Project Report

On

“HABIT TRACKING WEBAPP”

Submitted By

“Potgante Ankush Baburao”

MASTER OF SCIENCE IN COMPUTER SCIENCE

School of Computational Science

SWAMI RAMANAND TEERTH MARATHWADA UNIVERSITY

NANDED (M. S.) 431606

Year 2021-2022

Project Report

On

HABIT TRACKING WEBAPP

Submitted By

Potgante Ankush Baburao

[VZ90372]

Guided By

Prof. M.D. Wangikar

In partial fulfilment for the award of

MASTER OF SCIENCE IN COMPUTER SCIENCE

School of Computational Science

SWAMI RAMANAND TEERTH MARATHWADA UNIVERSITY

NANDED (M. S.) 431606

Year 2021-22

CERTIFICATE

This is to certify that, the project **“HABIT TRACKING WEBAPP”** submitted by

“POTGANTE ANKUSH BABURAO”

Is a bonafide work completed under my supervision and guidance in partial fulfilment for award of Master of Science in Computer Science Degree of Swami Ramanand Teerth Marathwada University, Nanded.

Place : Nanded

Date :

M.D.Wangikar
Guide

Dr. S.D. Khamitkar
Director

INDEX

1. INTRODUCTION

1.1 Introduction

What Is a Track me? It is a habit tracker, it is exactly what the name suggests: it's a way to help you track how well you're sticking with daily, weekly, or monthly habits. A good habit tracker can come in many forms, from a sophisticated app to a sheet of paper and a pen, here I have created a app "HABIT TRACKING WEBAPP".

If you want to stick with a habit for good, one simple and effective thing you can do is use HABIT TRACKING WEBAPP.

Elite performers will often measure, quantify, and track their progress in various ways. Each little measurement provides feedback. It offers a signal of whether they are making progress or need to change course.

Track me habit tracker is a simple way to measure whether you did a habit.

No matter what work you do, the key point is your habit tracker provides immediate evidence that you completed your habit. It's a signal that you are making progress. Of course, that's not all it does…

Habit tracking habit tracker is powerful for three reasons.

1. It creates a visual cue that can remind you to act.
2. It is motivating to see the progress you are making. You don't want to break your streak.
3. It feels satisfying to record your success in the moment.

HABIT TRACKING WEBAPP have a general format for tracker and will accommodate most of the tracker types and customer can view the necessary insights about the tracked data.

This project can be used by anyone who want to track some data about himself/herself and get some useful insights about the own tracked data.

1.2 Necessity

Habit tracking naturally builds a series of visual cues. When you look at the calendar and see your streak, you'll be reminded to act again.

Research has shown that people who track their progress on goals like losing weight, quitting smoking, and lowering blood pressure are all more likely to improve than those who don't. One study of more than sixteen hundred people found that those who kept a daily food log lost twice as much weight as those who did not. A habit tracker is a simple way to log your behaviour, and the mere act of tracking a behaviour can spark the urge to change it.

Track me habit tracker also keeps you honest. Most of us think we act better than we do. Measurement offers one way to overcome our blindness to our own behaviour and notice what's really going on each day. When the evidence is right in front of you, you're less likely to lie to yourself.

1.3 Existing System and Need for System

Remember, the best method does not have to be the one scoring high on all criteria, but the one that scores your own preferred level. Now, let's take a look at the few most popular habit tracking methods.

Bullet Journal:

Bullet Journal, at its core, is using a notebook to build a system of organizing our life. You can use it as a planner and habit tracker. Instead of storing all the tasks and habits you need to do in your mind and later forget about it, or writing down appointment schedules on random sticky notes and can't find them later, you have a system to plan ahead, remind you of what needs to be done.

Printed Template or Notebook Habit Tracker:

If you want to prevent yourself from forgetting to log in your habit daily, you can try printing out templates to a piece of paper, then stick them to somewhere obvious in your home or your working space. The door to your room, the refrigerator or on your desk,... anywhere you have to look at everyday.

You can easily find a lot of free printable templates online, all you have to do is pick the one that appeals most to you.

Excel Template or Google Sheet Template:

The key difference that sets spreadsheet from analog habit tracker is that it can generate statistics. Streak is not the single performance measurement anymore. Many other interesting statistics like completion rate towards goals, completion times counting, or comparison between different habits, can be automatically calculated.

Need for system

Bullet Journal takes quite a lot of time and effort to create the initial tracking template

Users may get lost in the artistic aspects, spending most of their time trying to make their tracker more beautiful, yet little time on working on their habits

Printed Template or Notebook Habit Tracker is not as flexible as Bullet Journal, since you have only a fixed template to fill in. Sometimes you want to add something, say, a few notes or changes, but you can't due to the limited space given.

If you use a printed paper, you might need more than one version: suppose you have only one at your home, then you have to wait till you get home to log in your progress.

If you want more support and automation, trying out habit tracking website is a must. Basically, everything a spreadsheet habit tracker can do, a habit tracking web app can do better. And one unique characteristic of this method out of the four mentioned, is that it masters the art of simplicity.

1.4 Scope of Work

This project will have a general format for tracker and will accommodate most of the tracker types and customer can view the necessary insights about the tracked data.

This project can be used by anyone who want to track some data about himself/herself and get some useful insights about the own tracked data.

1.5 Objectives

Progress display is optimized, every piece of statistics is carefully put in, so that users can navigate through all available data effortlessly, leaving behind the struggle of navigating on the complicated, filled up spreadsheet screen.

It has a high level of Automation. Almost no manual operation is required. Sometimes, with Siri shortcuts apps like HABIT TRACKING WEBAPP, you don’t even have to touch your smartphone to check off a habit.

It’s highly convenient - Now your habit tracker is in your phone, you can take it everywhere without realizing it’s even there. For HABIT TRACKING WEBAPP users specifically, habit data is synced on multiple devices (we are on iOS, Android, Mac and web), so their habit tracker is always seconds away.

You will never forget to track your habit. Every habit tracker app has a system of notifications that remind users to do the tracking. Through this, you are reminded to perform your habit and therefore, find sticking to it more easily.

2. ANALYSIS

3. PROPOSED SYSTEM

3.1 Proposed system :

In our proposed system we have the provision for adding the work or habit of the people by themselves. They can add the task daily and track their daily habit. Another advantage of the system is that it is very easy to edit the details of the habit and delete a habit when it found unnecessary. The records of the tracks are added in the database and so user can also view the data whenever they want.

Our proposed system has several advantages

- User friendly interface
- Fast access to database
- Less error
- More Storage Capacity
- Search facility
- Track records
- User-friendly environment

All the manual difficulties in managing the daily habit record details in a notebook or sticky notes have been rectified by implementing computerization.

