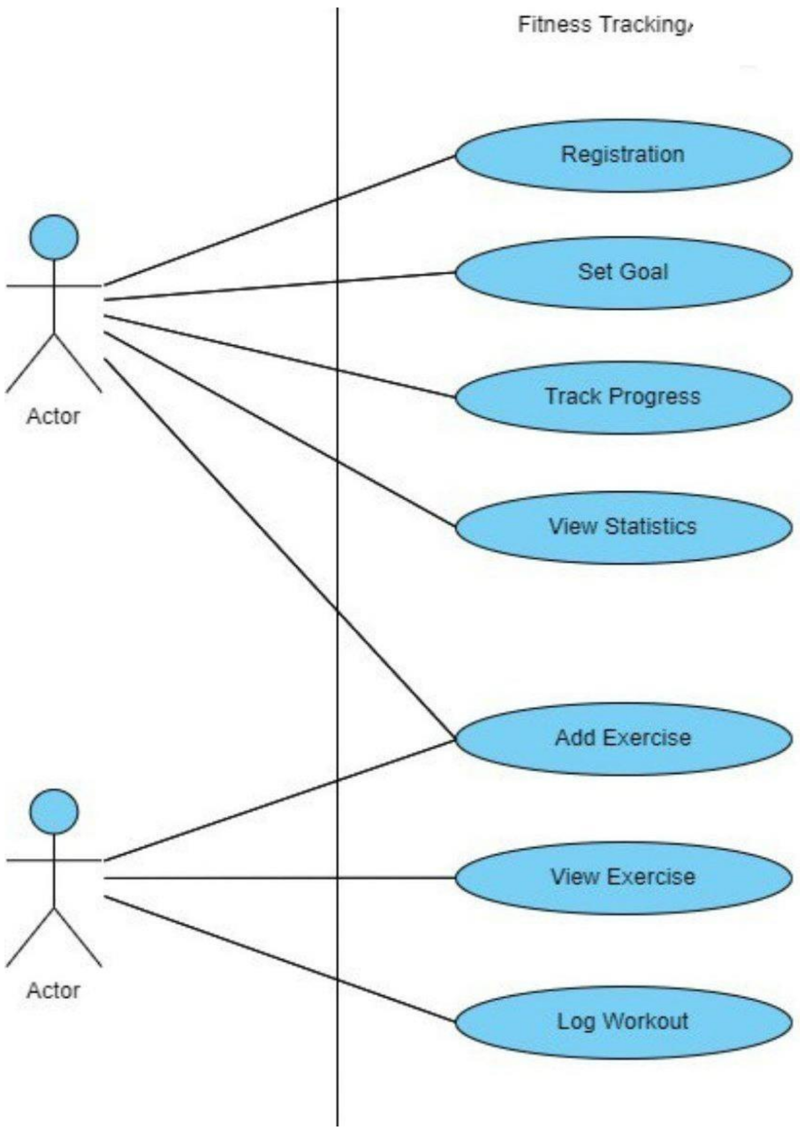


Fig 3: Use case Diagram

5.2 Class Diagram

In the illustration, the Fitness tracker class stands for the system's primary object or entity, which contains all of the methods and attributes that make the system work. Exercise, Workout activity details, User information, and User statistics are some of the properties that reflect the many sections or elements of the system. These properties offer the system its capabilities and services.

A few examples of the many things the system is capable of doing include opening and shutting the program, adding user data, examining statistics, creating workouts, and viewing them. A few parameters are passed into a method as inputs so that the method may carry out its work.

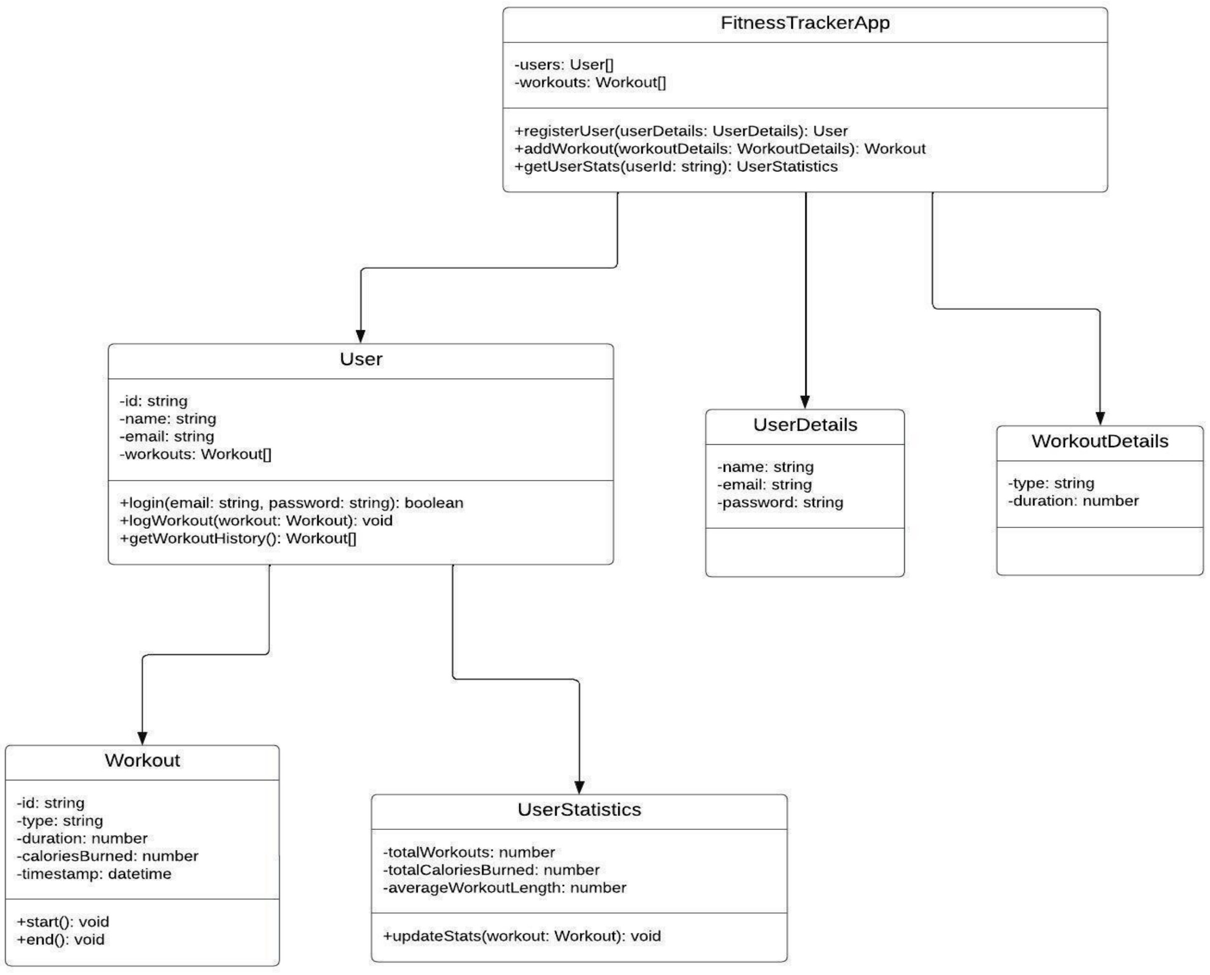
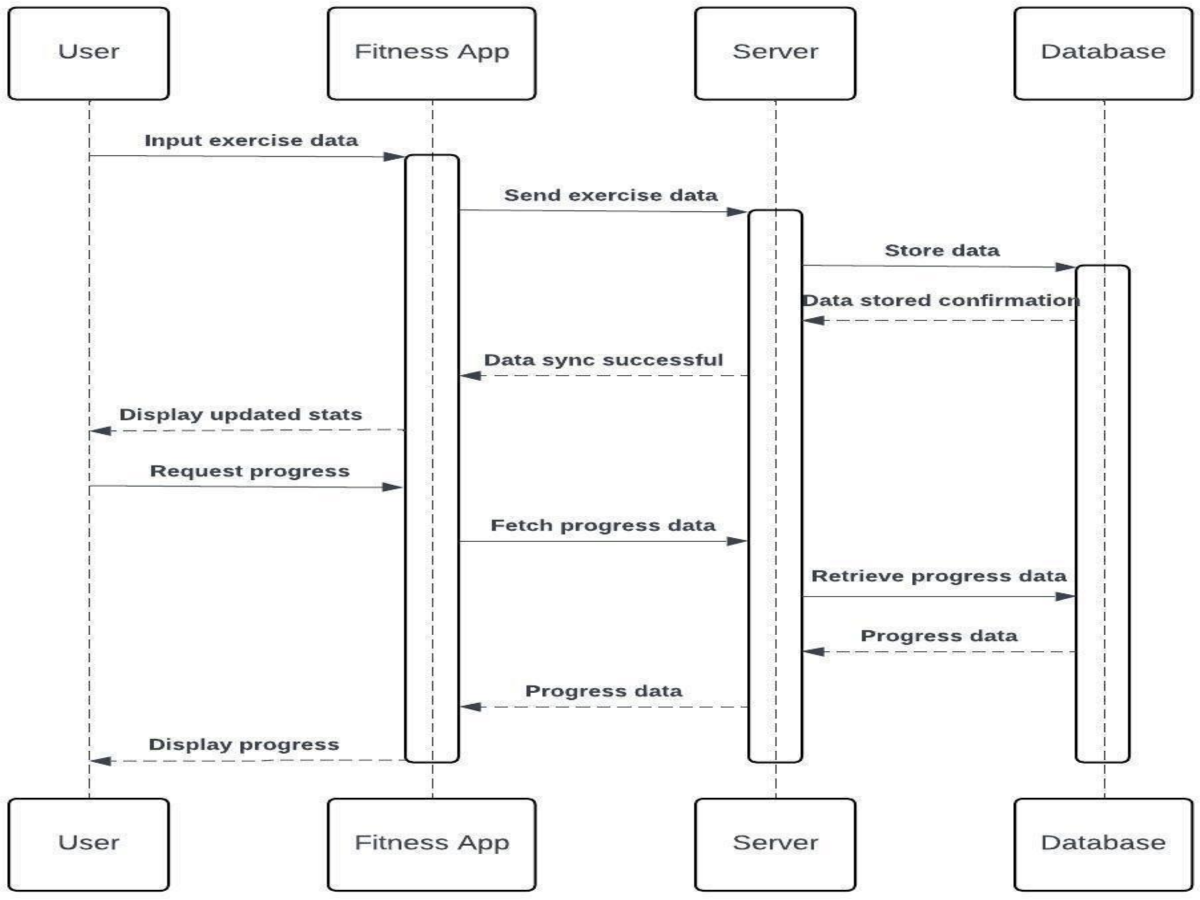


Fig 4: Class Diagram

5.3 Sequence Diagram

A user interacting with a fitness tracker is depicted by the actor in the diagram. This object stands in for the tracking app, which is in charge of handling all the required activities performed by the user.

You can see the user entering his workout data at the very beginning of the system's flowchart. The fitness monitoring software takes the user's instruction, runs it through the workout module, transforms the data to statistics, and then displays the results to the user in the form of graphs. The fitness monitoring software takes the user's intent into account and then uses its accessible modules and APIs to execute the appropriate actions.



.

Fig 5 : Sequence Diagram

5.4 Deployment Diagram

The deployment diagram illustrates the actual process of deploying the Virtual Assistant program and its modules to various nodes. An example of a user device might be a computer, tablet, or smartphone that the user uses to communicate with the Virtual Assistant. Whether it's a server in the cloud or on-premises, the Virtual Assistant's server stands for the machine running the Virtual Assistant program.

To perform its tasks, the Virtual Assistant's app relies on a number of third-party modules, including those for speech recognition, text-to-speech, web browsing, operating system, music player, email and weather API. Depending on the system's needs, these components might be installed on various nodes or computers.

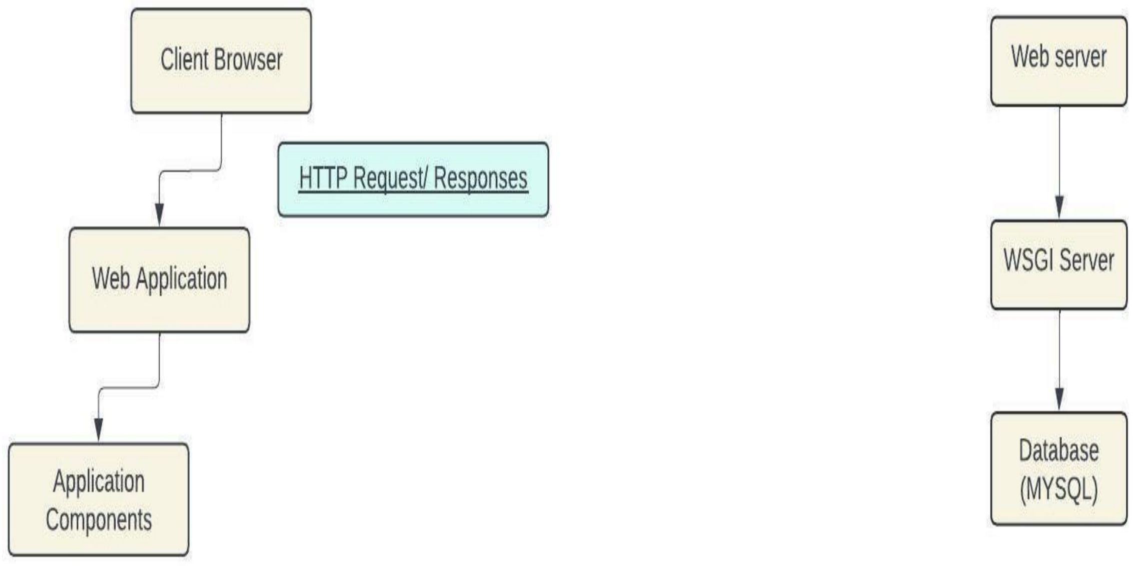


Fig 6 : Deployment Diagram

5.5 Activity Diagram

If you want to see what the Fitness Monitoring app does and how it all works, an activity diagram is a good place to start. It shows the different parts of the system as well as how they work together to make the app work.

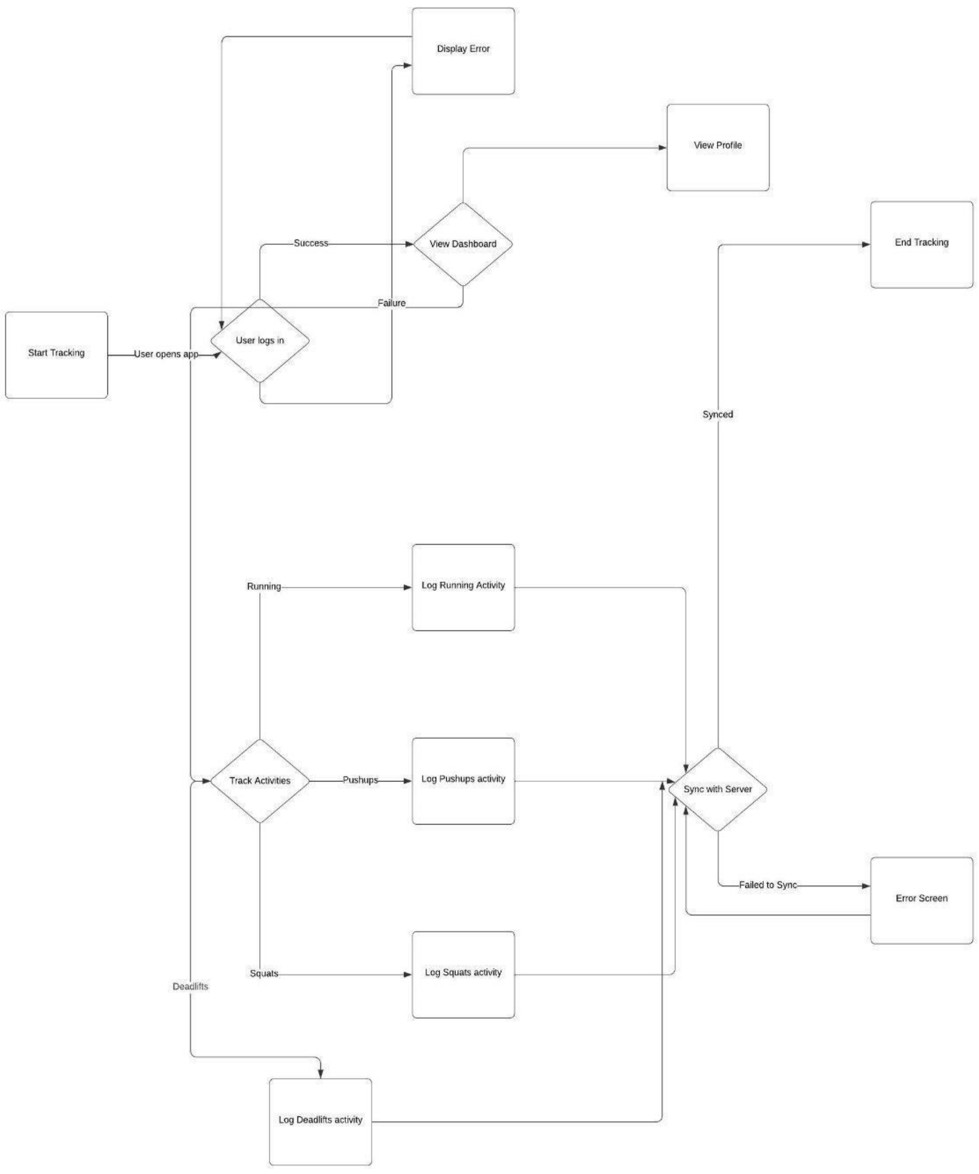


Fig 7 : Activity diagram

MODULAR DESCRIPTION

MODULES DEVELOPED IN THIS PROJECT

6.1.1 USER AUTHENTICATION MODULE

An essential part of any fitness monitoring program, the User Authorization Module restricts access to personal information and keeps user data safe. If you're developing an app to track your fitness, here is how you build a strong User Authentication Module.

Purpose: Oversees the process of registering, logging in, and securing users. Guarantees that authorized users are the only ones who can access user data.

Features: Options for social network integration and email/password user registration are available.
Protected login and process administration.
Systems for recovering lost passwords and authenticating users.

Technologies: bcrypt for password hashing, SSL/TLS for safe information transmission, and OAuth for social authentication.

Fig 8 : User Authentication Page

1. Requirements Definition

Compliance with rules (such as GDPR and HIPAA) and security of personal information are essential components of a secure system.

User-Friendly and Seamless Authentication Process Is A Must For A Good Experience For Users.

2. Choosing Authentication Methods

The standard authentication method: an email address and a password.

Make it easier for consumers to sign up and log in by letting them use their Facebook, Google, or other social network accounts.

A safe and fast way to authenticate, particularly for mobile devices, is via biometric authentication methods like fingerprint scanning or face recognition.

To further safeguard your account, you may set up two-factor authentication (2FA). This method often involves receiving a code by text message or using an authentication app.

3. Design the Authentication Flow

Describe the entire authentication process, beginning with account creation and ending with password recovery:

The first step in registering is to make an account creation form. Name, email, password, and permission to use this site's terms and privacy policies must be filled out.
Step One: Make a Login Form for Users to Fill Out. For better usability, think about adding "remember me" and similar features.

For users who have forgotten their passwords, it is important to have a secure mechanism in place. This process should involve authentication inquiries or verification via email links.

4. Developing the Authentication System

Backend Configuration: Establish a safe method of authentication on the server side. Make use of up-to-date frameworks and libraries that offer features like CSRF (Cross-Site Request Forgery) security, session management, and secure password hashing (e.g., bcrypt).

Integrating Databases: Safely keep user credentials. Always use hashes instead than storing passwords in plain text.

Authentication APIs: Create RESTful APIs that mobile clients and other front ends may utilize for authorization, login, and password reset.

5. Integrate with Front End

Developers working on the front end should make sure the login, password recovery, and registration screens are easy to use and quick to react. For a better user experience, implement client-side validation.

Implement a secure method of managing sessions by utilizing tokens such as JWT (JSON Web Tokens) or other comparable protocols.

6. Testing and Quality Assurance

Complete vulnerability scanning and penetration testing as part of your security testing to make sure your system is bug-free.

Make sure that all of the user flows (registration, login, logout, and credential recovery) function properly across all devices and browsers by conducting functional testing.

For optimal performance, test the authentication system under heavy use to make sure it can manage the anticipated volume.

7. Ongoing Maintenance and Updates

Be on the lookout at all times for signs of intruders or other security breaches. To stay ahead of emerging security dangers and make use of emerging technology, it is important to regularly upgrade authentication techniques and processes.

8. Documentation

User Documentation: Help users understand and implement their authentication settings by providing them with clear instructions and assistance.

Documentation for Developers: Keep comprehensive documentation about the architecture of the authenticating system, API ends, and security mechanisms.

Protecting customers' personal health and fitness information and earning their trust are two of the most important things you can do for your app's success. A strong User Authentication Module may help you achieve these goals.

6.1.2 ACTIVITY TRACKING MODULE

At the heart of every fitness monitoring software is the Activity monitoring Module, which records and analyzes user actions to reveal patterns, draw conclusions, and inspire better living. In order to provide users with a complete picture of their activity patterns, this module has to handle data on physical activities rapidly and reliably. A well-organized plan for creating an effective Activity Tracking Module is as follows:

1. Requirements Definition

Identify the activities that the module needs to track and the data that it should collect for each action. This data should include things like distance, duration, calories burnt, heart rate, and steps. Examples of such activities include walking, jogging, cycling, pushups, squats, and deadlifts.

The module must be able to process data in real-time without experiencing major delays and must function consistently across a wide range of environments and weather situations.