3.3 SYSTEM REQUIREMENTS

SOFTWARE SPECIFICATION
 WINDOWS 10
 PYCHARM
 PYTHON
 SQLITE3

HARDWARE SPECIFICATION
 INTEL® CORETM i5, 3.20GHz
 4 GB RAM
 256 GB STORAGE SPACE
 LAN CARD

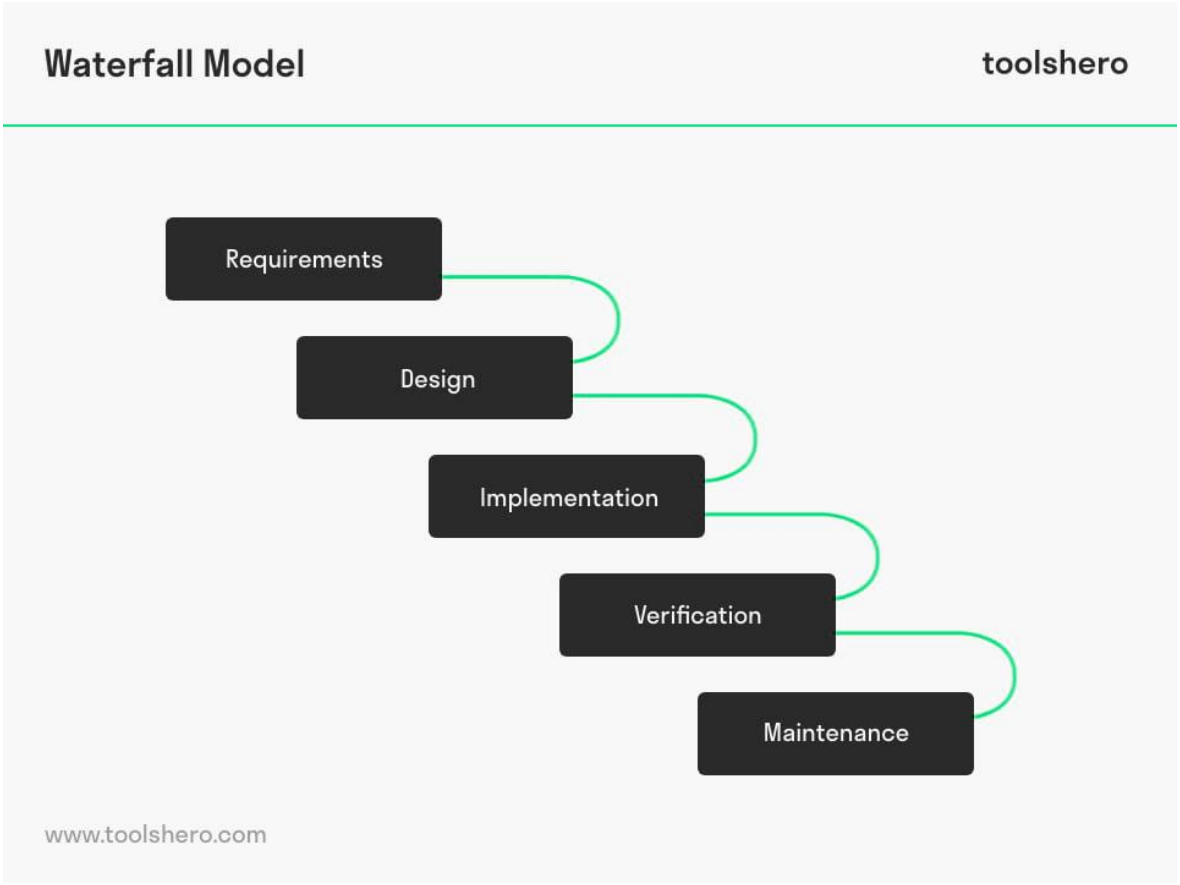
4. **SYSTEM DEVELOPMENT.**

4.1 Which SDLC Model is used?

WATERFALL MODEL

When applied in software development, the model is an iterative and flexible approach where progress, like the water of a waterfall, is largely in a single downward flow through the stages. The phases are: conception, initiation, analysis, design, construction, testing, implementation and maintenance.

The waterfall method is the first process model (SDLC) that was introduced. Originally, the waterfall method started in the construction industry.
Highly structured and physical environments meant that design changes during construction resulted in sky-high costs. Subsequently, when the demand for software increased, there was no recognized alternative to managing these projects.



1. System and software requirements

The first phase involves understanding what exactly needs to be developed, what the purpose is, the function, etc. It is in this phase that the requirements, deadlines, guidelines and other matters are established. The result is mostly a document in which the requirements are listed (Requirements Understanding Document).

2. Analysis

The information gathered in the first phase is used here to generate product models and logic. This is very important for the management of production. A feasibility test is also done at the end of this phase. This applies to the financial and technical resources.

3. Design

The third phase is the design phase. During this phase, the specifications from the first phase are studied and a system design is prepared. The system design helps with establishing the system- and hardware requirements.

Thereafter it helps with the definition of the general system architecture. This phase must be completed before going ahead with the coding phase. The code that will be developed in the next phase is in fact being prepared now. The result of this phase is a High-Level Design Document (HLD), or a Low-Level Design Document (LLD).

4. Coding

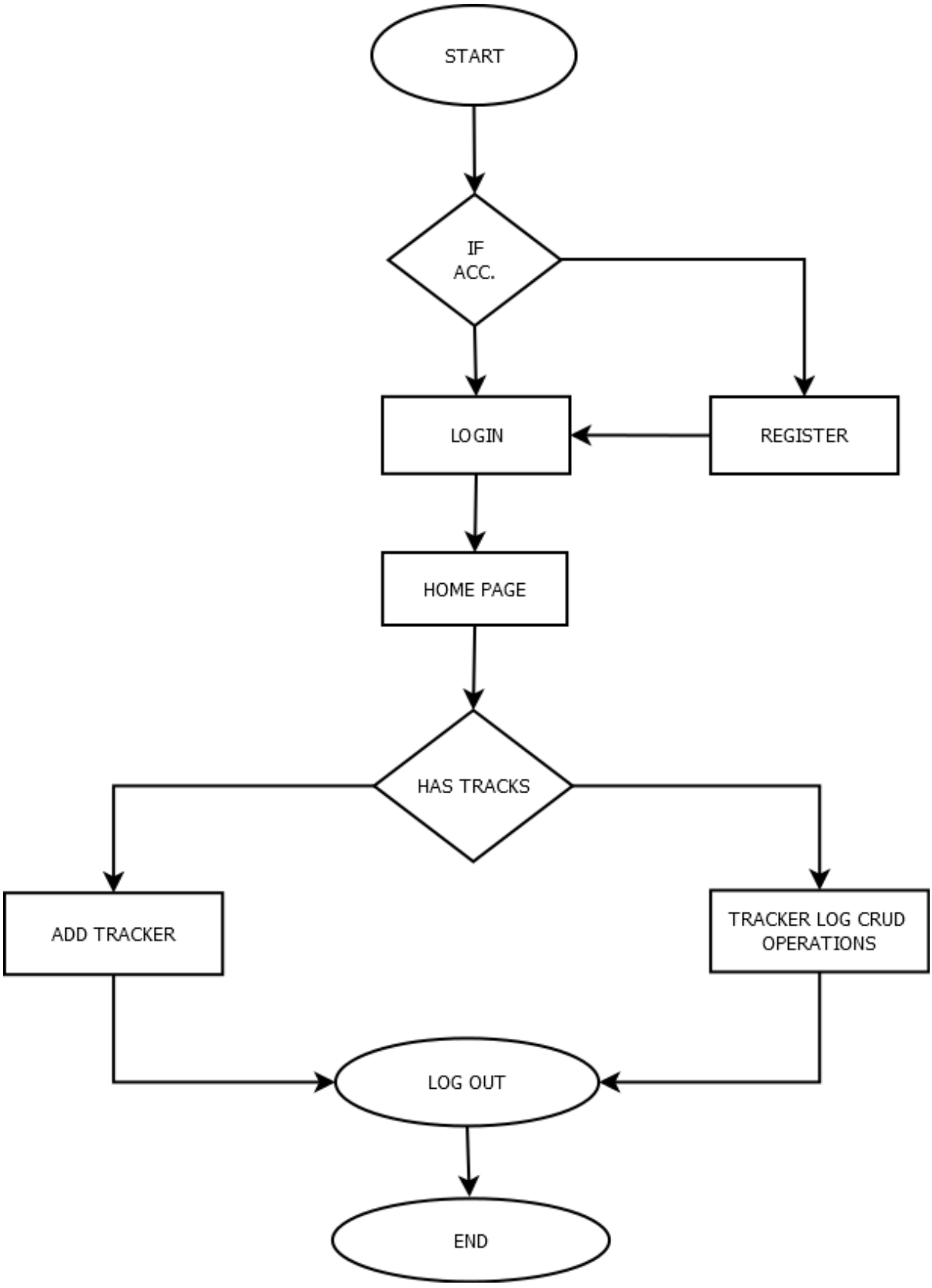
In the fourth phase, the code is developed using the product models, logic and requirements from the previous phases.

5. Testing

After coding is completed the system design is tested. This is to ensure that there are no errors in the system when the client starts using it. A variety of tests are applied depending on the project. Think of quality assurance, system tests, beta tests or unit tests.

If the system passes all the tests, the waterfall will continue to descend. The result of this phase is usually a test report, but in some cases also consists of a User Acceptance Test (UAT).

4.2 System Flowchart



4.3 DFD

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).
On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.
A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).