2. Data Collection and Processing

Data Accuracy: To make sure activity monitoring is accurate, use algorithms to remove noise and fix mistakes in sensor data.
Activity Recognition: Identify various forms of physical activity from sensor data using machine learning models.

3. Features Development

Keep tabs on your users in real time so they can see how they're doing while they work out.
Analysis of Historical Data: Make it possible to see data on past actions, so users can see how far they've come.
A formula or application programming interface (API) that predicts calories burnt depending on exercise type, time frame, user size, and effort should be integrated.

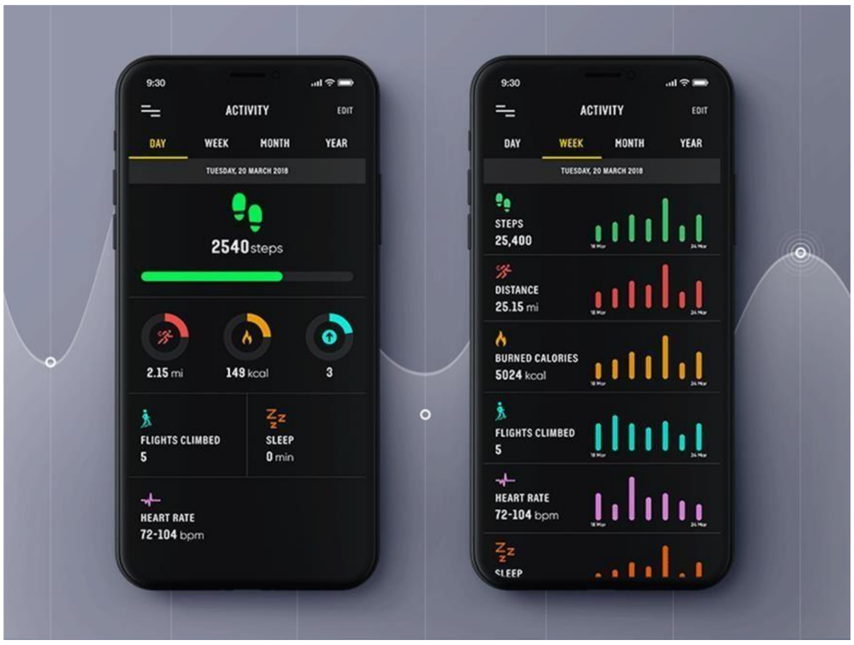


Fig 9 : Activity Tracking Module

4. Testing and Quality Assurance

Test each part of the module separately to make sure it works as expected. This is called unit testing.
The purpose of integration testing is to guarantee that the app's many modules, including the activity tracking one, are fully compatible with one another.
Test with Actual Users: Have actual users test the module's usability to make sure it's easy to use and fits their demands.

5. Security and Privacy

Protect sensitive user data, particularly that transferred via wearables, by implementing appropriate security measures.

Make sure the module informs users about data usage and obtains essential consents to comply with relevant data protection requirements like GDPR or HIPAA.

6. Enhancements and Scalability

Maintaining the module's scalability means making sure it can manage more user data as the app's user base expands.

Consider adding more sophisticated activity recognition algorithms or integrating with other kinds of wearable devices as potential future upgrades.

By methodically taking care of these details, the Activity monitoring Module can fulfill its primary function in a fitness monitoring app, which is to increase user engagement and give useful information about their physical activity.

6.1.3 Data Analysis and Reporting Module

An essential component of any fitness tracking software is the Information Analysis and Reporting Module, which takes the raw data gathered by the app and turns it into insightful visuals and useful insights. Users are able to track their health, progress, and performance with the aid of this module. A detailed blueprint for creating a powerful Analysis of Data and Reporting

Module is this:

Fig 10 : Data Analysis and reporting module

1. Requirements Gathering

Needs of Users: Determine what users care most about, such as patterns in their activity levels, how far along they are in reaching their objectives, and any relationships between various health indicators (such the quality of their sleep and the amount of exercise they do, for example).

Requirements for Data: Find out what kinds of information the unit needs to handle, such the number of steps taken, the number of calories burnt, the length of time exercised, and health

2. Designing the Reporting Interfaces

User-Friendly Dashboards: Make your dashboards easy to use by making them both visually appealing and functionally rich. Graphs, charts, and heat maps are all acceptable visual representations to use into these.

Personalization: Let users choose their preferred data to present and the format of metrics like sleep patterns and heart rate in reports.

6.1.4 Goal Setting Module

In order to help users establish and work toward personal health and fitness goals, the Set Goals Module is an essential component of any kind of fitness monitoring software. Users should be able to personalize their objectives according to their tastes and health data, therefore this module has to be both adaptable and easy to use. A well-designed and executed Set Goals Module looks like this:

1. Requirements Definition

Goal kinds: List the many goal kinds that users could create for themselves, such as the number of steps they take each day, the amount of time they exercise each week, weight reduction objectives, or even more specific accomplishments like finishing a 5k.
The user must be asked to provide some information in order to establish a goal. This information may include the objective's value, a timescale, and any particular requirements or characteristics.

2. Designing the User Interface

Exciting and Participatory User Interface (UI): Design an aesthetically pleasing and intuitive interface. Help people easily create goals by including forms, sliders, and drop-down choices. Build a system that displays an overview screen or a message of congratulations as soon as a goal is set; this will serve as immediate feedback.

Fig 11 : Goal Setting Module

3. Goal Customization and Flexibility

Personalized Goals: Allow users to create fully customized goals based on their personal fitness data and past performance.

4. Integration with Tracking and Notifications

Data Integration: Ensure that the goals set by users are closely integrated with the activity tracking module so that progress can be automatically tracked and updated.

5. Progress Tracking and Visualization

Graphs and progress bars are examples of dynamic visual representations that may be used to track users' progress towards their goals in real-time.

Make it possible for users to compare their present progress to their prior performance by comparing historical data.

If you follow these instructions, the Ability to Set Goals Module in your fitness tracking app may be a great tool for helping users set specific goals and maintain motivation while they work toward those goals. Not only does this module boost the app's total value, but it also promotes interaction between users.

EXISTING MODULES USED

6.2 Existing Modules

Historical fitness monitoring applications had a wide range of modules, some of which provided the framework for the sophisticated features seen in today's tracks. An outline of the typical components seen in older fitness monitoring apps is as follows:

6.2.1 Basic Activity Tracking Module

The goal was to record basic data on physical exercise.

Features: Using less complex sensor technologies and a lack of complex algorithms, this module usually counted steps and recorded basic workouts.

6.2.2 Simple Health Monitoring Module

I tracked vital signs and other basic health parameters.

Features: Older versions didn't have as much detail or precision, but they did measure heart rate and maybe even some sleep. For example, instead of doing thorough analysis of each stage of sleep, sleep tracking was frequently restricted to movement-based evaluations.

6.2.3 Caloric Tracking Module

Objective: Assisted users in maintaining a healthy weight by monitoring calorie consumption and expenditure.

Beneficial Features: Users have the option to manually record their food consumption and physical activity in order to calculate their net calories balance. These sections required more manual intervention from the user.

6.2.4 Goals Setting Module

Among its features is the ability for users to establish simple objectives, such as increasing their step count or decreasing their weight. Even without the ability to set individual goals, the app's operation remained simple.

6.2.5 Workout Logging Module

Designed to facilitate the manual input of exercise specifics.

Functionality: Users have the option to input exercise kinds, durations, and maybe even extra workout remarks. There was no way to monitor data in real-time or connect this module to other devices.

6.2.6 Data Syncing Module

Designed to facilitate the manual input of exercise specifics.

Functionality: Users have the option to input exercise kinds, durations, and maybe even extra workout remarks. There was no way to monitor data in real-time or connect this module to other devices.

6.2.7 Basic Notification Module

Served as a rudimentary notice and reminder system.

Notifications used to be more generic and less intelligent than the ones users get now; for example, they may remind users to log meals or to exercise after a particular amount of inactivity.

6.2.8 User Profile Module

Function: Preferences and data stored for users.

Features: The user could input their year of birth, weight, & height into the app, and it would utilize that data to determine how many calories they require and other health indicators.

6.2.9 Social Sharing Module

Goal: Permitting the posting of accomplishments on various social media platforms. Lacking the robust social networks and community challenges seen in competing applications, features are restricted to the simple sharing of accomplishments such as reaching a step goal or a weight reduction objective.

The fundamental features of previous fitness monitoring applications were these sections. Enhanced features such as GPS tracking, integrating health tracking, insights powered by artificial intelligence, and instantaneous sync across various devices and platforms are the result of system upgrades brought about by technological advancements, more mobile capacities, and user input. As a result of these developments, contemporary fitness applications are more engaging, precise, and useful for encouraging health and wellness.

FEASIBILITY STUDY

Feasibility Study:

The purpose of a feasibility study is to determine the project's chances of success by analyzing all of the pertinent aspects, such as financial, technical, legal, and time-related ones. In order to weigh the benefits and drawbacks of a project before devoting significant resources to it, project managers conduct feasibility studies. The usefulness of an idea may be determined with the help of a feasibility study. In case the initial plan doesn't work out or unforeseen problems arise, it's wise to have a backup plan.

The financial implications of the proposed system are the primary focus of a practicability study. It is possible to learn whether the proposed system is feasible by looking at the findings of the feasibility study. The feasibility study's good findings will allow us to move further with system development; otherwise, the project should not be undertaken. Project feasibility, or the likelihood that the system will be useful to the company, is one area that preliminary research looks at.

In order to add new modules and troubleshoot already functioning systems, the feasibility study primarily aims to assess the technical, operational, and economic feasibility. If there are endless resources and time, any system can be implemented. As part of the preliminary investigation's practicability research, there are unit aspects:

7.1 Operational practicability

7.2 Economical practicability

7.3 Technical practicability

7.1 Operational Feasibility:

A system's operational scope is evaluated in an operational feasibility study. The operational feasibility of the proposed system need to be high.

• If there are any errors during registration, the user will be notified.

• It has great usability.

Quick response time is a feature of this equipment.

• The application's design prioritizes ease of use, allowing users to easily find the information they need.

The user's only responsibility is with the program itself; all they have to do is click on the buttons to perform tasks like downloading, chatting, searching, and arranging diets, among others. Therefore, the proposed system may be implemented.

7.2 Economical Feasibility:

Finding the anticipated system's favorable economic edges is the goal of the economic practicability evaluation.

To implement the planned system, you'll need open-source programming tools and code, such as Humanoid Studio 2010, which may be found online for free.

We require a variety of resources, including as computer systems, internet connections, recommended space, and memory speed, to create the anticipated system.

When we compare these costs to the predicted system, we find a number of benefits, including:

• Data may not be available promptly because the current system is manual. Nevertheless, because the anticipated systems are computerized, we may circumvent all of the constraints of the current systems. Reduced labor is one benefit of this technique.

• People's time can be saved by using this system.

We therefore conclude that the proposed system is economically feasible, taking into account all of the aforementioned factors and conducting thorough examination with varying resource expenditures.

7.3 Technical Feasibility:

Gaining a grasp of these technological resources and how they relate to the predicted needs of the proposed system is the main focus of the technical practicable evaluation. It's an evaluation of the software and hardware suite and how well it satisfies the requirements of the intended system. The java program is responsible for the development of these strategies. Since we choose to take our time learning about these technologies, they are easy to pick up and might potentially advance at a breakneck speed. Anyone using the system throughout development and deployment will be able to access this webpage online.

Because it is primarily an online computer software, it gives users easy access. With the help of these roles, users would be granted permission.

Hence, it offers the technological assurance of precision, carelessness, and safety. So, the proposed system may be implemented.

CODING AND OUTPUT SCREENS

8.1 Backend Code

```
    From flask import Flask,request,render_template,url_for,redirect,flash,session
from flask_session import Session
import mysql.connector
from dmail import sendmail
from key import secret_key,salt1,salt2
from itsdangerous import URLSafeTimedSerializer
from stoken import token
from datetime import datetime
from flask import abort


app = Flask(__name__)
app.secret_key = secret_key
app.config['SESSION_TYPE']='filesystem'
mydb                                                                    =
mysql.connector.connect(host='localhost',user='root',password='Likhitha',db='fitn
ess_tracker')

@app.route('/')
def home():
    return render_template('title.html')

@app.route('/homepage')
def homepage():
    if session.get('user'):
        return render_template('index.html')
    else:
        return redirect(url_for('login'))

@app.route('/register',methods=['GET','POST'])
def register():
    if request.method=='POST':
        username=request.form['name']
        password=request.form['password']
        email=request.form['email']
        gender=request.form['gender']
        phone=request.form['phone']
        print(request.form)
        cursor=mydb.cursor(buffered=True)
        cursor.execute('select count(*) from user where username=%s',[username])
```

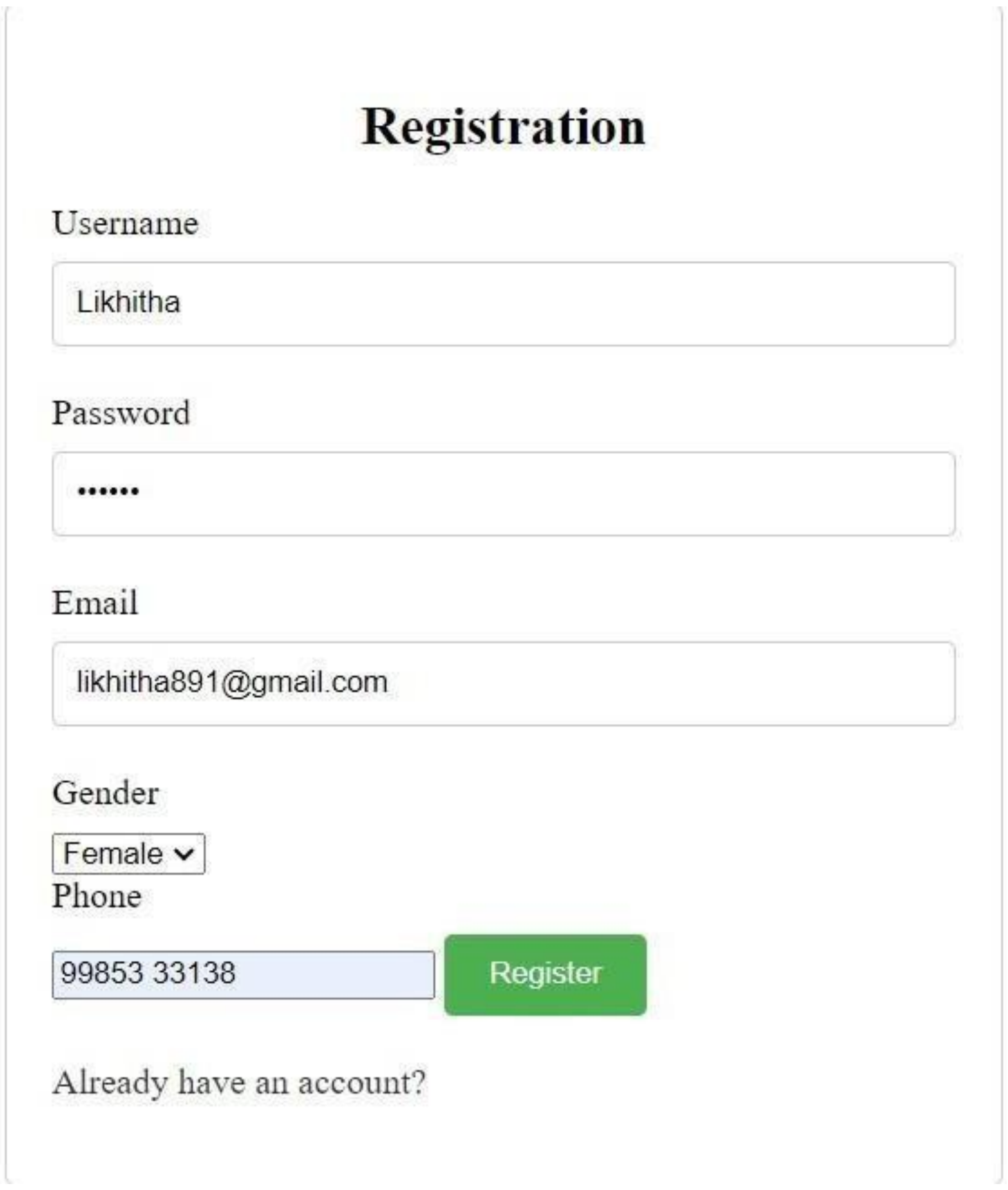
```
 count=cursor.fetchone()[0]
        cursor.execute('select count(*) from user where email=%s',[email])
        count1=cursor.fetchone()[0]
        cursor.close()
        if count==1:
            flash('username  already  in use')
            return render_template('register.html')
        elif count1==1:
            flash('Email already in use')
            return render_template('register.html')

data={'username':username,'password':password,'email':email,'gender':gender,'pho
ne':phone}
        subject='Email Confirmation'
        body=f"Thanks for signing up\n\nfollow this link for further steps-
{url_for('confirm',token=token(data,salt=salt1),_external=True)}"
        sendmail(to=email,subject=subject,body=body)
        flash('Confirmation link sent to mail')
        return redirect(url_for('login'))
    return render_template('register.html')

@app.route('/confirm/<token>')
def confirm(token):
    try:
        serializer=URLSafeTimedSerializer(secret_key)
        data=serializer.loads(token,salt=salt1,max_age=180)
    except Exception as e:
        print(e)
        return 'Link Expired register again'
    else:
        cursor=mydb.cursor(buffered=True)
        username=data['username']
        cursor.execute('select count(*) from user where username=%s',[username])
        count=cursor.fetchone()[0]
        if count==1:
            cursor.close()
            flash('You are already registerterd!')
            return redirect(url_for('login'))
        else:
            cursor.execute('insert into user
values(%s,%s,%s,%s,%s)',[data['username'],data['password'],data['email'],data['ge
nder'],data['phone']])
            mydb.commit()
```

```
   cursor.close()
            flash('Details registered!')
            return redirect(url_for('login'))

@app.route('/login',methods=['GET','POST'])
def login():
    if session.get('user'):
        return redirect(url_for('homepage'))
    if request.method == 'POST':
        username=request.form['name']
        password=request.form['password']
        cursor = mydb.cursor(buffered=True)
        cursor.execute('select count(*) from user where username=%s and 
password=%s',[username,password])
        count=cursor.fetchone()[0]
        if count==1:
            session['user']=username
            return redirect(url_for('homepage'))
        else:
            flash('Invalid username or password')
            return render_template('login.html')
    return render_template('login.html')

@app.route('/logout')
def logout():
    if session.get('user'):
        session.pop('user')
        flash('Successfully logged out')
        return redirect(url_for('login'))
    else:
        return redirect(url_for('login'))

@app.route('/reset/<token>',methods=['GET','POST'])
def reset(token):
    try:
        serializer=URLSafeTimedSerializer(secret_key)
        id1=serializer.loads(token,salt=salt1,max_age=180)
    except:
        abort(404,'Link Expired')
    else:
        if request.method=='POST':
            newpassword=request.form['npassword']
            confirmpassword=request.form['cpassword']
```

```
 if newpassword == confirmpassword:
            cursor = mydb.cursor(buffered=True)
            cursor.execute('update user set password=%s where
username=%s',[newpassword,id1])
            mydb.commit()
            flash('Reset Successful')
            return redirect(url_for('login'))
         else:
            flash('Passwords mismatched')
            return render_template('newpassword.html')
      return render_template('newpassword.html')
from itsdangerous import URLSafeTimedSerializer

@app.route('/forget',methods=['GET','POST'])
def forgot():
   if request.method=='POST':
      id1=request.form['name']
      cursor = mydb.cursor(buffered=True)
      cursor.execute('select count(*) from user where username=%s',[id1])
      count=cursor.fetchone()[0]
      print(count)
      cursor.close()
      if count==1:
         cursor = mydb.cursor(buffered=True)
         cursor.execute('SELECT email from user where username=%s',[id1])
         email=cursor.fetchone()[0]
         cursor.close()
         subject='Forget Password'
         confirm_link = url_for('reset',
token=token(data=id1,salt=salt1),_external=True)
         body=f"Use this link to reset your password-\n\n{confirm_link}"
         sendmail(to=email,body=body,subject=subject)
         flash('Reset link sent check your email')
         return redirect(url_for('login'))
      else:
         flash('Invalid email id')
         return render_template('forgot.html')
   return render_template('forgot.html')


##This is Admin login Creadentials
@app.route('/admin',methods=['GET','POST'])
def admin():
  if request.method == 'POST':
        admin=request.form['name']
        password=request.form['password']
        if admin == 'Admin' and password == 'admin@123':
            session['admin']=password
            return redirect(url_for('viewallusers'))
        else:
            flash('Invalid username or password')
            return render_template('alogin.html')
    return render_template('alogin.html')
@app.route('/alogout')
def alogout():
    if session.get('admin'):
        session.pop('admin')
        flash('successfully log out')
        return redirect(url_for('home'))
    else:
        return redirect(url_for('alogin'))


@app.route('/addprofile',methods=['GET','POST'])
def addprofile():
    if session.get('user'):
        if request.method=="POST":
            height=float(request.form['height'])
            weight=float(request.form['weight'])
            age=int(request.form['age'])
            cursor=mydb.cursor(buffered=True)
            cursor.execute('INSERT INTO profiles (full_name, height, weight, age)
VALUES (%s, %s, %s, %s)',[session['user'],height,weight,age])
            mydb.commit()
            mydb.close()
            flash('profile added sucessfully')
            return redirect(url_for('viewprofile'))
        return render_template('addprofile.html')
    return redirect(url_for('login'))
@app.route('/viewprofile',methods=['GET','POST'])
def viewprofile():
    if session.get('user'):
        cursor=mydb.cursor(buffered=True)
        cursor.execute('select * from profiles where full_name=%s',[session['user']])
        data= cursor.fetchall()
        return render_template('viewprofile.html',d=data)
  return redirect(url_for('login'))

@app.route('/update_profile',methods=['GET','POST'])
def update_profile():
    if session.get('user'):
        cursor=mydb.cursor(buffered=True)
        cursor.execute('select * from profiles where full_name = %s',[session['user']])
        data = cursor.fetchall()
        if request.method=="POST":
            height=float(request.form['height'])
            weight=float(request.form['weight'])
            age=int(request.form['age'])

            cursor=mydb.cursor(buffered=True)
            cursor.execute("UPDATE profiles SET height = %s, weight = %s, age = %s
WHERE full_name = %s",[height,weight,age,session['user']])
            mydb.commit()
            flash('updated sucessfully')
            return redirect(url_for('viewprofile'))
        return render_template('update_profile.html',d = data)
    return redirect(url_for('login'))

@app.route('/addexercise',methods=['GET','POST'])
def addexercise():
    if session.get('user'):
        if request.method=="POST":

            exercises = request.form.getlist('exercises[]')
            durations = request.form.getlist('durations[]')
            l=[]
            for i in durations:
                if i=='':
                    i=0
                    l+=[i]
                else:
                    l+=[i]
            print(l)
            weight = float(request.form.get('weight')) # Convert weight to float
            check_in_time = request.form.get('check_in_time')

            total_duration_minutes = sum(map(int, l))

            # Calculate calories burned per minute based on weight (example formula)
```

```
calories_per_minute = weight * 0.1  # Example formula, adjust as needed

            # Calculate total calories burned
            total_calories_burned = total_duration_minutes * len(exercises) *
calories_per_minute
            weight_after_calories = weight - (total_calories_burned / 7700)  #
Assuming 7700 calories burn 1 kg
            log_date = datetime.now().date()
            # Calculate weight after calories burned
            cursor=mydb.cursor(buffered=True)
            cursor.execute('INSERT INTO exercise_logs (user_id,
total_duration_minutes, check_in_time, weight_kg, calories_burned, log_date)
VALUES (%s, %s, %s, %s, %s, %s)',
                           (session['user'], total_duration_minutes, check_in_time,
weight_after_calories, total_calories_burned, log_date))


            flash(f'Total duration: {total_duration_minutes} minutes.', 'info')