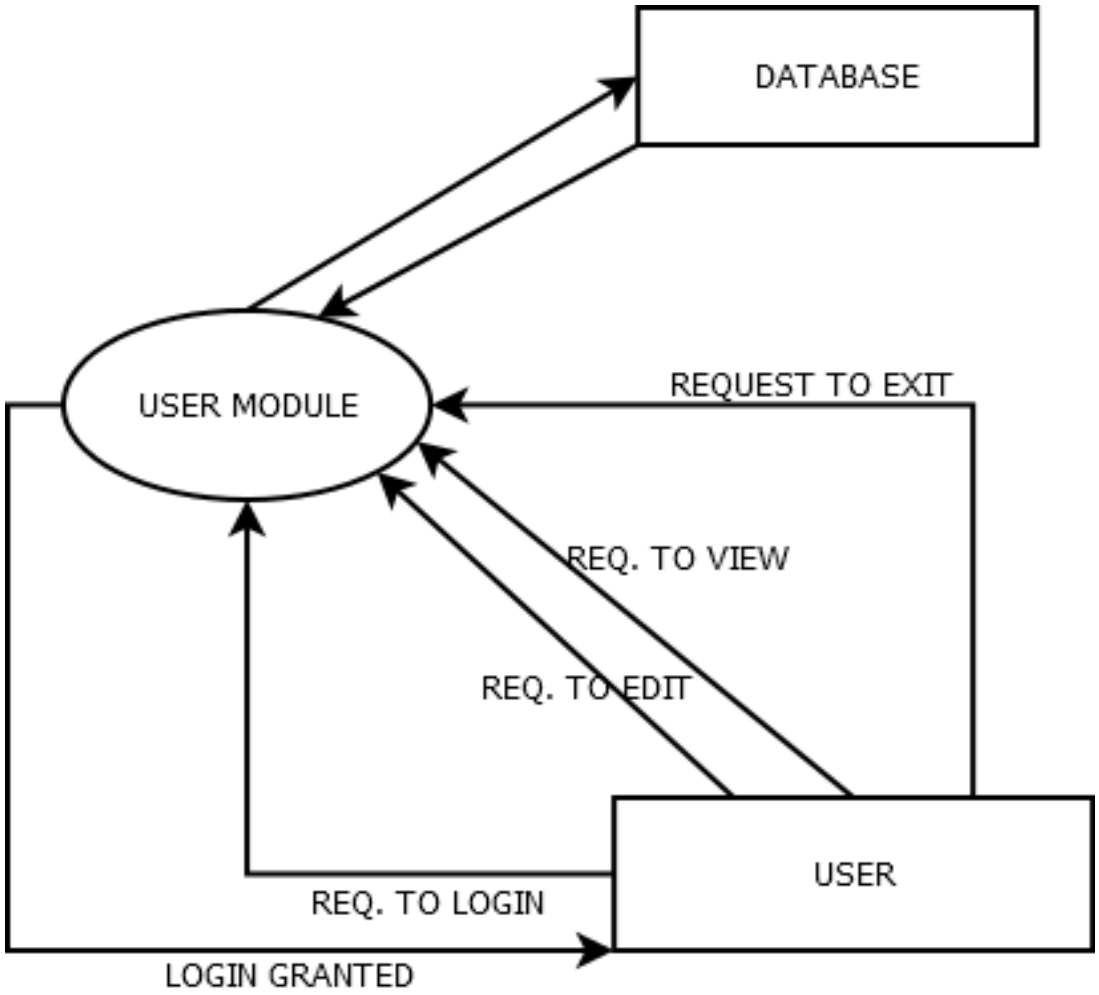
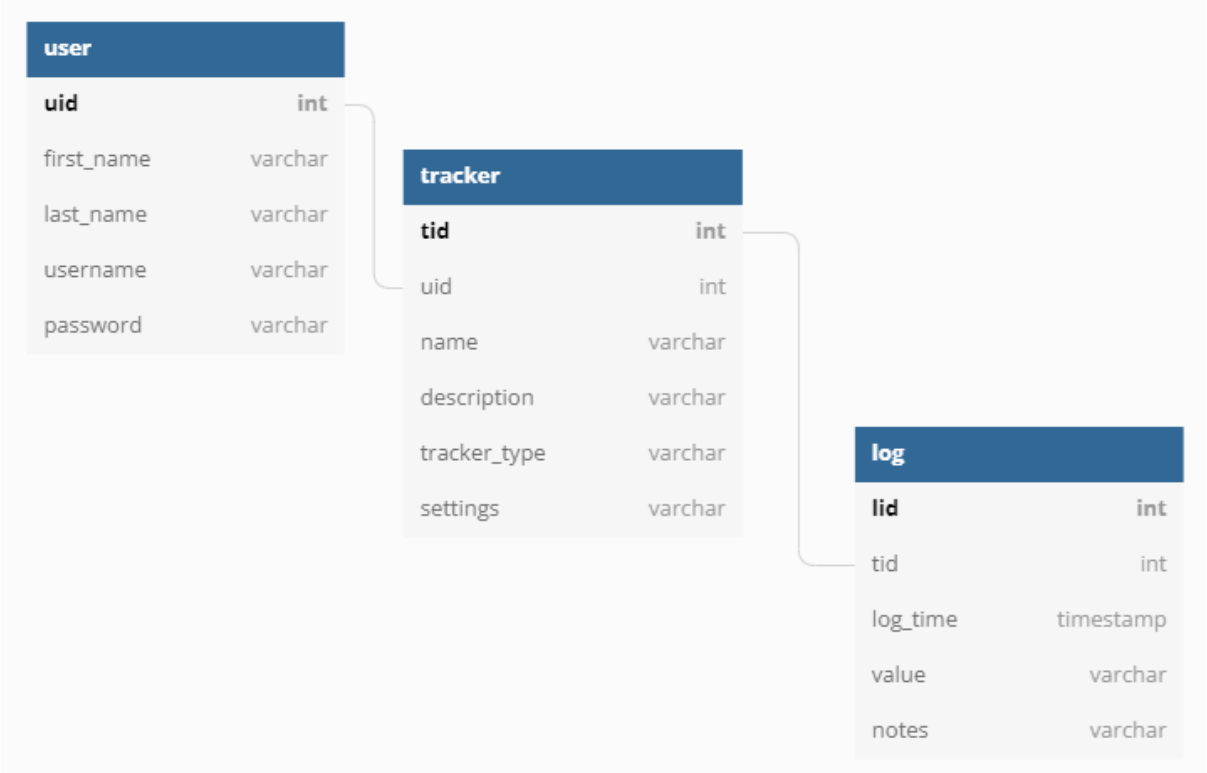
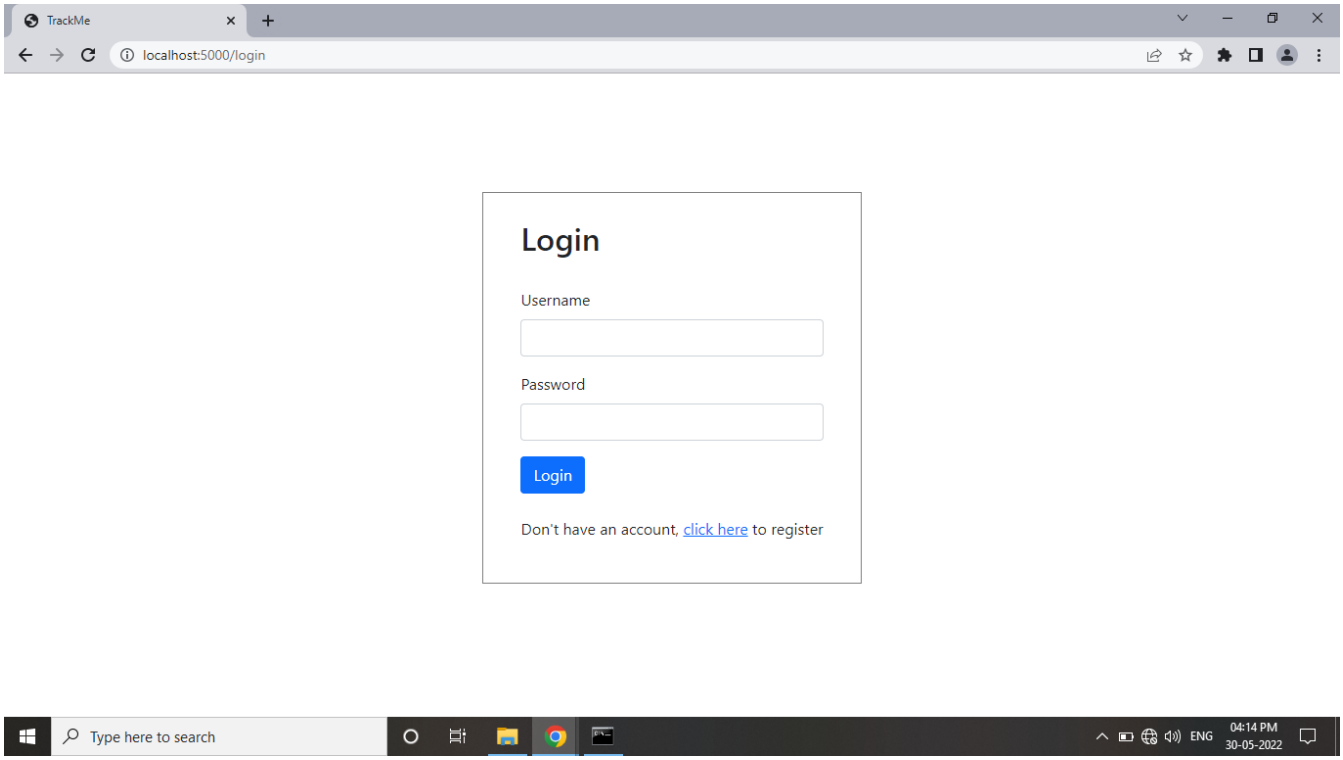