            flash(f'Total calories burned: {total_calories_burned} calories.', 'info')
            flash(f'User weight before exercise: {weight} kg.', 'info')
            flash(f'User weight after exercise: {weight_after_calories} kg.', 'info')
            mydb.commit()
            flash('data inserted sucessfully')
            return redirect(url_for('viewexerciselog'))

        return render_template('addexerciselog.html')
    return redirect(url_for('login'))
@app.route('/viewexerciselog')
def viewexerciselog():
    if session.get('user'):
        cursor=mydb.cursor(buffered=True)
        cursor.execute('select * from exercise_logs where
user_id=%s',[session['user']])
        data=cursor.fetchall()

        return render_template('viewexerciselog.html',d=data)
    return redirect(url_for('login'))
@app.route('/viewallusers')
def viewallusers():
    if session.get('admin'):
        cursor=mydb.cursor(buffered=True)
        cursor.execute('select * from exercise_logs')
        data = cursor.fetchall()
        return render_template('viewallusers.html',d=data)
    return redirect(url_for('login'))

from flask import session

@app.route('/set_goal', methods=['GET', 'POST'])
def set_goal():
    if session.get('user'):
        if request.method == 'POST':
            user_id = session.get('user')
            print("User ID from session:", user_id)  # Debugging: Print the user ID

            goal_type = request.form['goal_type']
            target_value = float(request.form['target_value'])
            start_date = request.form['start_date']
            end_date = request.form['end_date']
            cursor = mydb.cursor(buffered=True)
            cursor.execute('INSERT INTO goals (user_id, goal_type, target_value,start_date,
end_date) VALUES (%s, %s, %s, %s, %s)',
                           (user_id, goal_type, target_value, start_date, end_date))
            mydb.commit()
            flash('Goal set successfully')
            return redirect(url_for('homepage'))
        return render_template('set_goal.html')
    return redirect(url_for('login'))




# Add this part in your code where you initialize the database connection
# Define a function to create the goals table if it doesn't exist
def create_goals_table():
    cursor = mydb.cursor(buffered=True)
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS goals (
            id INT AUTO_INCREMENT PRIMARY KEY,
            user_id INT,
            goal_type VARCHAR(50),
            target_value FLOAT,
            start_date DATE,
            end_date DATE,
            FOREIGN KEY (user_id) REFERENCES user(id)
        )
    """)
    mydb.commit()
    cursor.close()

# Call the function to create the table when the application starts
create_goals_table()

import plotly.graph_objs as go

class DataAnalysis:
    @staticmethod
    def calculate_average_weight(user_id):
        # Example: Query database to get user's weight data
        weights = [70, 72, 71, 73, 75]  # Example weights
        if weights:
            average_weight = sum(weights) / len(weights)
            return average_weight
        else:
            return None

@app.route('/weight_trend_report')
def weight_trend_report():
    if session.get('user'):
        user_id = session['user']
        average_weight = DataAnalysis.calculate_average_weight(user_id)
        if average_weight is not None:
            # Generate plot
            fig = go.Figure(data=go.Scatter(x=[], y=[])) # Example plot
            fig.update_layout(title='Weight Trend Report', xaxis_title='Date',
yaxis_title='Weight (kg)')
            return render_template('report.html', plot=fig.to_html())
        else:
            return "No weight data available" # Handle case where no weight data is
available
    return redirect(url_for('login'))


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

8.2 Output Screens



Registration

Username

Likhitha

Password

••••••

Email

likhitha891@gmail.com

Gender

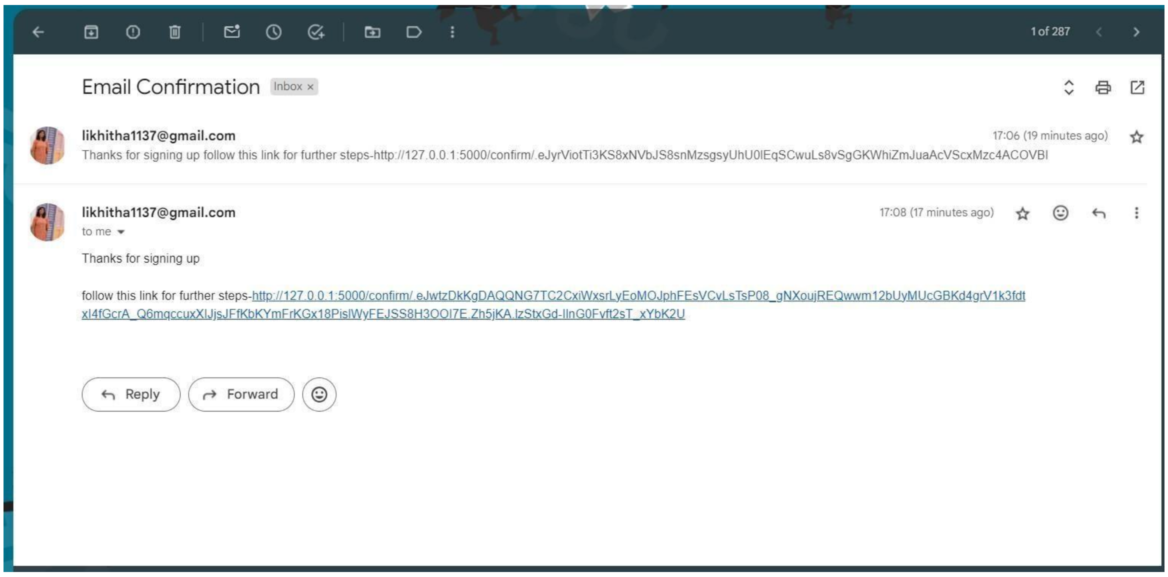
Female

Phone

99853 33138

Already have an account?

Fig 12 : Output Screen 1



Email Confirmation

likhitha1137@gmail.com 17:06 (19 minutes ago)
Thanks for signing up follow this link for further steps-http://127.0.0.1:5000/confirm/.eJyrViotTi3KS8xNVbJS8snMzsgsyUhU0lEqSCwuLs8vSgGKWhiZmJuaAcVScxMzc4ACOVBI

likhitha1137@gmail.com 17:08 (17 minutes ago)
to me

Thanks for signing up

follow this link for further steps-http://127.0.0.1:5000/confirm/.eJwtzDkKgDAQQNG7TC2CxiWxsrLyEoMOJphFEsVCvLsTsP08_gNXoujREQwwm12bUyMUcGBKd4grV1k3fdt
xl4fGcrA_Q6mqccuxXIJjsJFfKbKYmFrKGx18PislWyFEJSS8H3OOI7E.Zh5jKA.lzStxGd-IlnG0Fvft2sT_xYbK2U

Fig 13 : Output Screen 2

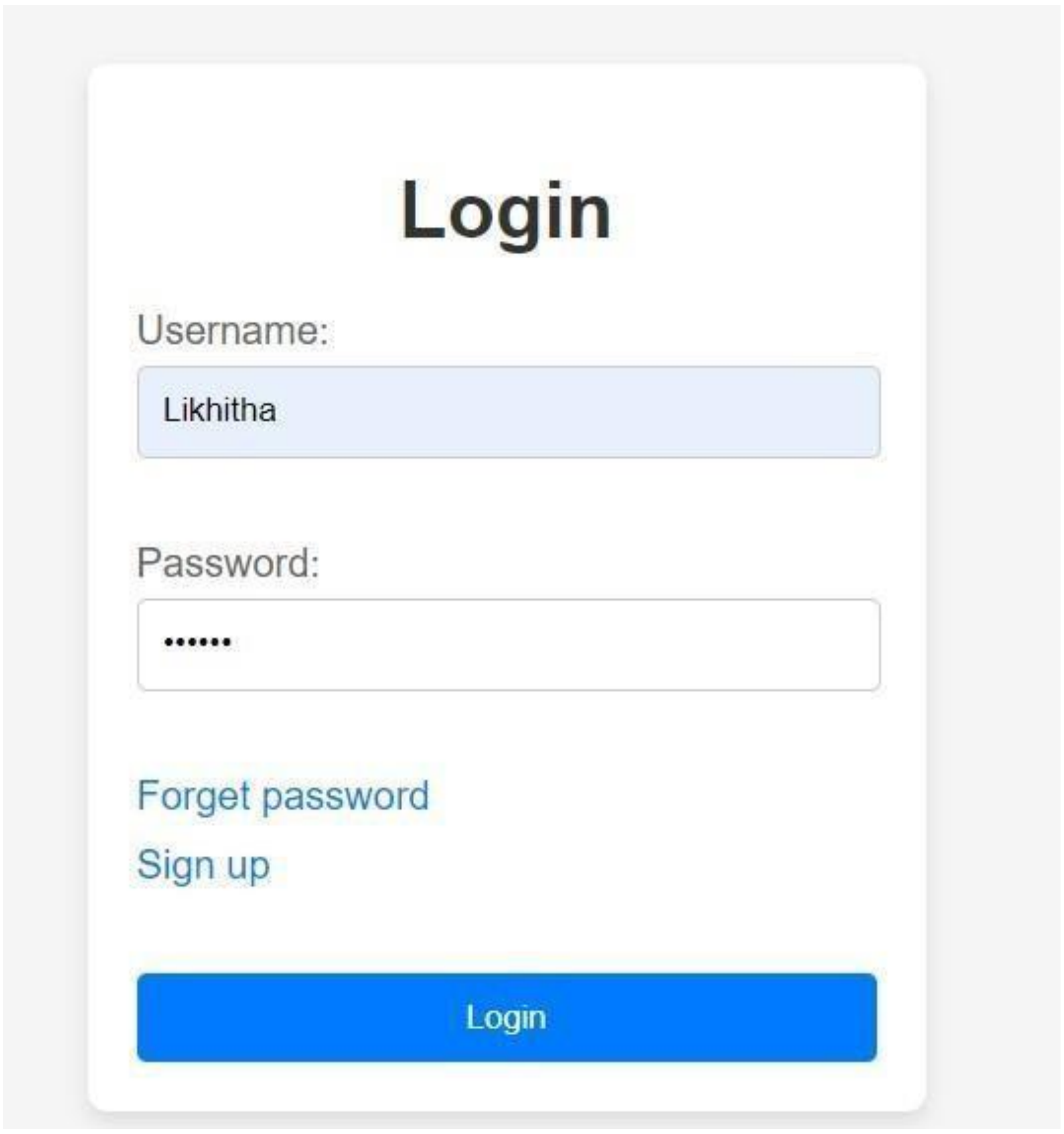


Fig 14 : Output Screen 3

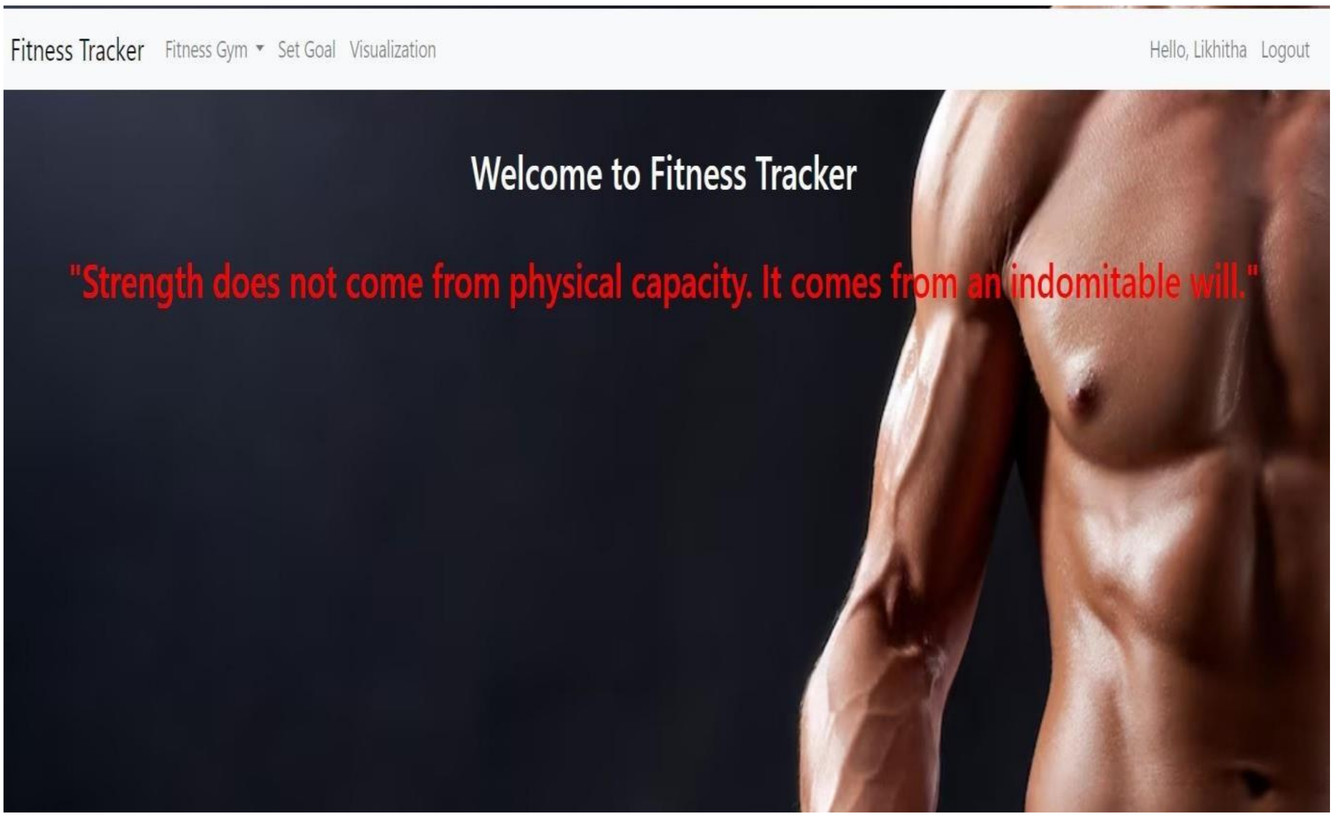
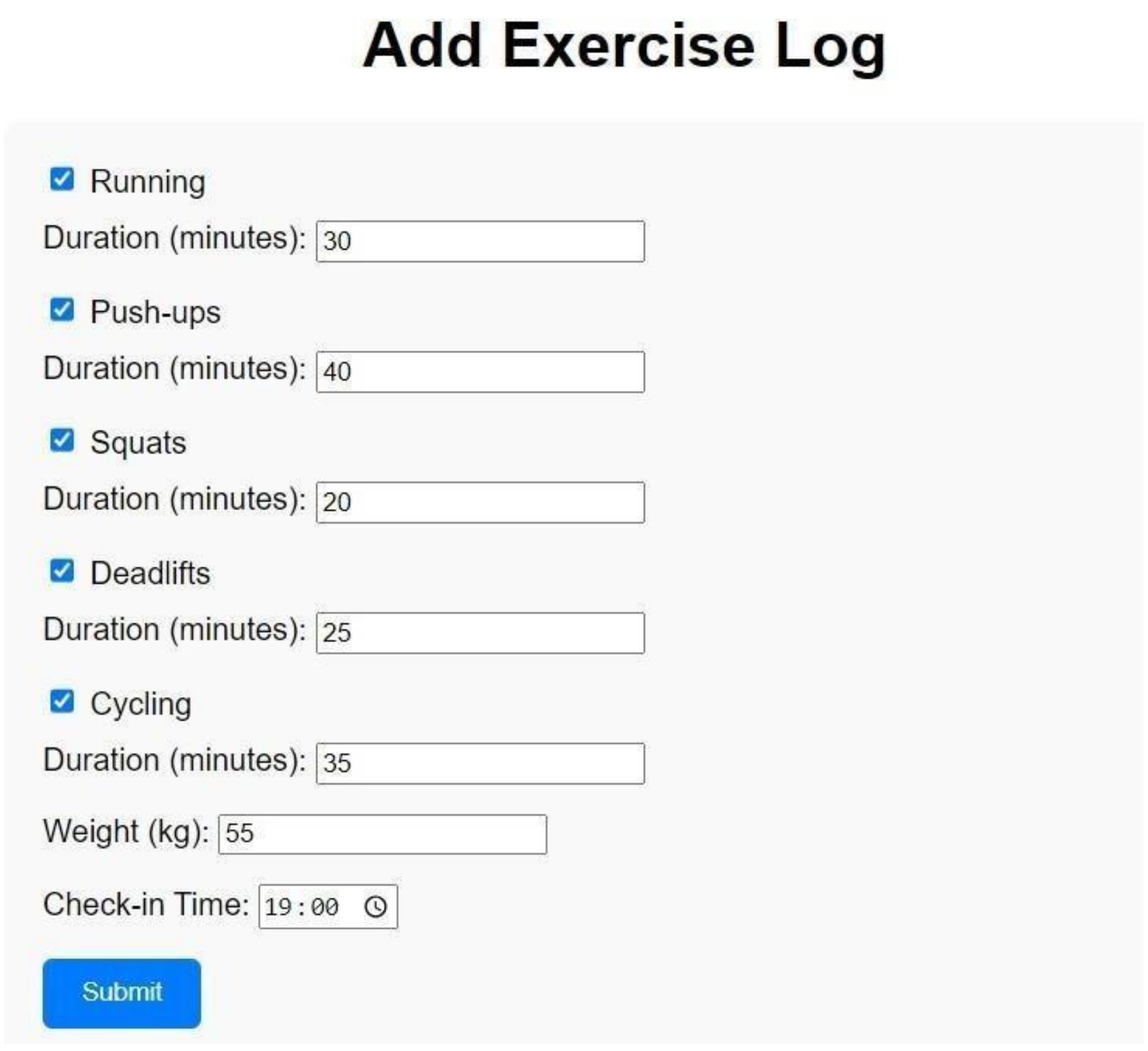

Fitness Tracker
Fitness Gym
Set Goal
Visualization
Hello, Likhitha
Logout
Welcome to Fitness Tracker
"Strength does not come from physical capacity. It comes from an indomitable will."

Fig 15 : Output Screen 4


Add Exercise Log

☑ Running

Duration (minutes): 30

☑ Push-ups

Duration (minutes): 40

☑ Squats

Duration (minutes): 20

☑ Deadlifts

Duration (minutes): 25

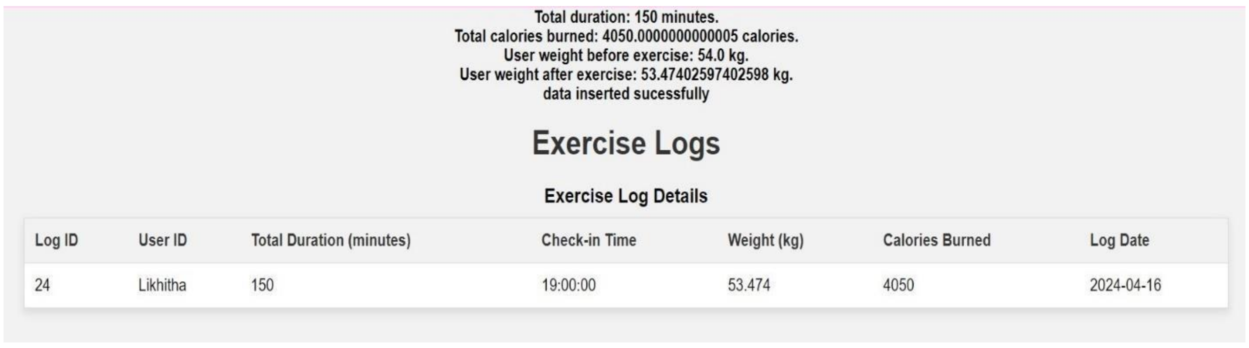
☑ Cycling

Duration (minutes): 35

Weight (kg): 55

Check-in Time: 19:00

Fig 16 : Output Screen 5



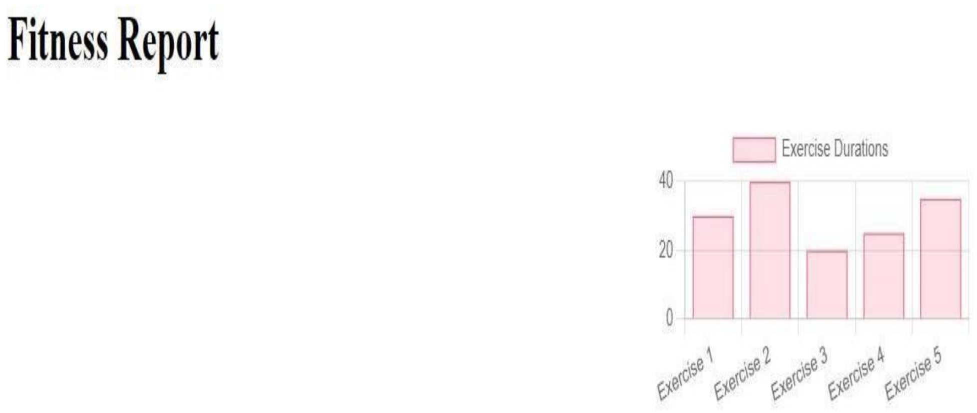
Total duration: 150 minutes.
Total calories burned: 4050.0000000000005 calories.
User weight before exercise: 54.0 kg.
User weight after exercise: 53.47402597402598 kg.
data inserted sucessfully

Exercise Logs

Exercise Log Details

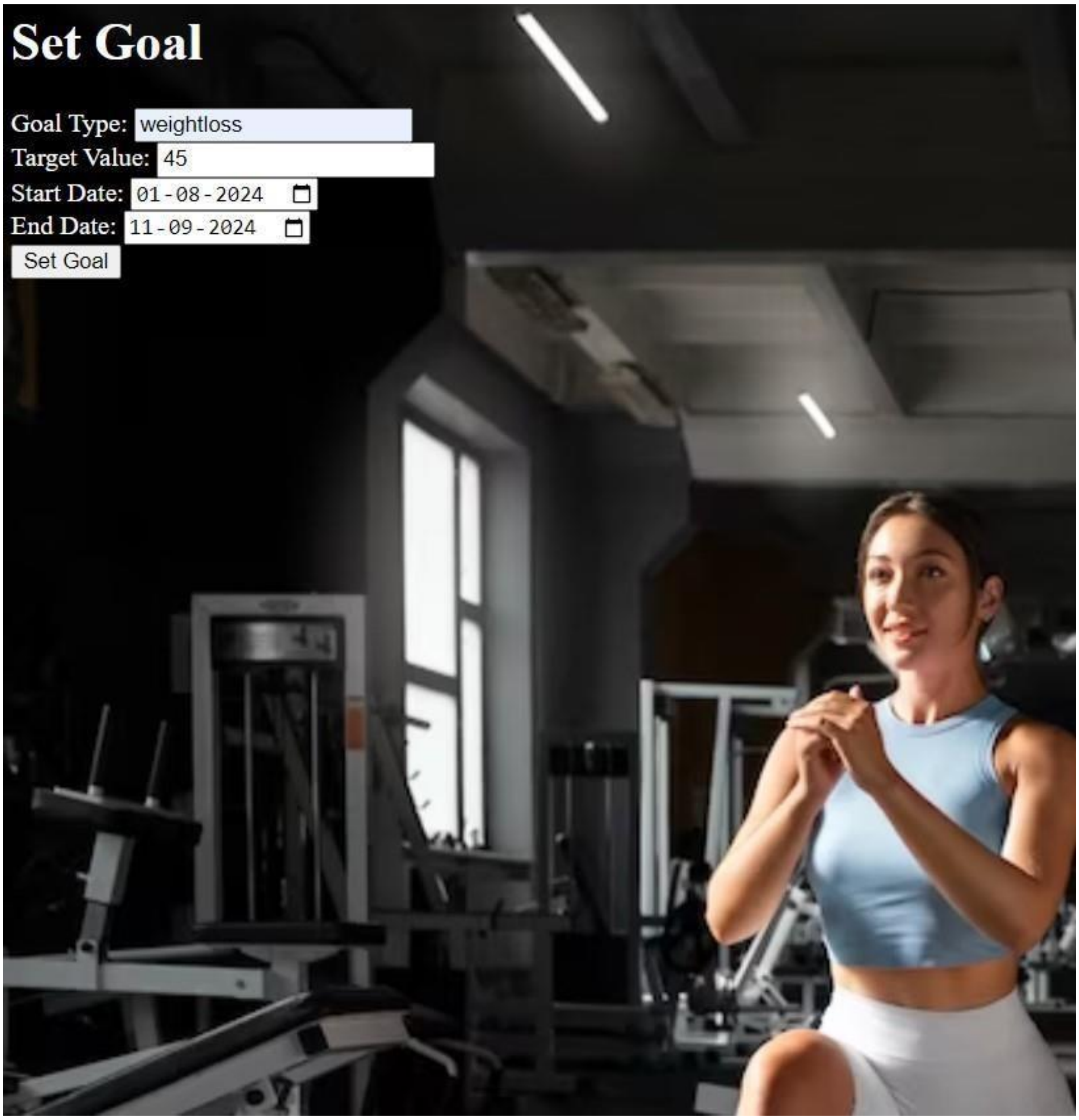
Log ID	User ID	Total Duration (minutes)	Check-in Time	Weight (kg)	Calories Burned	Log Date
24	Likhitha	150	19:00:00	53.474	4050	2024-04-16

Fig 17 : Output Screen 6



Fitness Report

Fig 18 : Output Screen 7



Set Goal

Goal Type: weightloss
Target Value: 45
Start Date: 01-08-2024
End Date: 11-09-2024

Fig 19 : Output Screen 8

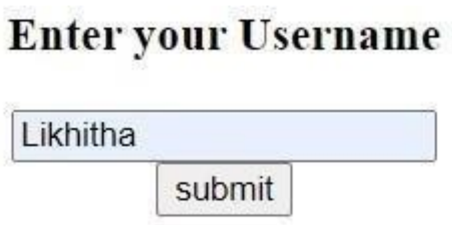


Fig 20 : Output Screen 9

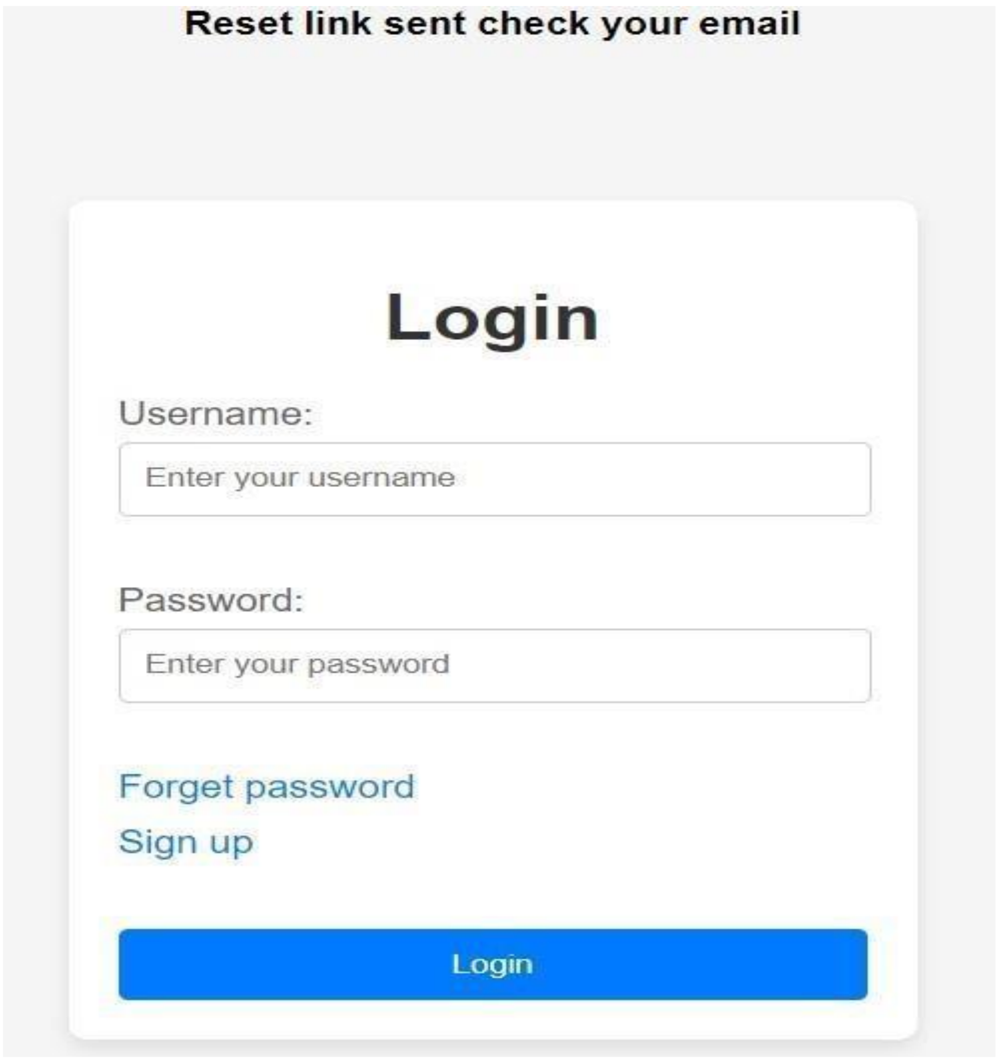


Fig 21 : Output Screen 10

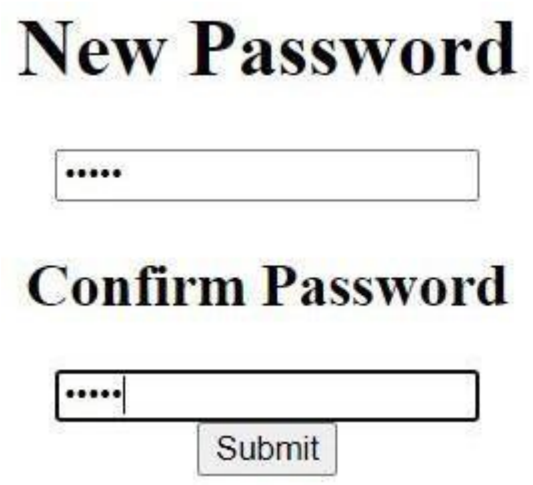


Fig 22 : Output Screen 11

SYSTEM TESTING

9. Testing

9.1 Introduction to Testing

Finding mistakes is what testing is all about. During testing, we look for any and all potential issues with a product or service. Assemblies, subassemblies, components, and the final product can all be tested in this manner. Testing is the act of putting software through its paces in order to make sure it doesn't crash or otherwise behaves in a way that users would find undesirable. Many different kinds of checks exist. A specific testing requirement is reported by each check sort.

The last step in ensuring software is of high quality, system testing involves reviewing the requirements, design, and code. The purpose of running a program for the intention of discovering mistakes is called testing. A good test will have a chance of discovering a mistake that has not been found before. Nothing is finished without testing, and the main goal of test is to find and fix errors in the created system. The system's success depends on testing. Code testing involves checking the produced system's logic. To do this, we run the entire program module by module in search of an error. Examining the requirements begins with the program's intended behavior and how it should function in different scenarios in order to carry out a specification test. System testing focuses on the integration of each element inside the system rather than the software in its whole. Making sure each module works with the others is the main issue. Locate the parts of the code where modules have varied data length, type, and element name requirements.

Following the created system's implementation, the most crucial tasks are testing and validation. The purpose of the system's testing is to find and fix any bugs in the final product. In order to identify the problematic modules, the software has to be run many times.

The overarching purpose and result of doing a test is known as the test objective. To make sure the program is bug-free before release, the goal of testing is to uncover as many software problems as possible.

• A test is considered successful if it finds an error that has not been found yet.To find an error, if one exists, an appropriate test scenario should have the potential to do so.

• There may be mistakes that the test is missing.
• The program meets all quality and reliability requirements.

9.2 Types of Testing

Unit Testing:

By creating test cases, unit testing ensures that the program's core logic is working as intended and that legitimate inputs result in valid outputs. It is important to check every choice branches and the internal code flow. Unit testing is the process of evaluating certain parts of an application's code. This step is performed following the conclusion of a single unit prior to integration. To do this intrusive structural exam, one must be familiar with its build.

By focusing on a single business process, application, or system configuration, unit tests are able to do fundamental component-level testing. The purpose of unit testing is to verify that all individual business process paths adhere to the established requirements and have well-defined inputs and outputs.

Integration Testing:

To verify that two or more pieces of software can function together as intended, developers create integration tests. As an event-driven process, testing cares primarily about the most fundamental results of windows or fields.
Although the components were adequate individually (as demonstrated by successful unit testing), integration tests indicate that the blend of components is proper and consistent. The goal of integration testing is to identify and fix issues that occur when different parts are used together.

Functional Testing:

The purpose of functional testing is to systematically prove that the tested functions are available and meet the criteria laid forth in the user manual, system documentation, business and technical specifications, and other relevant sources.
The focus of functional testing is on the following:
Valid Input: identified classes of valid input must be accepted.
Invalid Input: identified classes of invalid input must be rejected.
Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: system or process interfaces must be used. The goals, critical functionalities, or unique test cases are the focal points of functional test planning and execution. Additionally, testing should take into account systematic coverage of identifying business process flows, data fields, preset procedures, and consecutive processes. Finding the efficient value of existing tests and identifying more tests are both done before the functional assessment is finished.

System Testing:

When all parts of an integrated software system are tested, it guarantees that it will work as expected. It verifies a setup to make sure the outcomes are predictable and understood. The configuration-oriented integrated system test exemplifies testing systems. Process flows and descriptions provide the basis of system testing, with an emphasis on integration points and pre-driven procedure connections.

White Box Testing:

In white box testing, the software tester is familiar with the program's purpose, structure, and language, and has understanding of how the program works. It serves a function. For testing purposes, it fills in the gaps that a black box level cannot.

Black Box Testing:

If you want to test a software module's functionality, structure, or language but don't know anything about it, you should look at black box testing. To write a black box test, as with most other types of tests, you need a final source document, such a specifications or requirements document. This kind of testing involves seeing the programme in question as an opaque entity. It's impossible to "see" inside. Without taking the software's operation into account, the test simply supplies inputs and replies to outputs.

Regression Testing:

Regression testing involves checking pre-existing software programs for broken functionality after an update or modification. In addition to making sure that problems that have been previously eliminated remain dead, it also catches bugs that may have been mistakenly introduced into a fresh build or release candidate. It is possible to verify that recent modifications to a program have not led to a regression or rendered previously functional components useless by repeating testing situations that were originally created when identified issues were initially resolved. The use of a tool for automated testing is usually necessary since, although such tests may be executed manually on smaller projects, it is usually too tedious and difficult to contemplate running a series of tests every time a modification is made.

Smoke Testing:

One way to check if the software build you just delivered is stable is to use smoke testing. The QA team may move forward with software testing when smoke testing confirms everything is in order. It is comprised of a basic collection of tests that are executed on every build to ensure that the program functions as expected. Some other names for smoke testing include "Build Verification Testing" and "Confidence Testing.".

Alpha Testing:

Alpha testing is the initial comprehensive evaluation of a product to guarantee it satisfies the needs of the company and operates as intended. Employees from within the company usually carry it out in a controlled laboratory or performance setting. To make sure the product performs as expected, an alpha test is conducted.

Beta Testing:

As a subset of UAT, beta testing involves distributing a nearly completed product to a select sample of end customers for feedback on how well it works in practice. There is currently no agreed-upon method or template for conducting beta tests. You should ensure that the testing technique is pertinent to your testing objectives.

9.3 Test cases and Test Reports

Testing Unit: User Registration Page

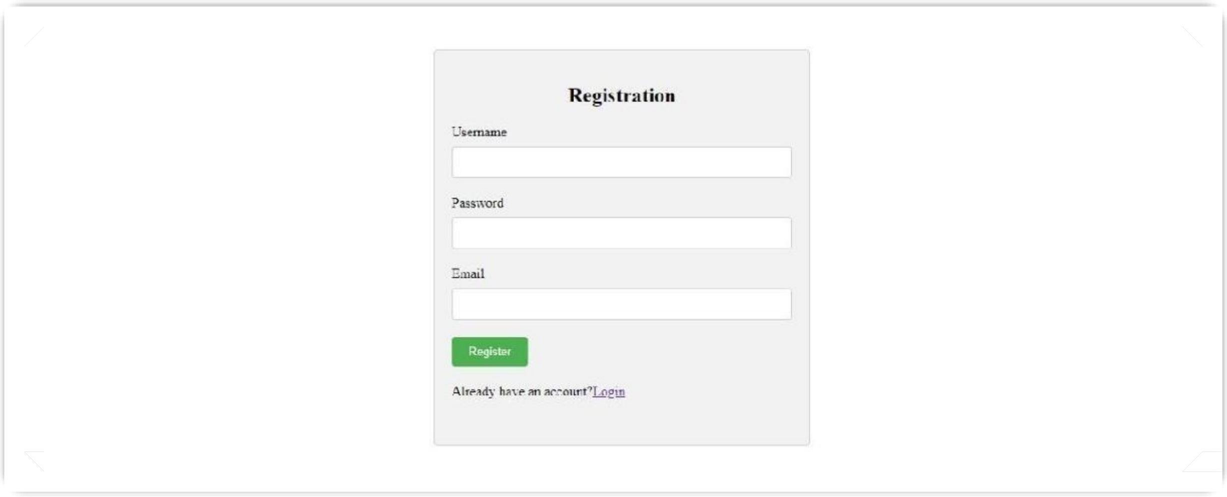
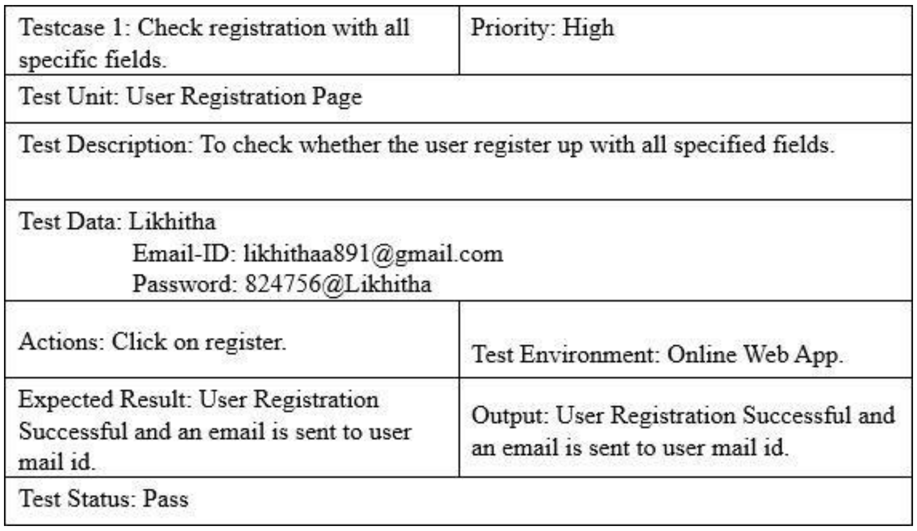


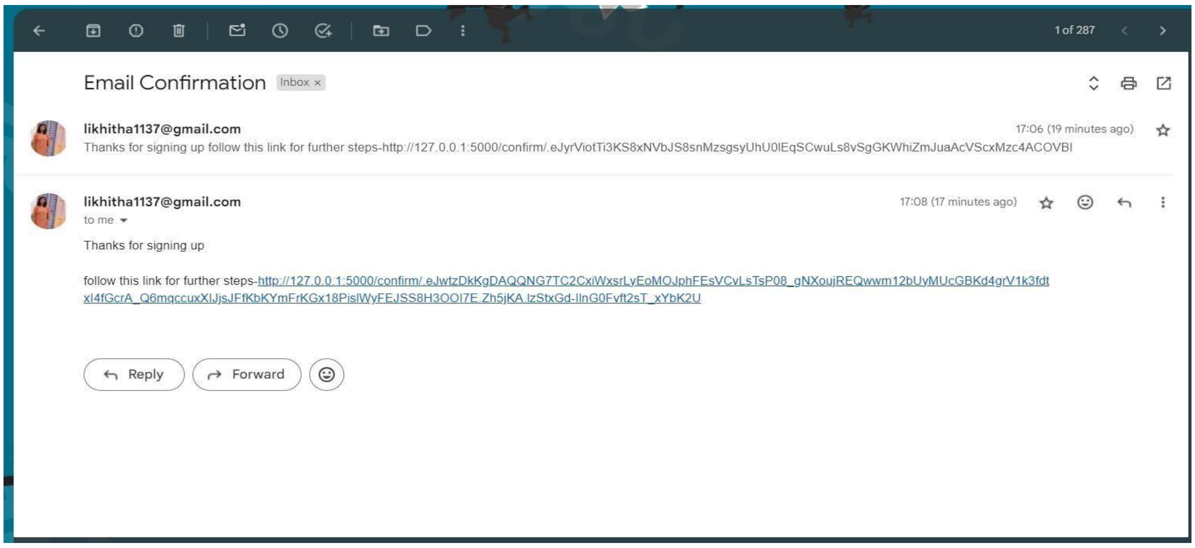
Table 1: First test case for User Registration Page.

Result is shown below.



Testcase 1: Check registration with all specific fields.	Priority: High
Test Unit: User Registration Page	
Test Description: To check whether the user register up with all specified fields.	
Test Data: Likhitha Email-ID: likhithaa891@gmail.com Password: 824756@Likhitha	
Actions: Click on register.	Test Environment: Online Web App.
Expected Result: User Registration Successful and an email is sent to user mail id.	Output: User Registration Successful and an email is sent to user mail id.
Test Status: Pass	

Result is shown below.



Email Confirmation

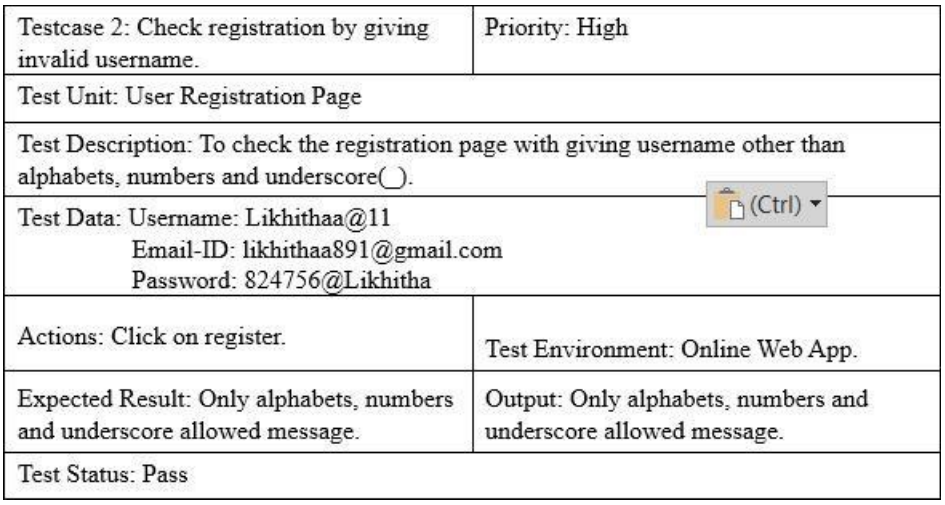
likhitha1137@gmail.com
17:06 (19 minutes ago)
Thanks for signing up follow this link for further steps-http://127.0.0.1:5000/confirm/.eJyrViotTi3KS8xNVbJS8snMzsgsyUhU0lEqSCwuLs8vSgGKWhiZmJuaAcVScxMzc4ACOVBI

likhitha1137@gmail.com
to me
17:08 (17 minutes ago)

Thanks for signing up

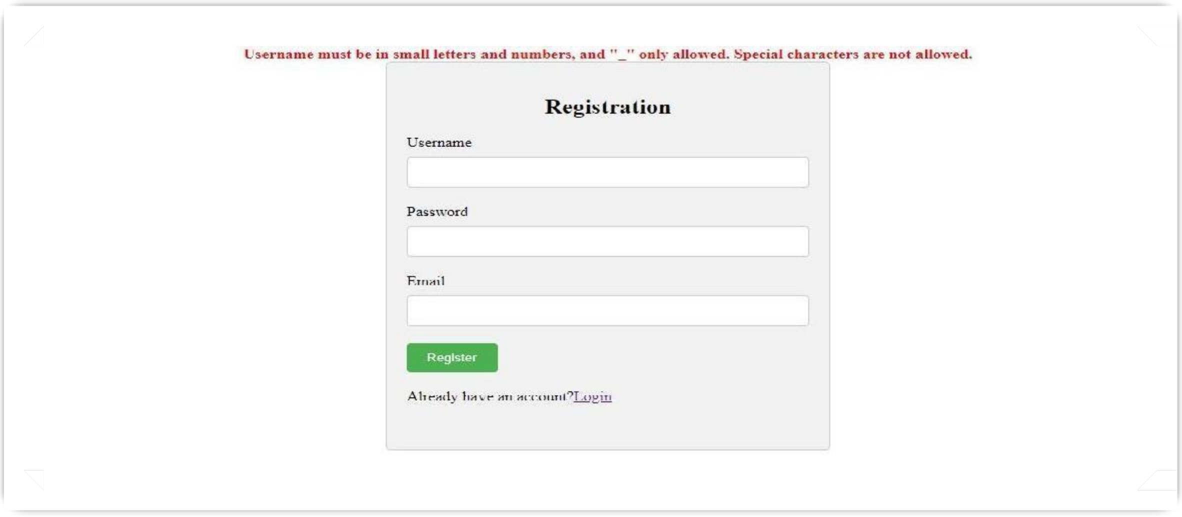
follow this link for further steps-http://127.0.0.1:5000/confirm/.eJwtzDkKgDAQQNG7TC2CxiWxsrLyEoMOJphFEsVCvLsTsP08_gNXoujREQwwm12bUyMUcGBKd4grV1k3fdt
xl4fGcrA_Q6mqccuxXIJjsJFfKbKYmFrKGx18PislWyFEJSS8H3OOI7E.Zh5jKA.IzStxGd-IlnG0Fvft2sT_xYbK2U

Table 2: Second test case for User Registration Page.



Testcase 2: Check registration by giving invalid username.	Priority: High
Test Unit: User Registration Page	
Test Description: To check the registration page with giving username other than alphabets, numbers and underscore(_).	
Test Data: Username: Likhithaa@11 Email-ID: likhithaa891@gmail.com Password: 824756@Likhitha	
Actions: Click on register.	Test Environment: Online Web App.
Expected Result: Only alphabets, numbers and underscore allowed message.	Output: Only alphabets, numbers and underscore allowed message.
Test Status: Pass	

Result is shown below.



Username must be in small letters and numbers, and "_" only allowed. Special characters are not allowed.

Table 3: Third test case for User Registration Page.

Testcase 3: Check registration by giving different password.	Priority: High
Test Unit: User Registration Page	
Test Description: To check the registration page with giving password without uppercase letter, a number and a special character.	
Test Data: Username: Likhitha Email-ID: likhithaa891@gmail.com Password: 824756@likhitha	

Result is shown below

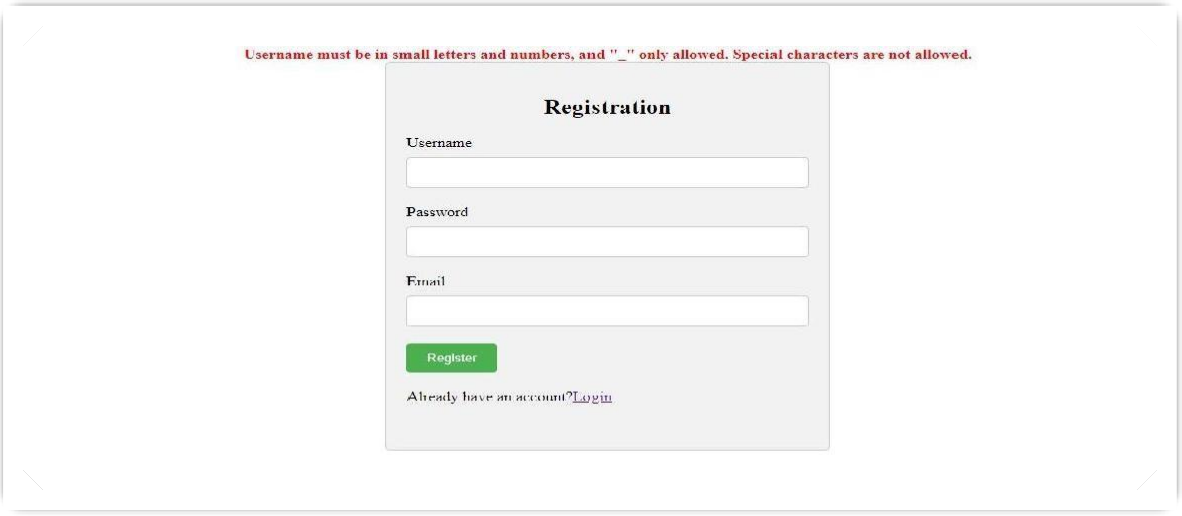
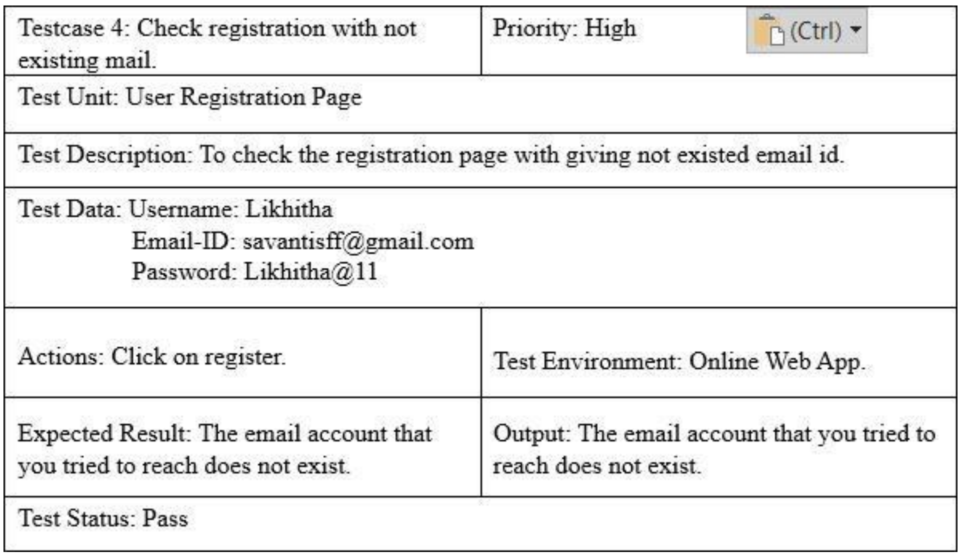

Username must be in small letters and numbers, and "_" only allowed. Special characters are not allowed.

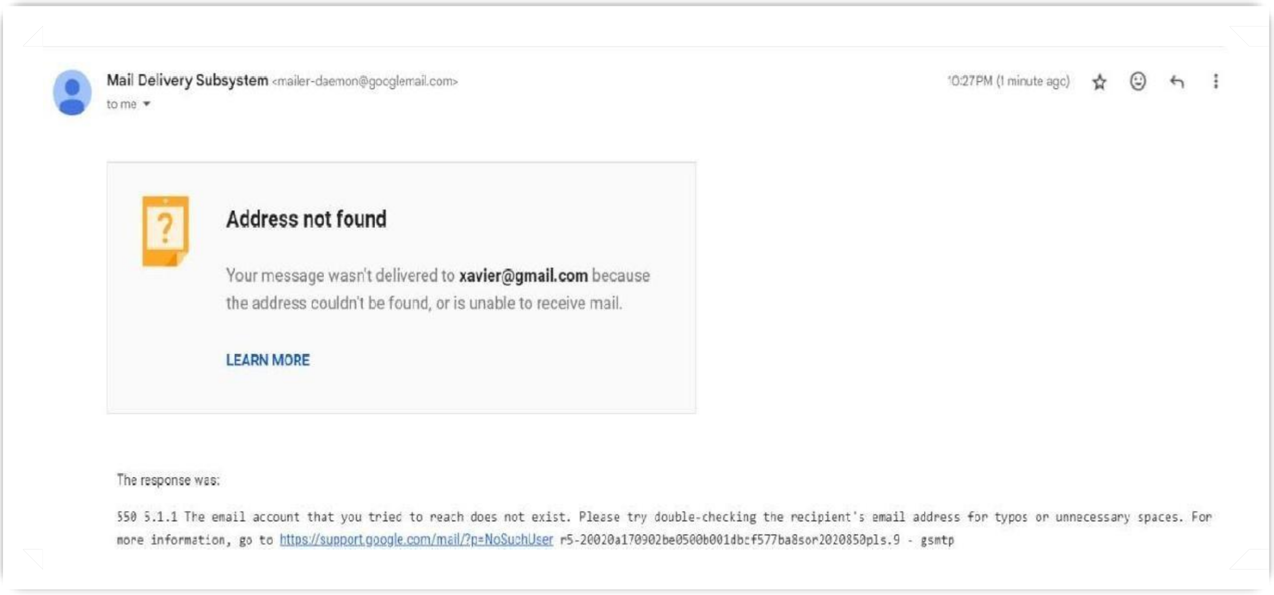
Table 4: Fourth test case for User Registration Page.

Result is shown below.



Testcase 4: Check registration with not existing mail.	Priority: High (Ctrl)
Test Unit: User Registration Page	
Test Description: To check the registration page with giving not existed email id.	
Test Data: Username: Likhitha Email-ID: savantisff@gmail.com Password: Likhitha@11	
Actions: Click on register.	Test Environment: Online Web App.
Expected Result: The email account that you tried to reach does not exist.	Output: The email account that you tried to reach does not exist.
Test Status: Pass	

Result is shown below.



Mail Delivery Subsystem <mailer-daemon@googlemail.com>
to me

10:27PM (1 minute ago)

Address not found

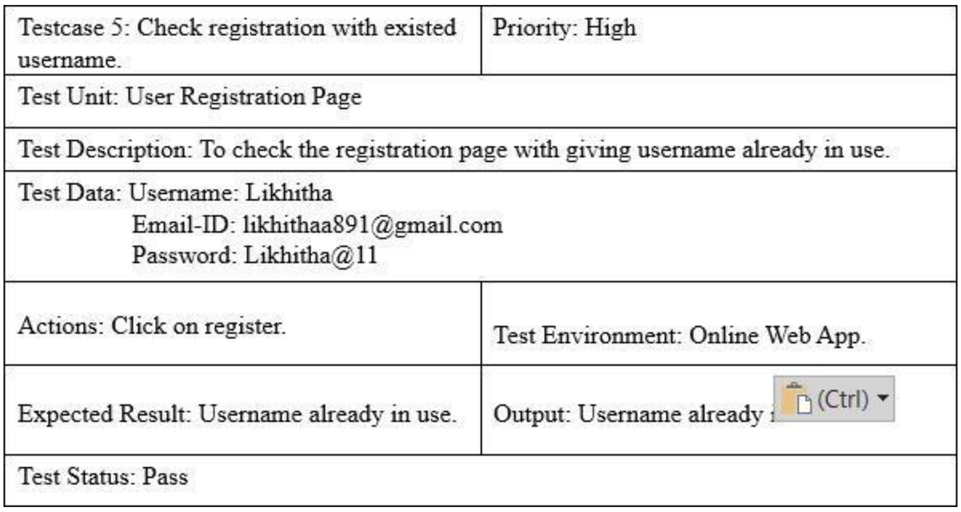
Your message wasn't delivered to **xavier@gmail.com** because the address couldn't be found, or is unable to receive mail.

LEARN MORE

The response was:

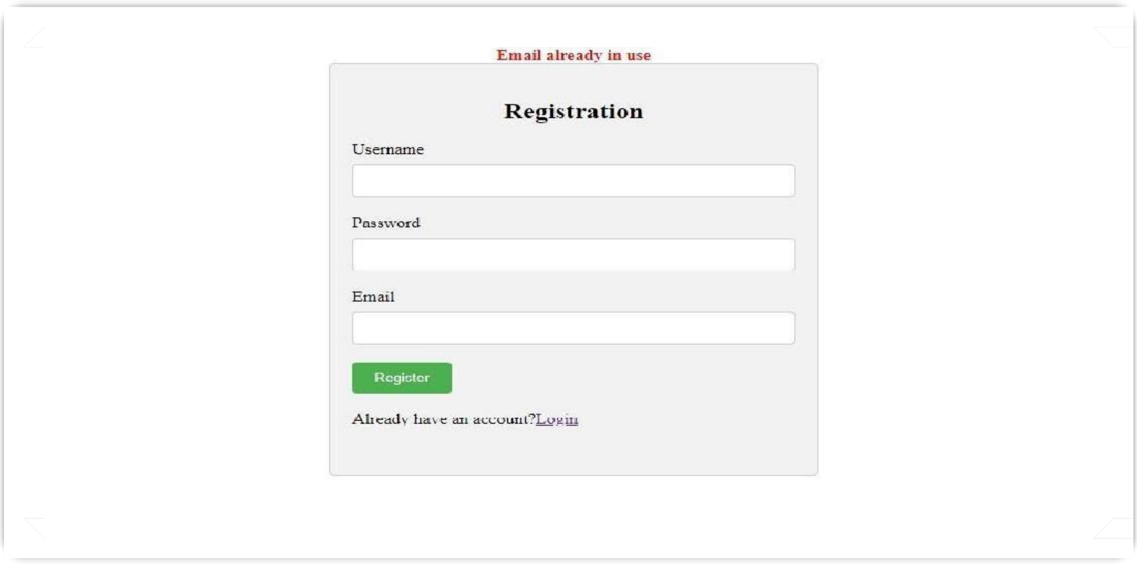
```
550 5.1.1 The email account that you tried to reach does not exist. Please try double-checking the recipient's email address for typos or unnecessary spaces. For more information, go to https://support.google.com/mail/?p=NoSuchUser r5-20020a170902be0500b001dbcf577ba8sor2020850pls.9 - gsmtp
```

Table 5: Fifth test case for User Registration Page.



Testcase 5: Check registration with existed username.	Priority: High
Test Unit: User Registration Page	
Test Description: To check the registration page with giving username already in use.	
Test Data: Username: Likhitha Email-ID: likhithaa891@gmail.com Password: Likhitha@11	
Actions: Click on register.	Test Environment: Online Web App.
Expected Result: Username already in use.	Output: Username already i
Test Status: Pass	

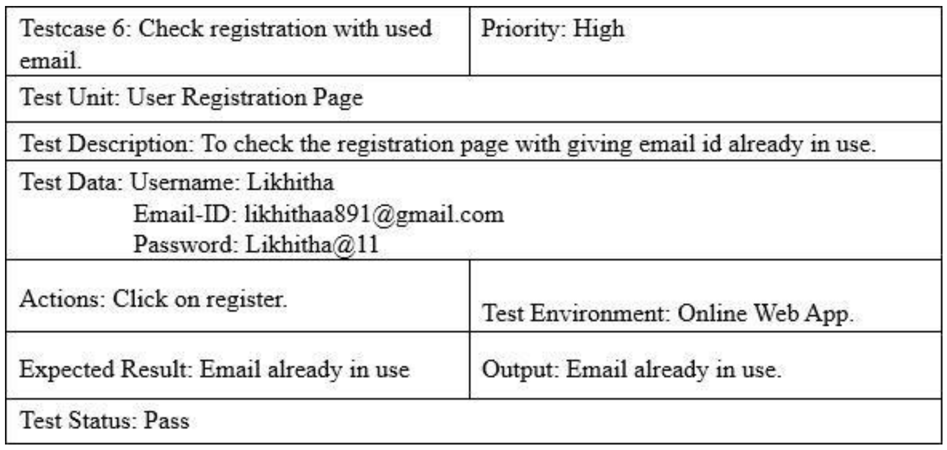
Result is shown below.



Email already in use

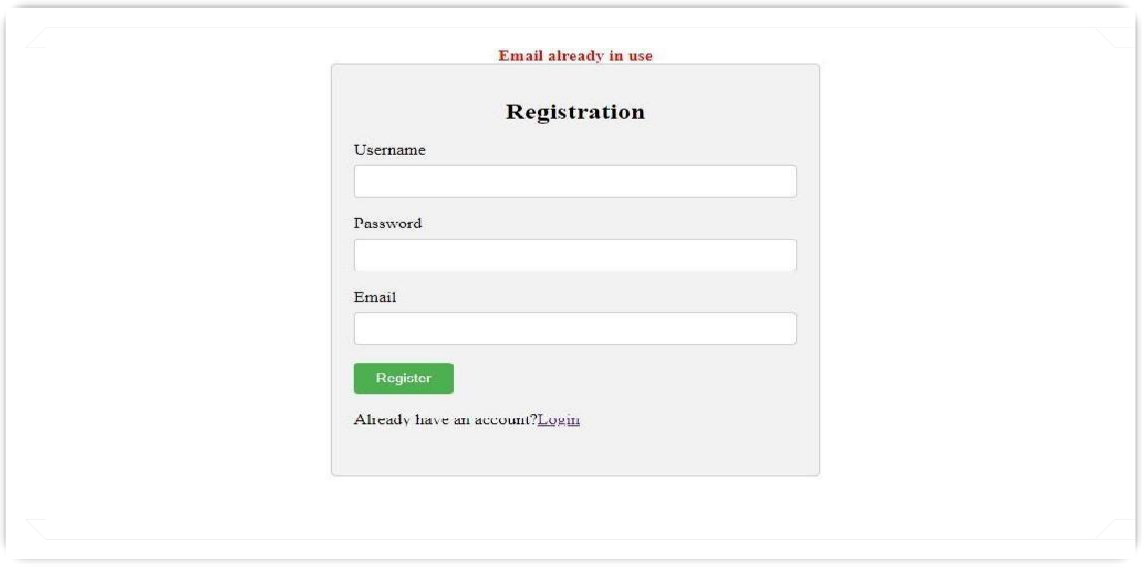
Registration

Table 6: Sixth test case for User Registration Page.



Testcase 6: Check registration with used email.	Priority: High
Test Unit: User Registration Page	
Test Description: To check the registration page with giving email id already in use.	
Test Data: Username: Likhitha Email-ID: likhithaa891@gmail.com Password: Likhitha@11	
Actions: Click on register.	Test Environment: Online Web App.
Expected Result: Email already in use	Output: Email already in use.
Test Status: Pass	

Result is shown below.



Email already in use

Testing Unit : User Login Page.

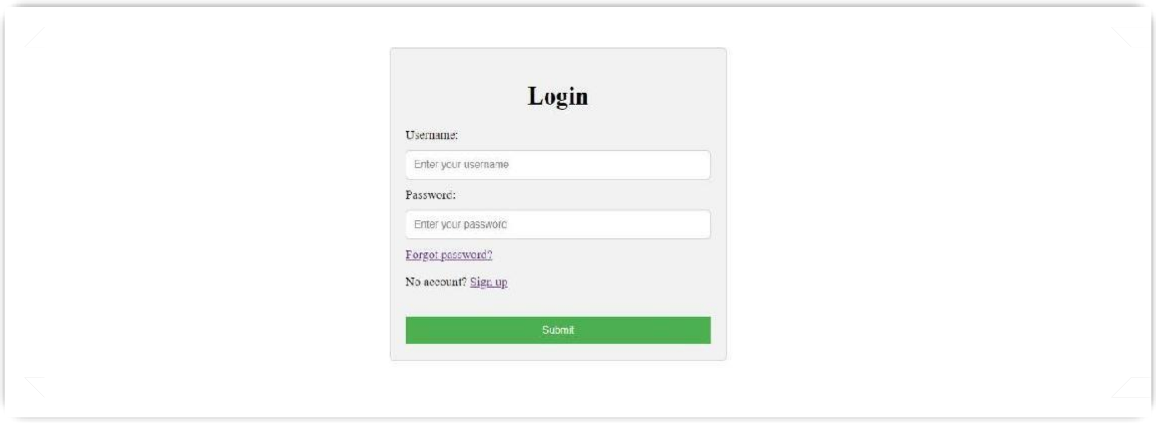
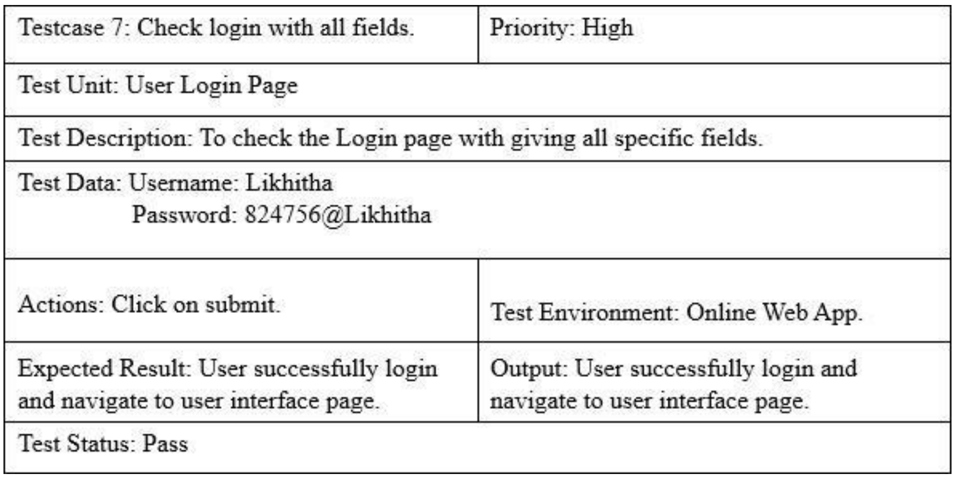
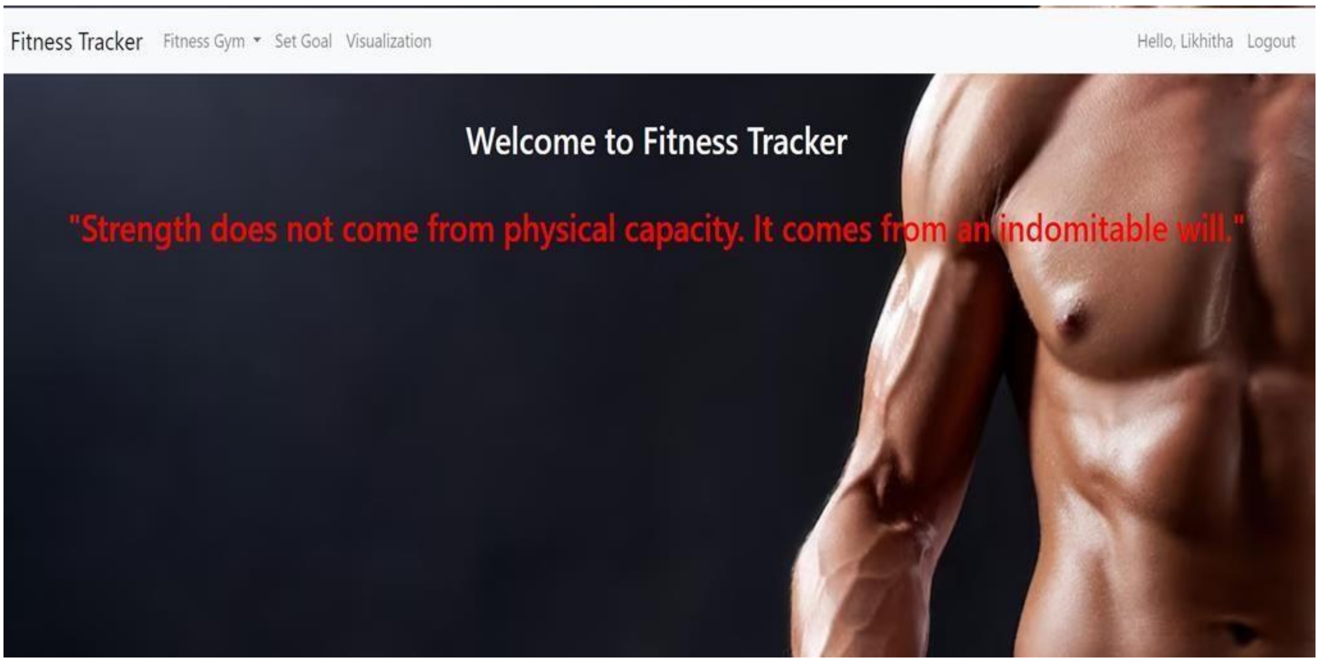


Table 7: First test case for User Login Page.



Testcase 7: Check login with all fields.	Priority: High
Test Unit: User Login Page	
Test Description: To check the Login page with giving all specific fields.	
Test Data: Username: Likhitha Password: 824756@Likhitha	
Actions: Click on submit.	Test Environment: Online Web App.
Expected Result: User successfully login and navigate to user interface page.	Output: User successfully login and navigate to user interface page.
Test Status: Pass	

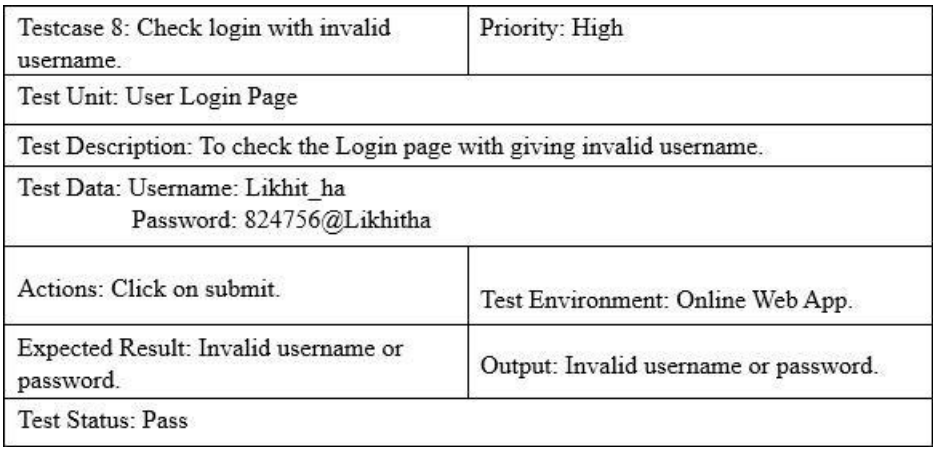
Result is shown below.



Welcome to Fitness Tracker

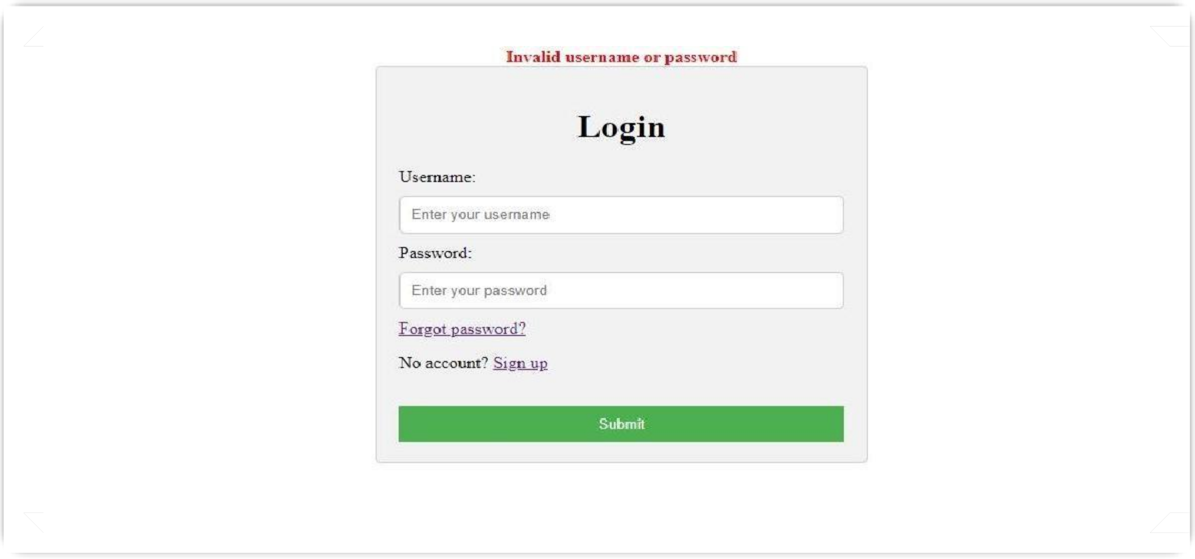
"Strength does not come from physical capacity. It comes from an indomitable will."

Table 8: Second test case for User Login Page.



Testcase 8: Check login with invalid username.	Priority: High
Test Unit: User Login Page	
Test Description: To check the Login page with giving invalid username.	
Test Data: Username: Likhit_ha Password: 824756@Likhitha	
Actions: Click on submit.	Test Environment: Online Web App.
Expected Result: Invalid username or password.	Output: Invalid username or password.
Test Status: Pass	

Result is shown below.



Invalid username or password

Login

Username:

Enter your username

Password:

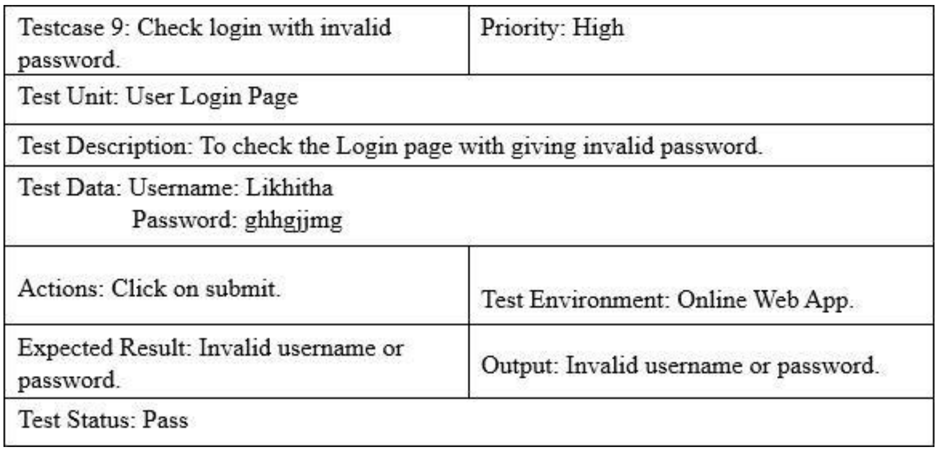
Enter your password

Forgot password?

No account? Sign up

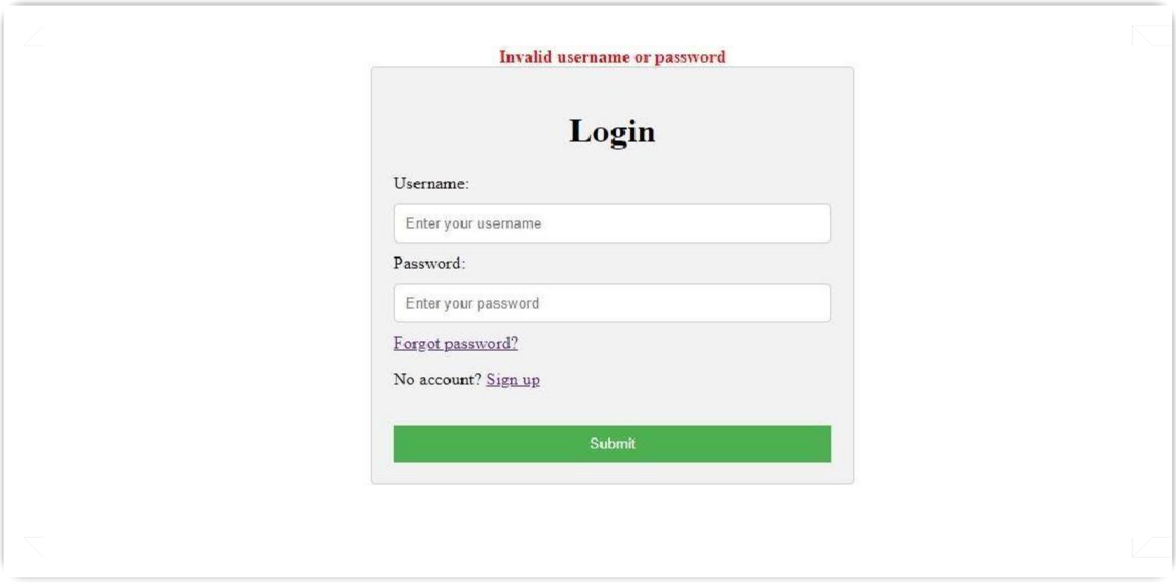
Submit

Table 9: Third test case for User Login Page.



Testcase 9: Check login with invalid password.	Priority: High
Test Unit: User Login Page	
Test Description: To check the Login page with giving invalid password.	
Test Data: Username: Likhitha Password: ghhgjjmg	
Actions: Click on submit.	Test Environment: Online Web App.
Expected Result: Invalid username or password.	Output: Invalid username or password.
Test Status: Pass	

Result is shown below.

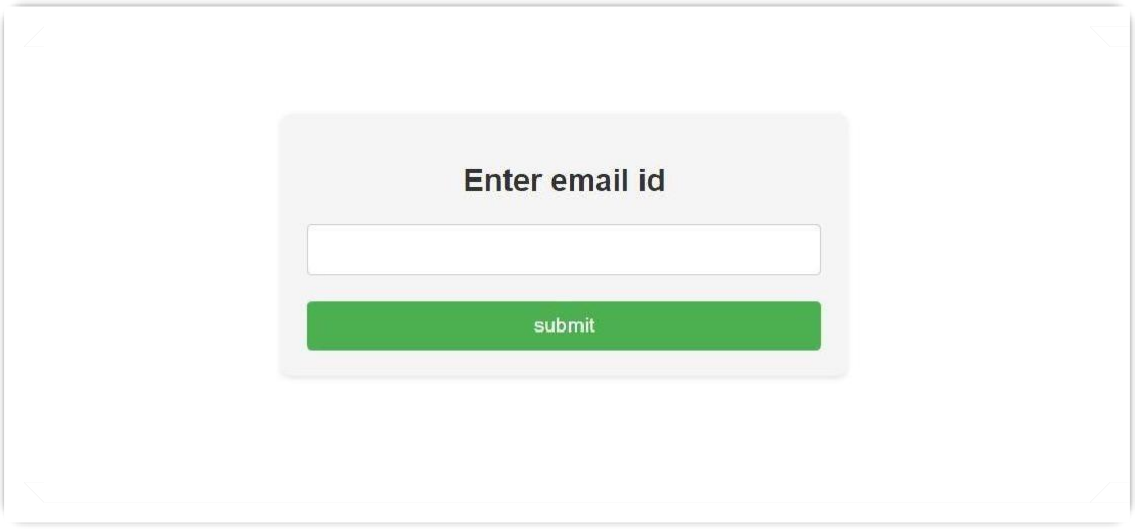


Invalid username or password

Table 10: Fourth test case for User Login Page.

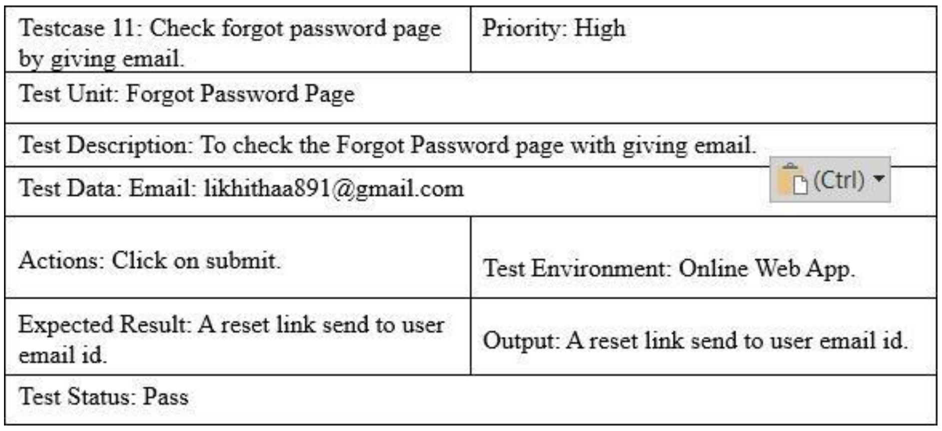
Testcase 10: Check login by clicking forgot password.	Priority: High
Test Unit: User Login Page	
Test Description: To check the Login page with clicking forgot password.	
Test Data: Click forgot password.	
Actions: Click on submit.	Test Environment: Online Web App.
Expected Result: Navigate to forgot password page.	Output: Navigate to forgot password page.
Test Status: Pass	

Result is shown below.



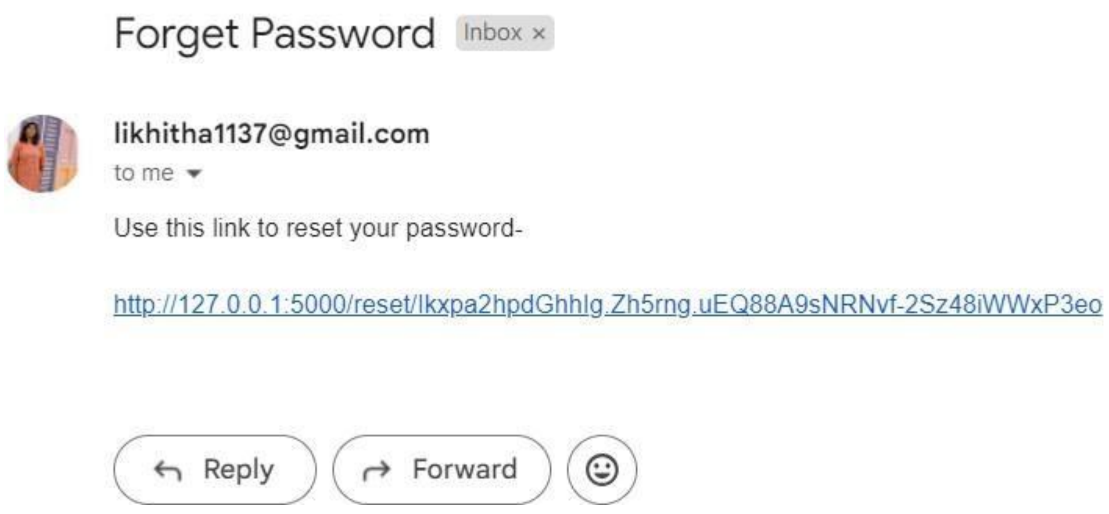
Testing Unit: Forgot Password Page.

Table 11: First test case for Forgot Password Page.



Testcase 11: Check forgot password page by giving email.	Priority: High
Test Unit: Forgot Password Page	
Test Description: To check the Forgot Password page with giving email.	
Test Data: Email: likhithaa891@gmail.com	
Actions: Click on submit.	Test Environment: Online Web App.
Expected Result: A reset link send to user email id.	Output: A reset link send to user email id.
Test Status: Pass	

Result is shown below.



Forget Password

likhitha1137@gmail.com

to me

Use this link to reset your password-

http://127.0.0.1:5000/reset/Ikxpa2hpdGhhIg.Zh5rng.uEQ88A9sNRNvf-2Sz48iWWxP3eo

Table 12: Second test case for Forgot Password Page.

Testcase 12: Checking reset link in email.	Priority: High
Test Unit: Forgot Password Page	
Test Description: To check the Forgot Password page by clicking reset link send to mail.	
Test Data: clicking reset password link sent to email.	
Actions: Click on reset link.	Test Environment: Online Web App.
Expected Result: Navigate to new password page.	Output: Navigate to new password page.
Test Status: Pass	

Result is shown below.

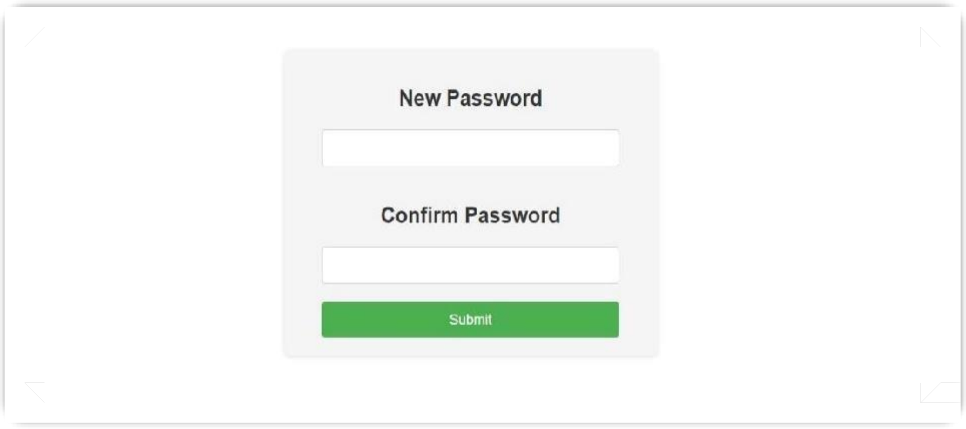


Table 13: Third test case for Forgot Password Page.

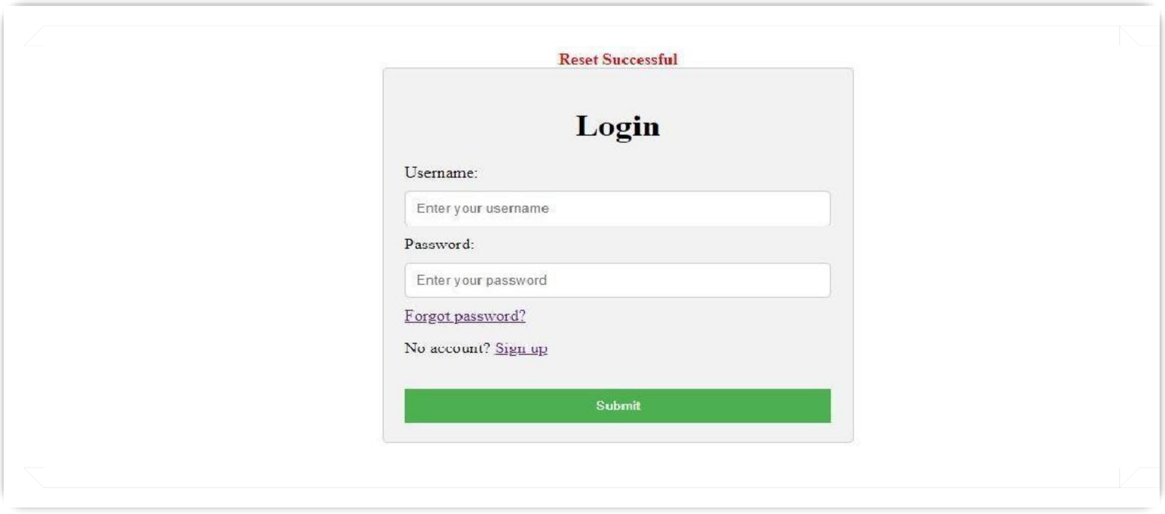
Testcase 13: Check forgot password page by giving invalid mail.	Priority: High
Test Unit: Forgot Password Page	
Test Description: To check the Forgot Password page by giving invalid email.	
Test Data: clicking reset password link sent to email.	
Actions: Click on submit.	Test Environment: Online Web App.
Expected Result: Invalid email id.	Output: Invalid email id.
Test Status: Pass	

Result is shown below.

Table 14: First test case for New Password Page.

Testcase 14: Check new password page by giving all fields.	Priority: High
Test Unit: New Password Page	
Test Description: To check the New Password page by all specific fields.	
Test Data: New password: Likhitha&123 Confirm password: Likhitha&123	
Actions: Click on submit.	Test Environment: Online Web App.
Expected Result: Reset Successful.	Output: Reset Successful.
Test Status: Pass	

Result is shown below.



Reset Successful

Login

Username:

Enter your username

Password:

Enter your password

Forgot password?

No account? Sign up

Submit

Table 15: Second test case for New Password Page.

Testcase 15: check new password page by giving different password.	Priority: High
Test Unit: New Password Page	
Test Description: To check the registration page with giving password without uppercase letter, a number and a special character.	
Test Data: New Password: nomore Confirm Password: nomore	
Actions: Click on submit	Test Environment: Online Web App.
Expected Result: Password must be 8 characters long and contain at least one uppercase letter, one number, and one special character message.	Output: Password must be 8 characters long and contain at least one uppercase letter, one number, and one special character message.
Test Status: Pass	

Result is shown below.

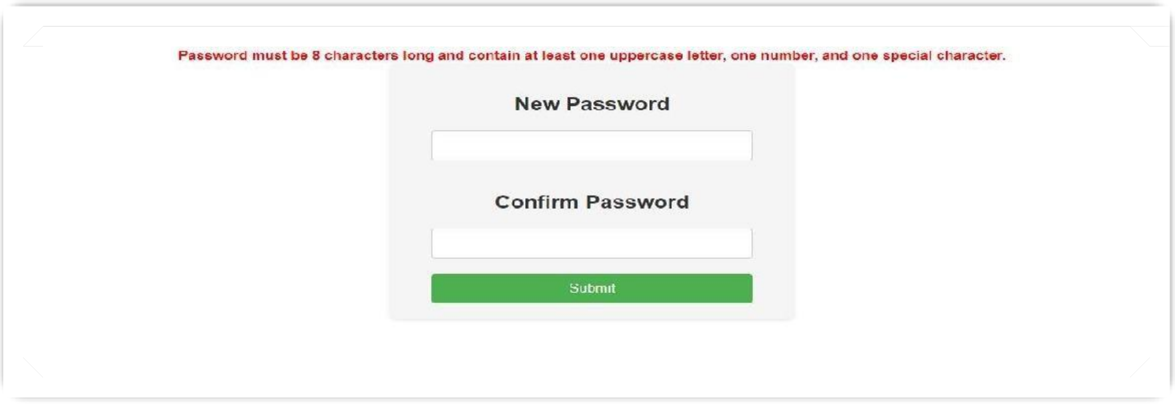
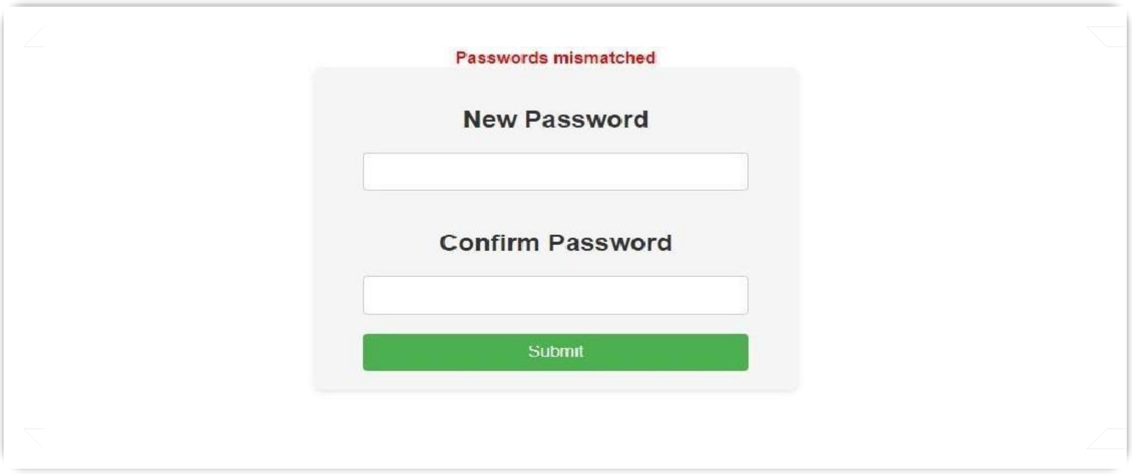

Password must be 8 characters long and contain at least one uppercase letter, one number, and one special character.

Table 16: Third test case for New Password Page.

Testcase 16: Giving new password and confirm password different.	Priority: High
Test Unit: New Password Page	
Test Description: To check the New Password page by giving new password and confirm password different.	
Test Data: New password: Likhitha&123 Confirm password: Likhitha#123	
Actions: Click on submit.	Test Environment: Online Web App.
Expected Result: Passwords mismatched.	Output: Passwords mismatched.
Test Status: Pass	

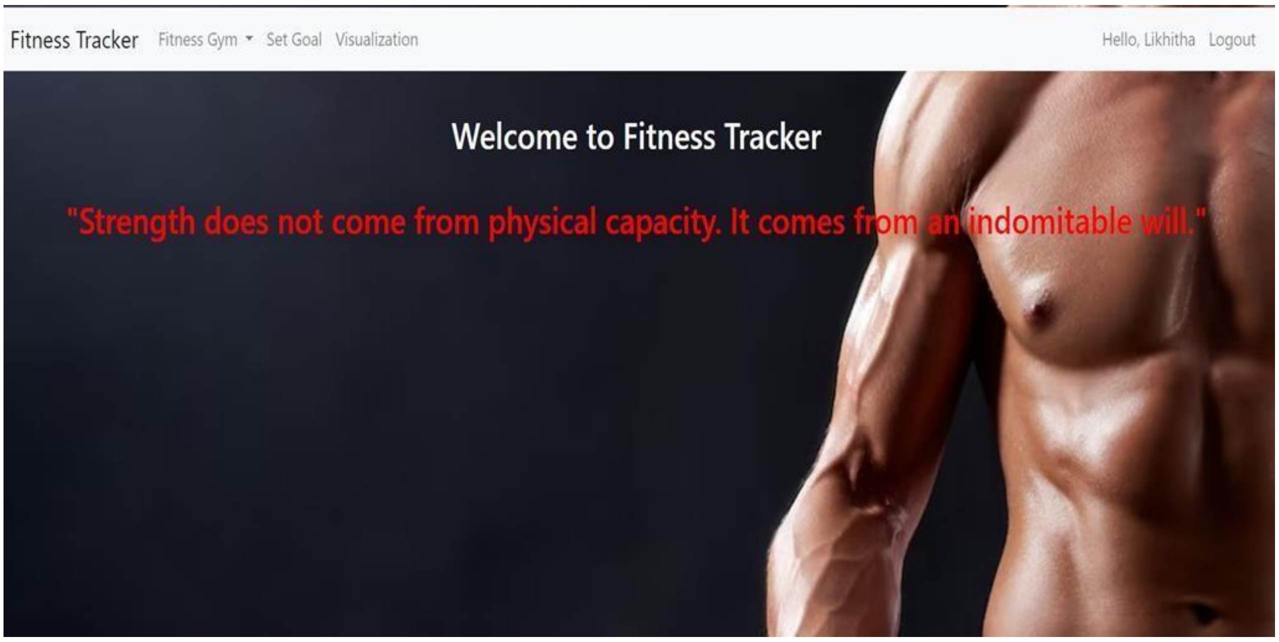
Result is shown below.


Passwords mismatched

New Password

Confirm Password

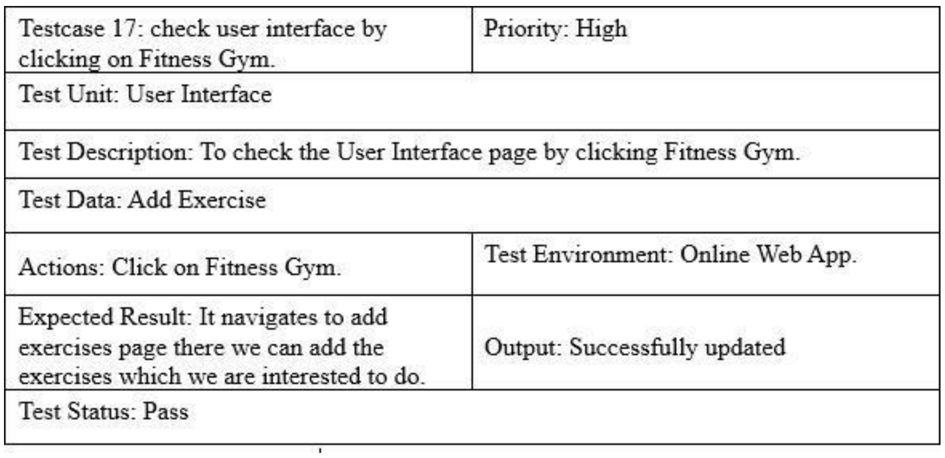
Testing unit: User Interface Page



Welcome to Fitness Tracker

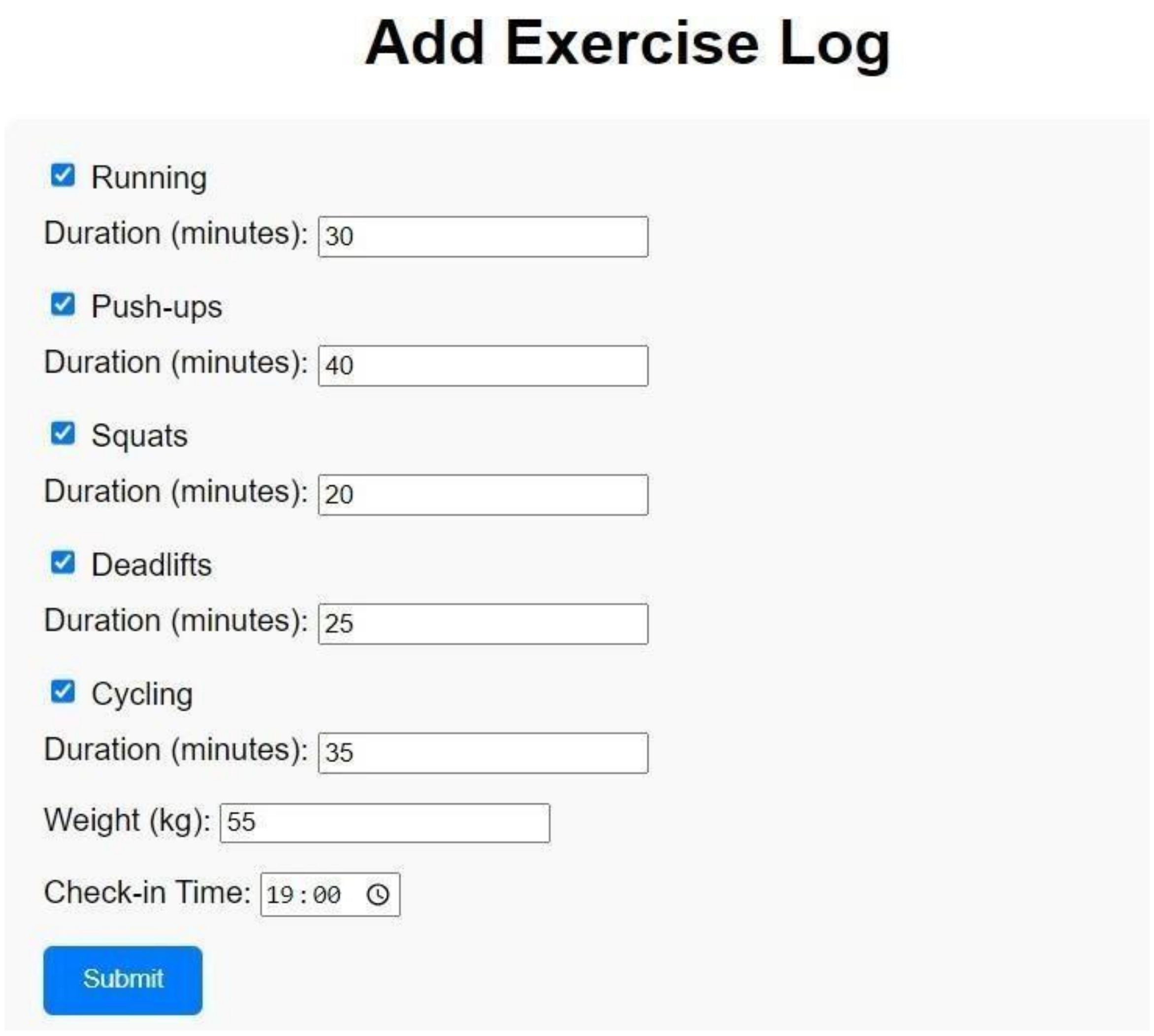
"Strength does not come from physical capacity. It comes from an indomitable will."

Table 17: First test case for User Interface Page.



Testcase 17: check user interface by clicking on Fitness Gym.	Priority: High
Test Unit: User Interface	
Test Description: To check the User Interface page by clicking Fitness Gym.	
Test Data: Add Exercise	
Actions: Click on Fitness Gym.	Test Environment: Online Web App.
Expected Result: It navigates to add exercises page there we can add the exercises which we are interested to do.	Output: Successfully updated
Test Status: Pass	

Result shown below



Add Exercise Log

☑ Running
Duration (minutes): 30

☑ Push-ups
Duration (minutes): 40

☑ Squats
Duration (minutes): 20

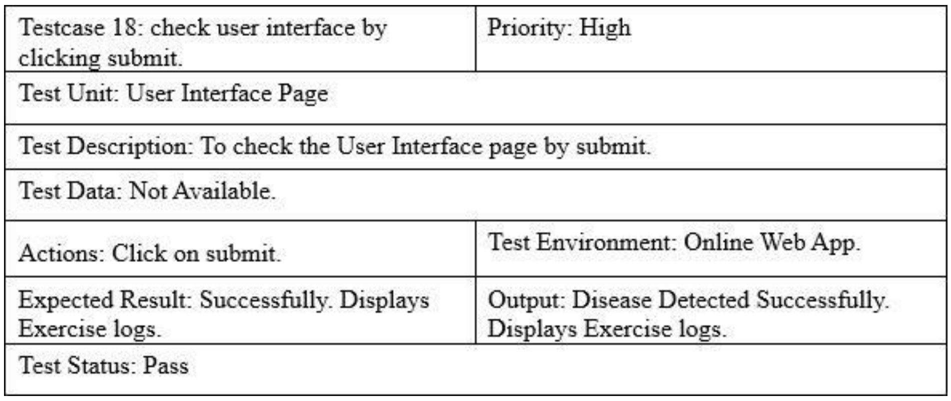
☑ Deadlifts
Duration (minutes): 25

☑ Cycling