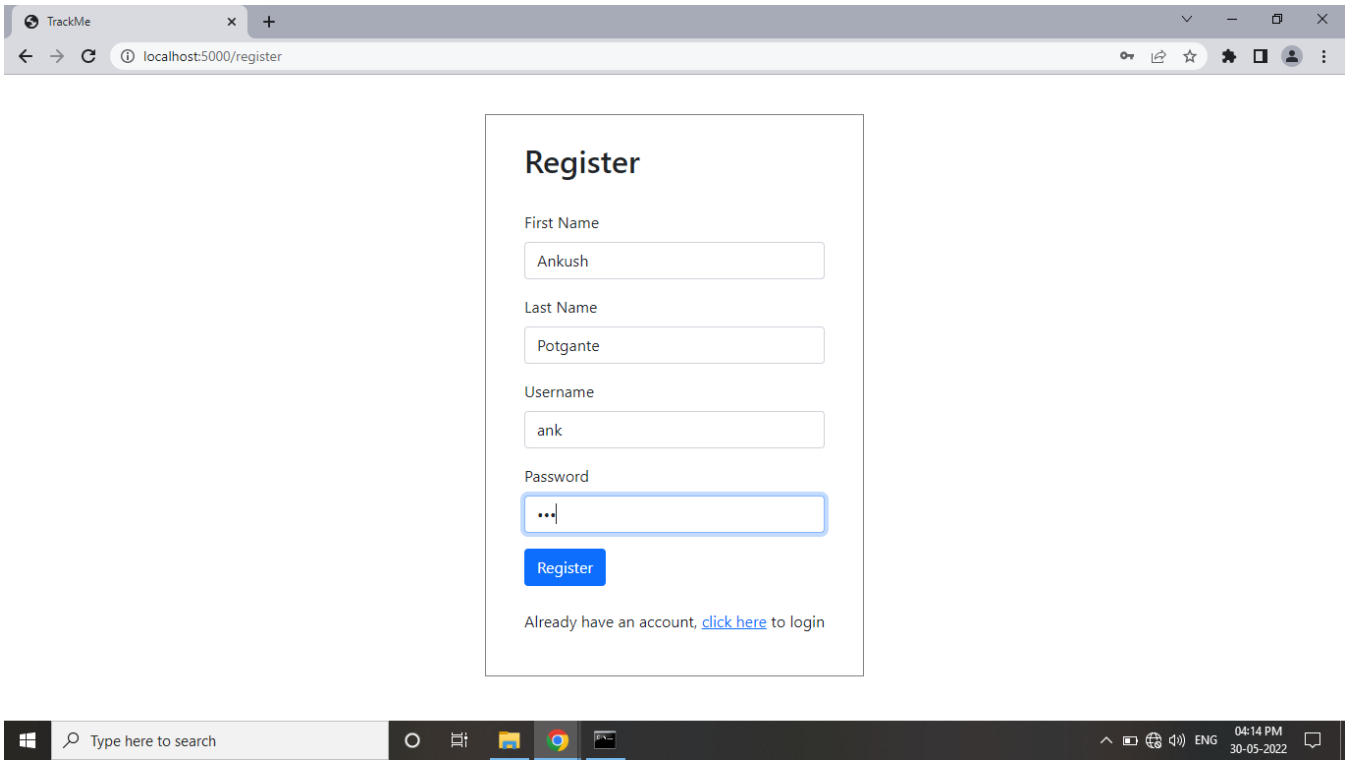
DIAGRAM DFD DIAGRAM OF HABIT TRACKING WEBAPP