Duration (minutes): 35

Weight (kg): 55

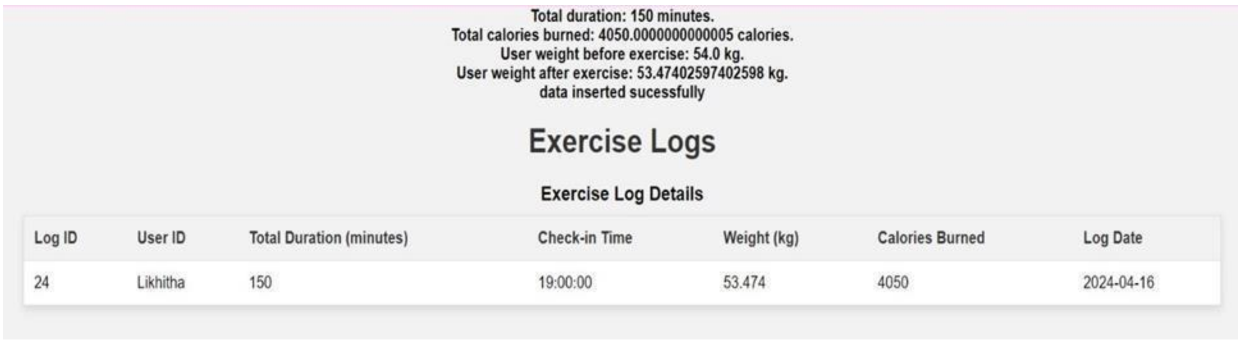
Check-in Time: 19:00

Table 18: Second test case for User Interface Page.



Testcase 18: check user interface by clicking submit.	Priority: High
Test Unit: User Interface Page	
Test Description: To check the User Interface page by submit.	
Test Data: Not Available.	
Actions: Click on submit.	Test Environment: Online Web App.
Expected Result: Successfully. Displays Exercise logs.	Output: Disease Detected Successfully. Displays Exercise logs.
Test Status: Pass	

Result is shown below.



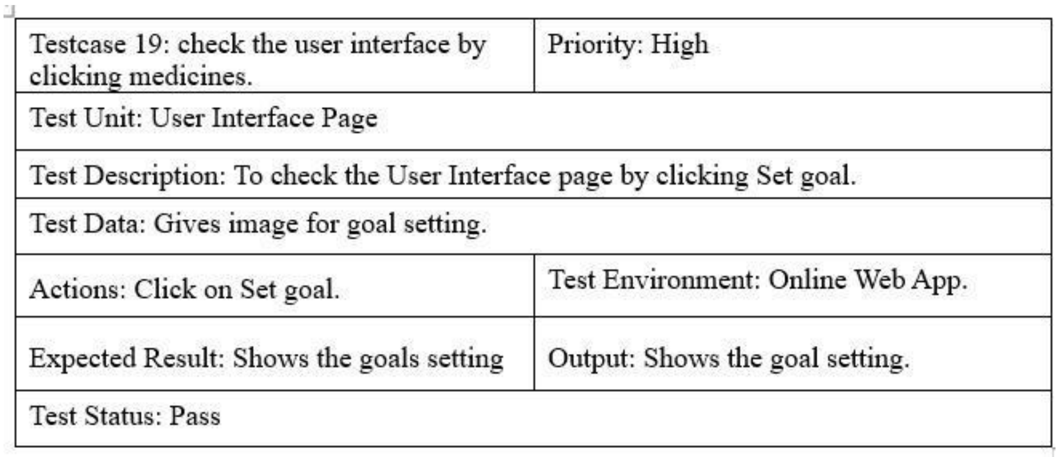
Total duration: 150 minutes.
Total calories burned: 4050.0000000000005 calories.
User weight before exercise: 54.0 kg.
User weight after exercise: 53.47402597402598 kg.
data inserted sucessfully

Exercise Logs

Exercise Log Details

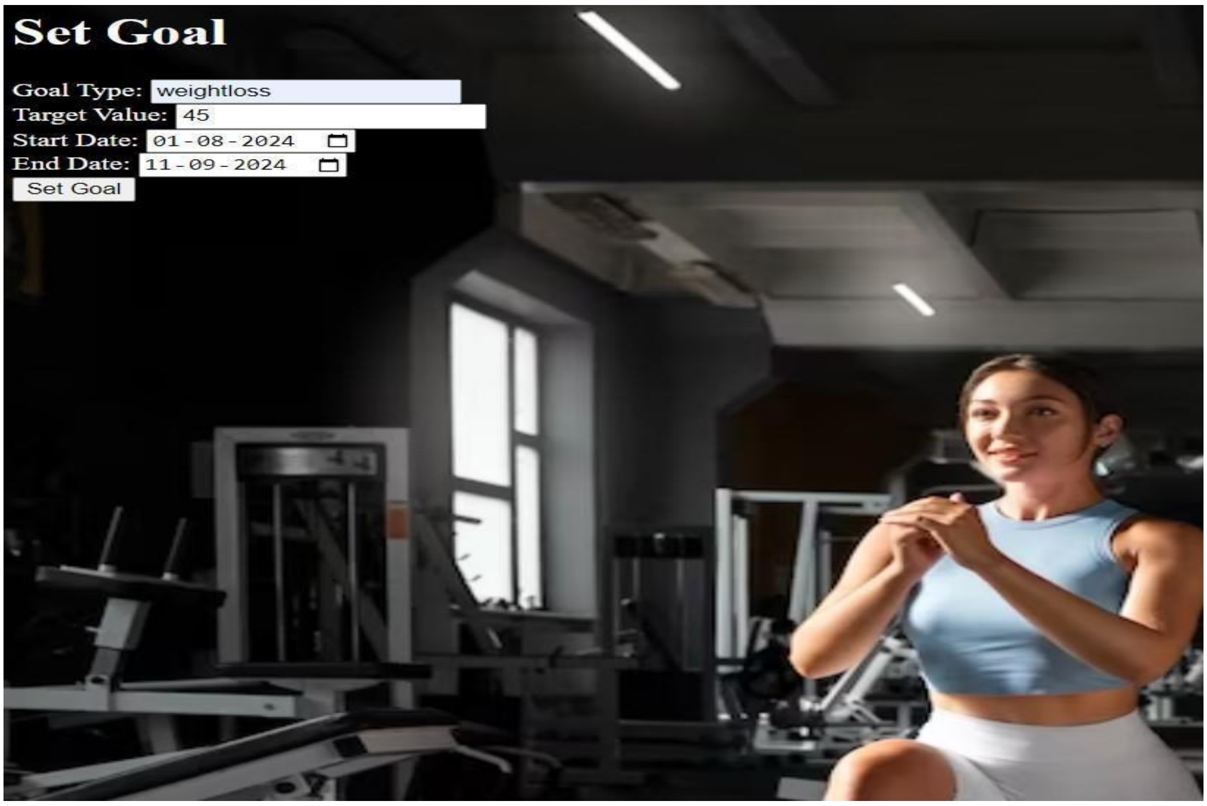
Log ID	User ID	Total Duration (minutes)	Check-in Time	Weight (kg)	Calories Burned	Log Date
24	Likhitha	150	19:00:00	53.474	4050	2024-04-16

Table 19: Third test case for User Interface Page.



Testcase 19: check the user interface by clicking medicines.	Priority: High
Test Unit: User Interface Page	
Test Description: To check the User Interface page by clicking Set goal.	
Test Data: Gives image for goal setting.	
Actions: Click on Set goal.	Test Environment: Online Web App.
Expected Result: Shows the goals setting	Output: Shows the goal setting.
Test Status: Pass	

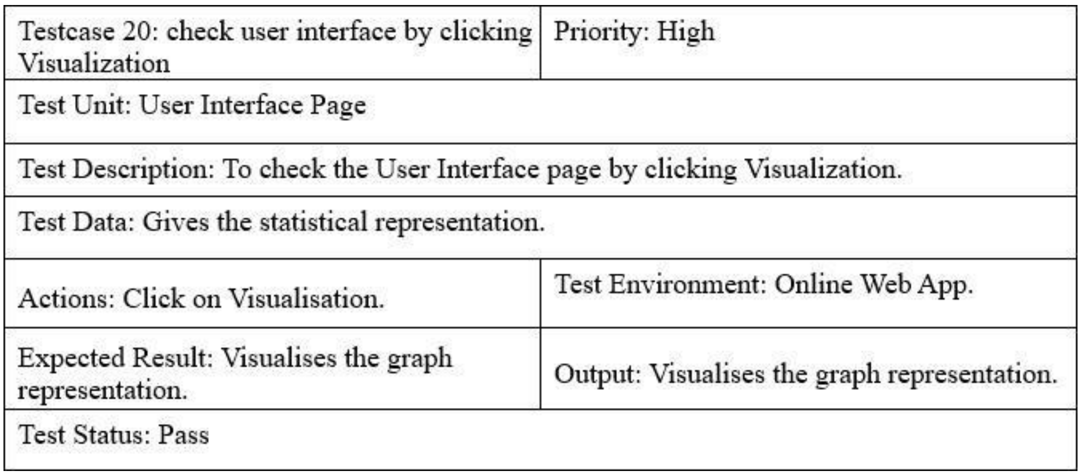
Result is shown below.



Set Goal

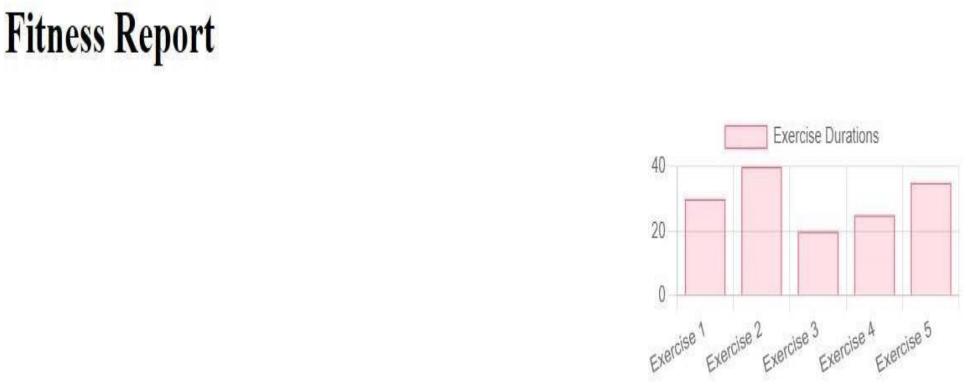
Goal Type: weightloss
Target Value: 45
Start Date: 01-08-2024
End Date: 11-09-2024

Table 20: Fourth test case for User Interface Page.



Testcase 20: check user interface by clicking Visualization	Priority: High
Test Unit: User Interface Page	
Test Description: To check the User Interface page by clicking Visualization.	
Test Data: Gives the statistical representation.	
Actions: Click on Visualisation.	Test Environment: Online Web App.
Expected Result: Visualises the graph representation.	Output: Visualises the graph representation.
Test Status: Pass	

Result is shown below.



Fitness Report

SOFTWARE ENVIRONMENT

10.1 Python Software Environment

If Python is used for the development of the Virtual Desktop for Voice Assistant system, the software platform would include:

- OS: Any OS that supports Python can be used to create the system, however a recent OS like Windows is preferred for the best performance.
- Python Version: As of September 2021, the most recent version of Python is 3, which may be used to construct the system.
- Python Library: The system can execute its necessary functionalities with the help of several Python modules and libraries. You can utilize libraries and modules like these:
    - A Python module called Speech Recognition makes it easy to use speech recognition APIs like Google Voice API, Microsoft speech API, and also CMU Sphinx.
    - The Python package pyttsx3 makes it simple to access several text-to-speech APIs, including the ones provided by Google, Microsoft, and talk.
    - Python module called "Wikipedia" that facilitates simple data retrieval from the online encyclopedia.
    - Selenium is a Python package that facilitates interaction with websites and the execution of web-based operations.
    - The smtplib Python package makes it simple to send emails over the Simple Mail Transfer Protocol (SMTP).

• An Integrated Development Environment (IDE): To build and test the system, you can use an IDE like Spyder, Visual Studio Code, or PyCharm.

• To review, a Python-based software environment for The Virtual Desktop's Voice Assistant would include an OS, Python 3, a number of Python modules and libraries, and an integrated development environment (IDE) for creating and testing the system.

10.2 VS Code Software Environment

Visual Studio Code (VS Code) is Microsoft's free and open-source code editor. Thanks to its customizable features, support for several programming languages and tools, and lightweight architecture, it is rapidly becoming one of the foremost popular code editors among developers. When creating software, VS Code has several advantages, including as:

- VS Code's user interface is designed to be intuitive, allowing developers to work swiftly and concentrate on creating code.

- Codes formatting, syntax highlights, and debugging tools are just a few examples of the many available extensions for Visual Studio Code on the extensive marketplace.

- Integrated Terminal: Developers may run scripts and commands right from the editor with Visual Studio Code's integrated terminal.

- Developers may handle version control and work with other team members with VS Code's built-in support for Git.

- VS Code's cross-platform compatibility allows developers to work seamlessly across several platforms, since it operates on Windows, MacOS, and Linux, among others.

- Robust Debugging assistance: Visual Studio Code simplifies the process of finding and fixing bugs in code by providing strong debugging assistance for a number of languages.

- Developers may save time and effort using VS Code's intelligent code recommendations and auto-completion features, which enhance code quality and efficiency.

- To summarize, Visual Studio Code is an adaptable code editor that offers a plethora of tools and features to help developers build software with ease and efficiency.

CONCLSION AND FUTURE ENHANCEMENT

CONCLUSION

To sum up, the fitness tracking software is a great resource for keeping tabs on all the many parts of a person's fitness journey. It has helped people achieve their fitness objectives by giving them a better picture of their activity, heart rate, and sleep habits.

Users have seen improvements in their exercise programs, diets, and overall quality of life as a result of using this software. It has become possible for them to see trends, pinpoint problem areas, and monitor their development over time. Because of this, they have been able to maintain their drive and dedication to their exercise program.

Having said that, the fitness monitoring software might yet use some improvements in the future. Incorporating more sophisticated and precise sensors, like GPS & heart rate trackers, might be one way to enhance the product and give consumers more precise and comprehensive data regarding their exercises. The user experience might be even better if ideas and recommendations were more tailored to each person's unique needs and interests.

An other way to boost engagement and motivation is to make the app more social by letting users compete with each other and their friends. Possible inclusions here include buddy-based virtual exercise, leaderboards, and challenges.

In order to provide users a complete picture of their health, the app might potentially look into forming collaborations with other wellness platforms in the future. This could include applications that track diet or wearable gadgets.

Users have found the fitness monitoring software to be an invaluable resource on their paths to better health. With ongoing updates and improvements, it can help users even more in reaching their fitness objectives.

FUTURE ENHANCEMENT

1. First and foremost, fitness trackers of the future will be able to integrate with other wearable devices, such as smartwatches and fitness bands. This would make it possible to monitor vital signs like heart rate, sleep duration, and calorie expenditure more precisely and in real time.
2. State-of-the-art machine learning algorithms: Making use of state-of-the-art machine learning techniques would allow the app to tailor suggestions and recommendations to each user by analyzing their data. To help users track their fitness progress and choose the best exercises for their needs, these algorithms can look for trends and patterns in their activity data.
3. Challenges and social integration: Users might interact and compete with their friends through the addition of elements that allow social integration, resulting in a fitness experience led by the community. Users may participate in fitness-related challenges to boost their motivation, build camaraderie, and hold themselves accountable.
4. Incorporating online training and personal coaches within the program would allow users to get individualized assistance and direction. Bringing the experience of a professional workout coach straight to the user's device, these virtual trainers may provide training regimens, show appropriate technique, and provide feedback on performance.
5. One may get a more comprehensive approach to wellness and health by integrating diet tracking with workout tracking. People may monitor what they eat every day, establish dietary objectives, and learn more about the nutritional content of the food they eat. Depending on the user's objectives in terms of physical fitness, the app may also provide healthy eating regimens.
6. Incorporating gamification aspects: The fitness monitoring experience might be greatly enhanced by incorporating gamification elements like virtual prizes, leveling up systems, accomplishment badges, and more. These game-like elements might make people more engaged and motivated to achieve their fitness objectives.
7. Workouts using augmented reality (AR): This software may place virtual trainers or workout spaces over the user's actual environment. By supplying the user with immersive graphics and instructions throughout exercise regimens, this would improve the user's workout experience.

8. Adding tools to monitor and control stress levels would be a great boon to consumers' efforts to keep their lives in check. In response to user inputs, the app may track stress levels and offer relaxation methods, meditation exercises, and other tools for stress management in an effort to make users feel better physically and mentally.
9. Integration of voice-controlled capabilities would allow users to engage with the program hands-free; this brings us to our ninth point, voice command and natural language processing. With the use of natural language processing, users may more easily access information, begin exercises, or inquire about their fitness status just by speaking their requests.
10. Integrating health data: The fitness tracking app might get a more complete picture of the user's health if it worked with medical information platforms and electronic medical records. In order to make tailored suggestions and guarantee user safety while exercising, this information's integration can incorporate medical histories, allergies, and other pertinent health information.

BIBILOGRAPHY

BIBILOGRAPHY

1. Cadmus-Bertram LA, Marcus BH, Patterson RE, Parker BA, Moreno MA. Randomized Trial of a Fitbit-Based Physical Activity Intervention for Women. Am J Prev Med. 2015 Oct;49(4):614-8. doi: 10.1016/j.amepre.2015.01.020. PMID: 25863291.

2. Fan J, Congdon CB. Personal tracking of physical activity with wearable devices: a systematic review of current literature. Health informatics journal. 2017 Dec;23(4): 323-332. doi: 10.1177/1460458216641366. PMID: 27465218.

3. Jagphani T, Kumaravel S, Srinivasan K. Enhancement in healthcare using wearable technology. International Journal of Applied Engineering Research. 2015;10(24):44903-44907.

4. Patel MS, Asch DA, Volpp KG. Wearable devices as facilitators, not drivers, of health behavior change. JAMA. 2015 Feb 3;313(5):459-60. doi: 10.1001/jama.2014.14781. PMID: 25647203.

5. Shih PC, Han K, Poole ES, Rosson MB, Carroll JM. Use and adoption challenges of wearable activity trackers. iConference 2015 Proceedings; 2015: 618-624.

6. Silva BM, Rodrigues JJ, de la Torre Díez I, López-Coronado M, Saleem K. Mobile-health: A review of current state in 2015. J Biom Biostat. 2016;7(1):311. doi:10.4172/2155-6180.1000311

7. Ryan ML, Shochet R, Hay Thornthwaite JA, Lasko TA, Rothman DJ. Implementation of a wearable fitness tracking device and its effect on physical activity. JPHMP. 2017;23(4):339-348. doi:10.18553/jmcp.2017.17046

8. Wang JB, Cadmus-Bertram LA, Natarajan L, White MM, Madanat H, Nichols JF, et al. Wearable sensor/device (Fitbit One) and SMS text-messaging prompts to increase physical activity in overweight and obese adults: A randomized controlled trial. Telemed J E Health. 2015;21(10):782-792. doi:10.1089/tmj.2014.0196

9. Welk GJ. Physical activity assessments for health-related research. Human Kinetics; 2002.

10. Zhang Z, Serdar B, Ocak H, Same M, Cheung YK, Bargi R, et al. A smartphone-based

physical activity coaching intervention for individuals with Type 2 diabetes: Feasibility randomized controlled trial. JMIR mHealth uHealth. 2018;6(1):e29. doi:10.2196/mhealth.7461

11. Jette, M., & Sidney, K. (2018). Mobile Health Apps for Exercise and Physical Activity Promotion: Review and Evaluation of the App Marketplace. Journal of Medical Internet Research, 20(6), e162. doi: 10.2196/jmir.9255

12. Ashari, S. H., & Kurniawan, D. H. (2017). A Review of Mobile Health Apps for Physical Activity: Role of Gamification and Social Networking Features. Journal of Telemedicine and Telecare, 23(8), 701-707. doi: 10.1177/1357633x16674069

13. Liu, Y., et al. (2019). Wearable and Mobile Sensing Deep Learning Models for Context-Aware Human Activity Recognition: A Comprehensive Review. Neurocomputing, 365, 4-22. doi: 10.1016/j.neucom.2019.08.112

14. Ma, Y., et al. (2020). Gamifying Physical Activity: A Systematic Review of Gamification Techniques and Their Effects on Physical Activity Behaviors. Psychology of Sport and Exercise, 50, 101744. doi: 10.1016/j.psychsport.2020.101744

15. Moher, D., et al. (2009). Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement. PLOS Medicine, 6(7), e1000097. doi: 10.1371/journal.pmed.1000097

16. Paechter, C., et al. (2018). Gamified Features in Personal Informatics Applications: A Survey and Case Study. Journal of Biomedical Informatics, 78, 114-122. doi: 10.1016/j.jbi.2017.12.009

7. Stuckey, M. I., & Shapiro, S.L. (2009). Rethinking the Mobile Health Revolution: Achieving Minimalism and Glamour in Mobile Health Applications. Journal of the American Medical Informatics Association, 16(4), 607-613. doi: 10.1197/jamia.m3059

8. Sullivan, A. N., & Lachman, M. E. (2017). Behavior Change with Fitness Technology in Sedentary Adults: A Review of the Evidence for Increasing Physical Activity. Frontiers in Public Health, 4, 289. doi: 10.3389/fpubh.2016.00289

9. Wang, J., et al. (2019). A Comprehensive Framework for Personalized Health Informatics Applications. Journal of Biomedical Informatics, 97, 103253. doi: 10.1016/j.jbi.2019.103253

10. World Health Organization. (2021). mHealth: New Horizons for Health through Mobile Technologies. Retrieved from https://www.who.int/goe/publications/goe_mhealth_web.pdf

Final_document[1].docx

by Raghupal Reddy

Submission date: 17-Apr-2024 05:47PM (UTC-0700)
Submission ID: 2351925838
File name: Final_document_1_.docx (4.22M)
Word count: 14808
Character count: 88556

Final_document[1].docx

ORIGINALITY REPORT

12%	9%	5%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	lbrce.ac.in Internet Source	4%
2	Submitted to Gujarat Technological University Student Paper	1%
3	Submitted to Jawaharlal Nehru Technological University Kakinada Student Paper	1%
4	www.coursehero.com Internet Source	<1%
5	sist.sathyabama.ac.in Internet Source	<1%
6	sites.google.com Internet Source	<1%
7	anyflip.com Internet Source	<1%
8	Submitted to Southern New Hampshire University - Distance Education Student Paper	<1%
9	Submitted to Engineers Australia	

	Student Paper	<1%
10	vdocuments.net Internet Source	<1%
11	cse.rvrjcce.ac.in Internet Source	<1%
12	en.wikipedia.org Internet Source	<1%
13	Submitted to British Institute of Technology and E-commerce Student Paper	<1%
14	kipdf.com Internet Source	<1%
15	Submitted to VIT University Student Paper	<1%
16	www.studymode.com Internet Source	<1%
17	Submitted to University of Wolverhampton Student Paper	<1%
18	Submitted to Gitam University Student Paper	<1%
19	umbertotassoni.biz Internet Source	<1%
20	vgu.ac.in Internet Source	

<1%

21 Submitted to Scarsdale High School
Student Paper <1%

22 mafiadoc.com
Internet Source <1%

23 smartech.gatech.edu
Internet Source <1%

24 www.iare.ac.in
Internet Source <1%

25 Abubakar Aliyu, Emeka Ogbuju, Taiwo Kolajo, Ihinkalu Olalekan, Francisca Oladipo. "Government regulatory policies on telehealth data protection using artificial intelligence and blockchain technology", Institution of Engineering and Technology (IET), 2023
Publication <1%

26 Advances in Intelligent Systems and Computing, 2016.
Publication <1%

27 Submitted to Indian Institute Of Management, Jammu
Student Paper <1%

28 helda.helsinki.fi
Internet Source <1%

29	mhealth.jmir.org Internet Source	<1%
30	sportdocbox.com Internet Source	<1%
31	www.ijarnd.com Internet Source	<1%
32	Yining Zhu, Yuhan Zhao, Ying Wu. "Effectiveness of mobile health applications on clinical outcomes and health behaviors in patients with coronary heart disease: A systematic review and meta-analysis", International Journal of Nursing Sciences, 2024 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Final_document[1].docx

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66

PAGE 67

PAGE 68

PAGE 69

PAGE 70

PAGE 71

PAGE 72

PAGE 73

PAGE 74

PAGE 75

PAGE 76

PAGE 77

PAGE 78

PAGE 79

PAGE 80

PAGE 81

PAGE 82

PAGE 83

PAGE 84

PAGE 85

PAGE 86

PAGE 87

PAGE 88

PAGE 89

PAGE 90

PAGE 91

PAGE 92

PAGE 93

PAGE 94

PAGE 95

PAGE 96

PAGE 97

PAGE 98

PAGE 99

PAGE 100

PAGE 101

PAGE 102

PAGE 103

PAGE 104

PAGE 105

PAGE 106

PAGE 107

PAGE 108

PAGE 109

PAGE 110

PAGE 111

PAGE 112

PAGE 113

PAGE 114

PAGE 115

PAGE 116

PAGE 117

PAGE 118

PAGE 119

PAGE 120

PAGE 121