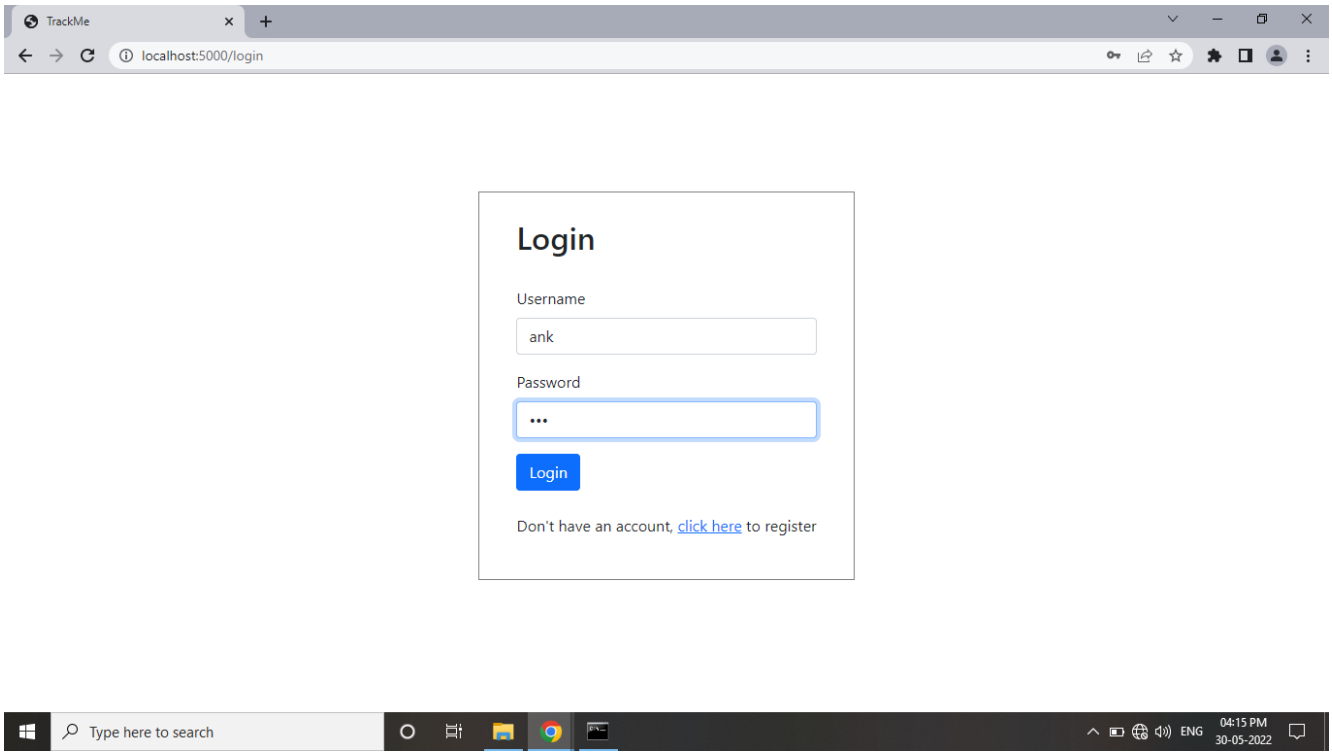
4.5 Data Dictionary, Table Design

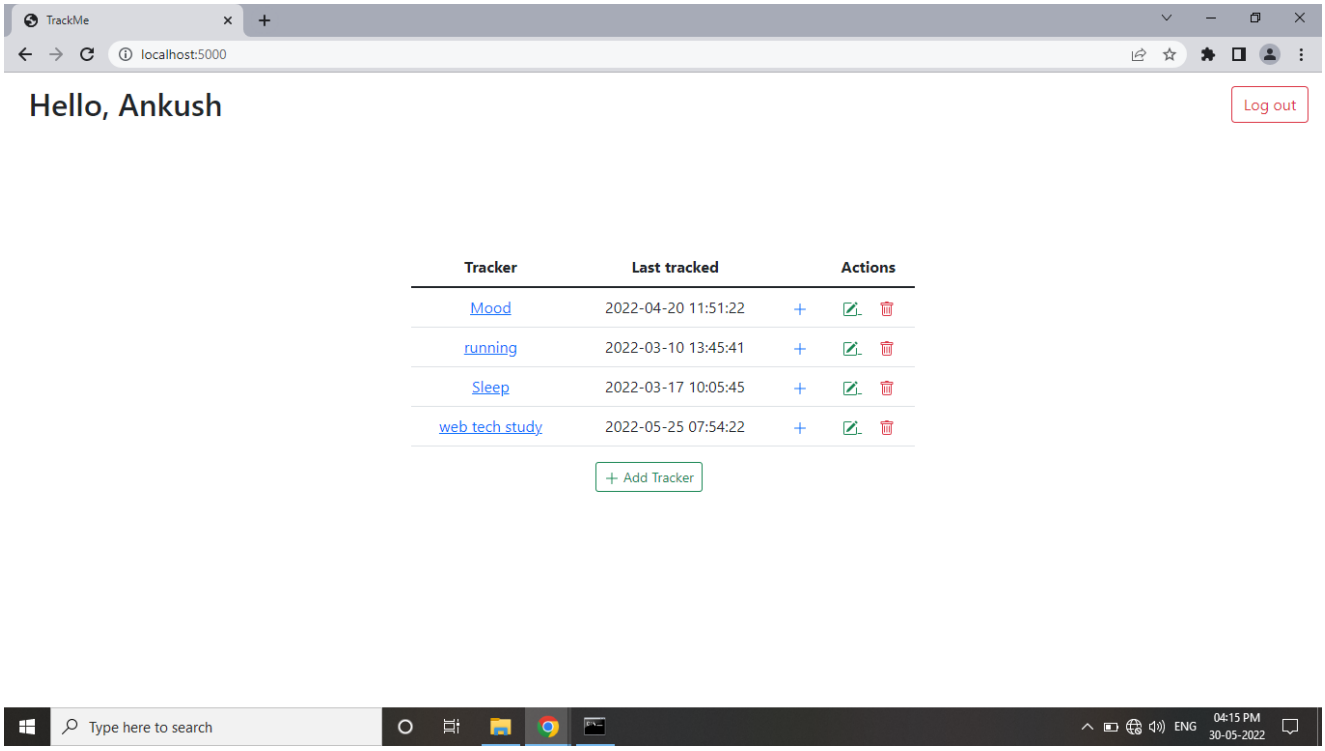


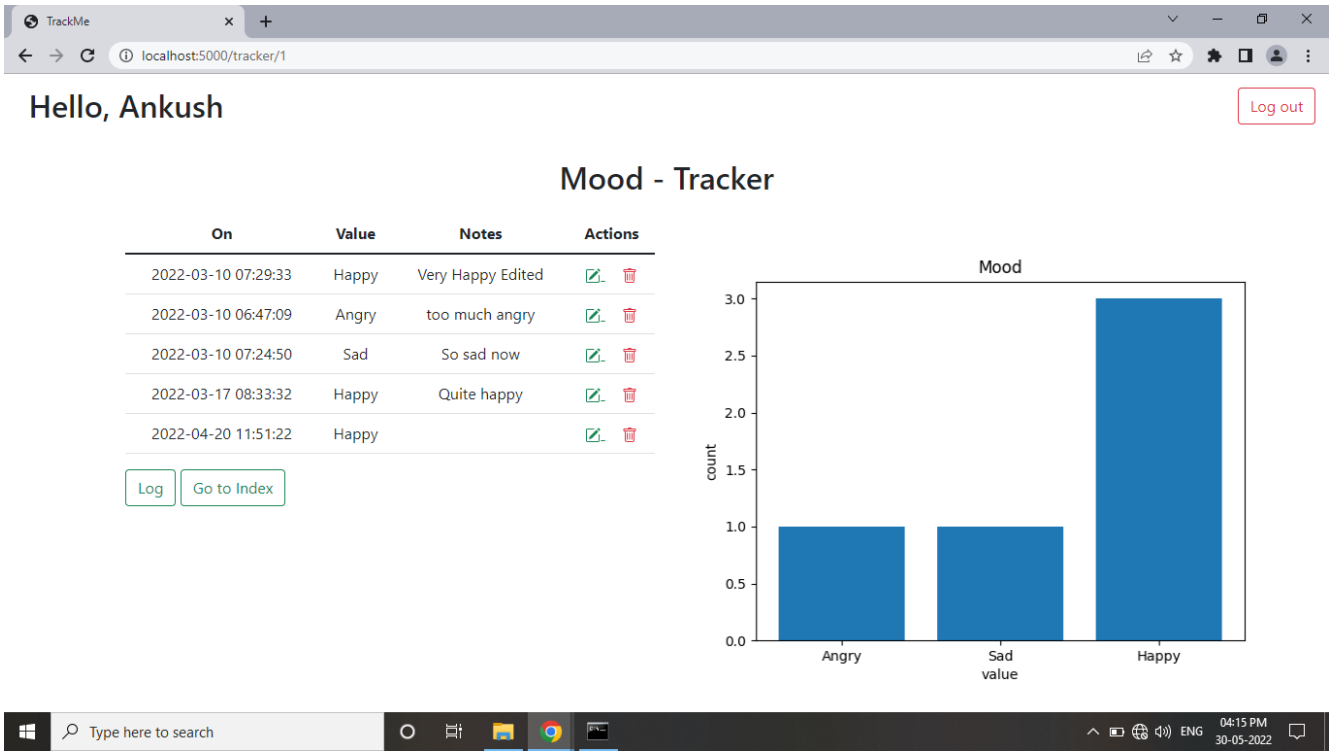
4.6 Front End Design, Menu Tree, Menu Screens, Input Screens:

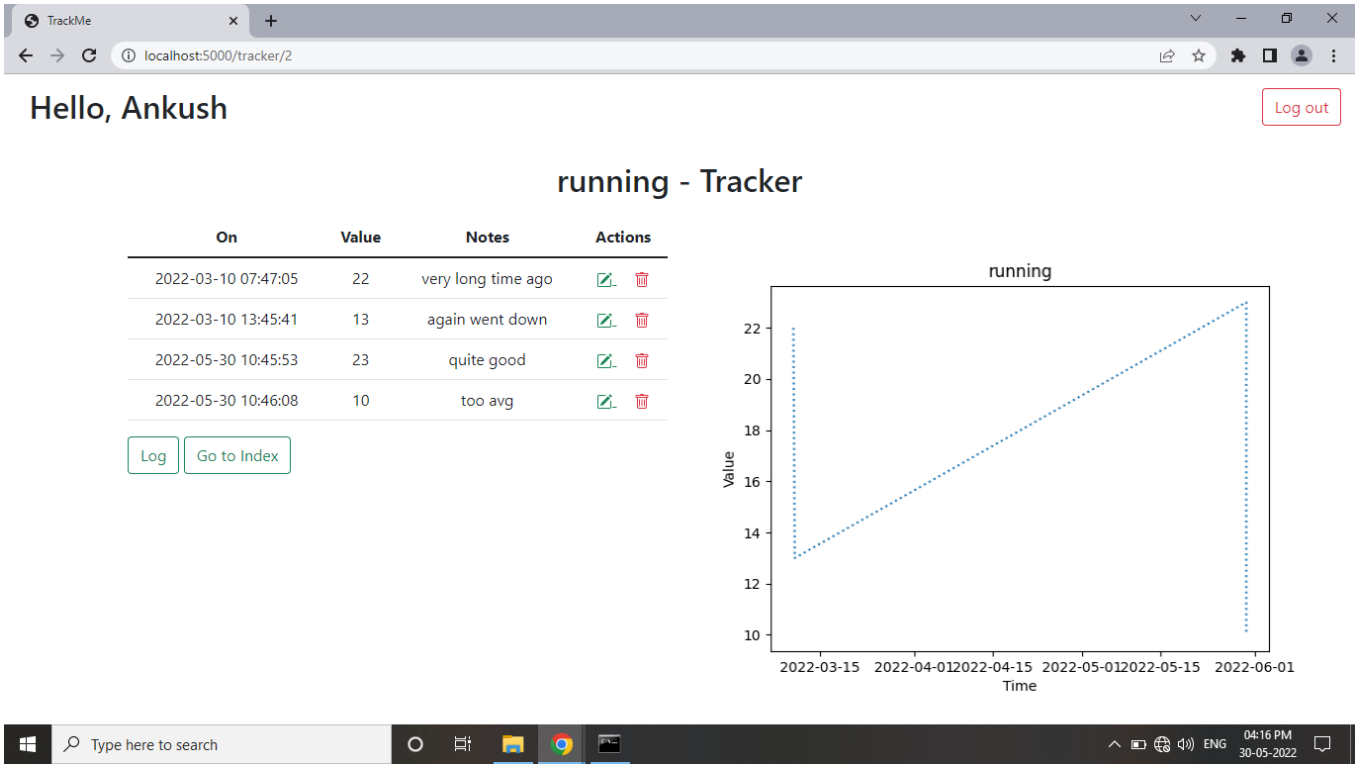


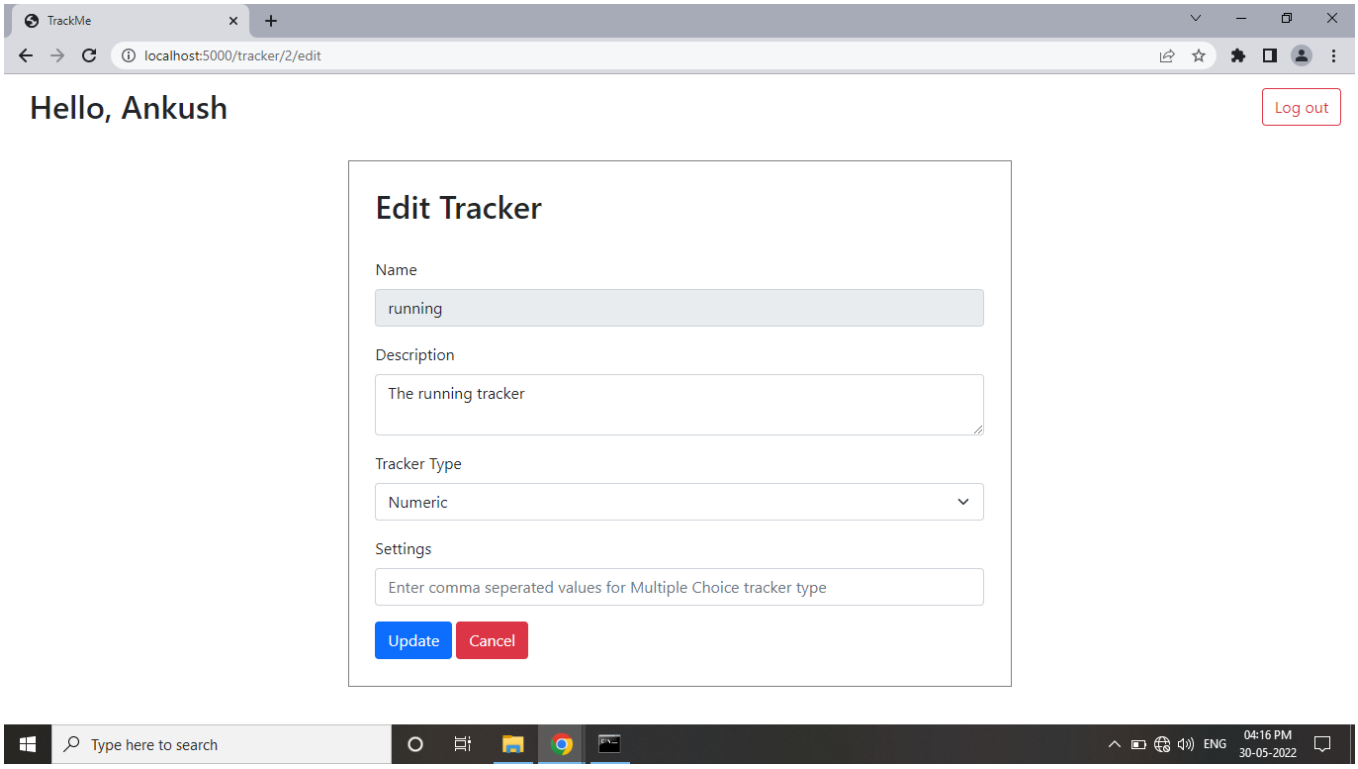

TrackMe
localhost:5000/register
Register
First Name
Ankush
Last Name
Potgante
Username
ank
Password
Register
Already have an account, click here to login
Type here to search
ENG
04:14 PM
30-05-2022

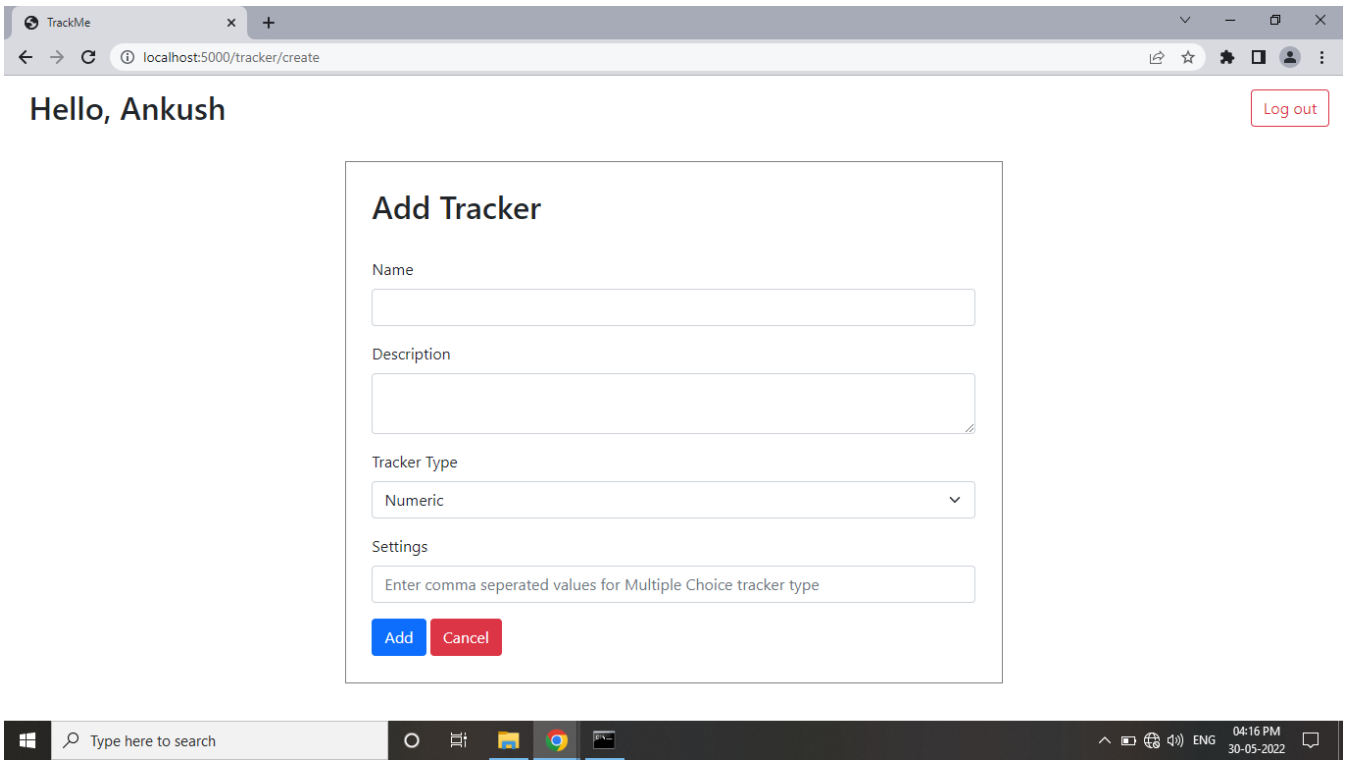

TrackMe
localhost:5000/login
Login
Username
ank
Password
Login
Don't have an account, click here to register
Type here to search
ENG
04:15 PM
30-05-2022

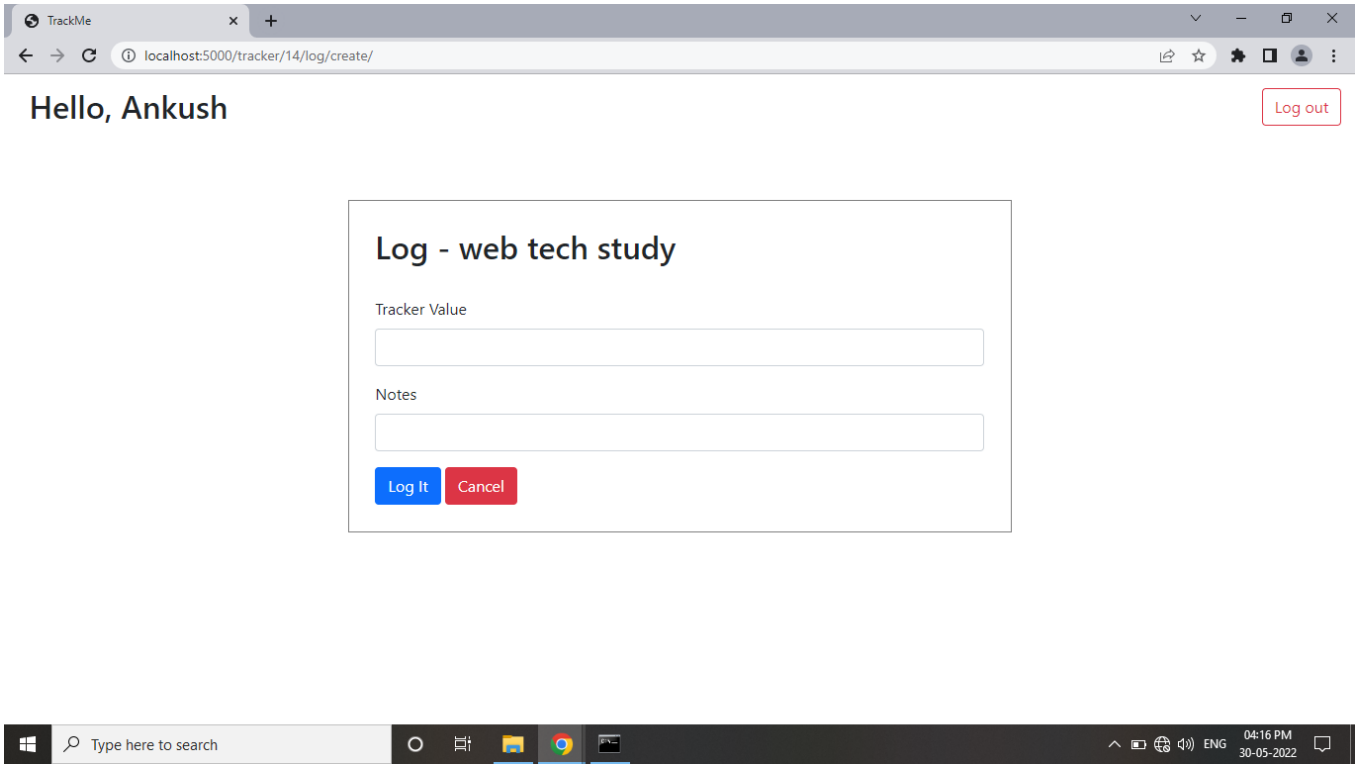

TrackMe
localhost:5000
Hello, Ankush
Log out
Tracker
Last tracked
Actions
Mood
2022-04-20 11:51:22
running
2022-03-10 13:45:41
Sleep
2022-03-17 10:05:45
web tech study
2022-05-25 07:54:22
+ Add Tracker
Type here to search
ENG
04:15 PM
30-05-2022

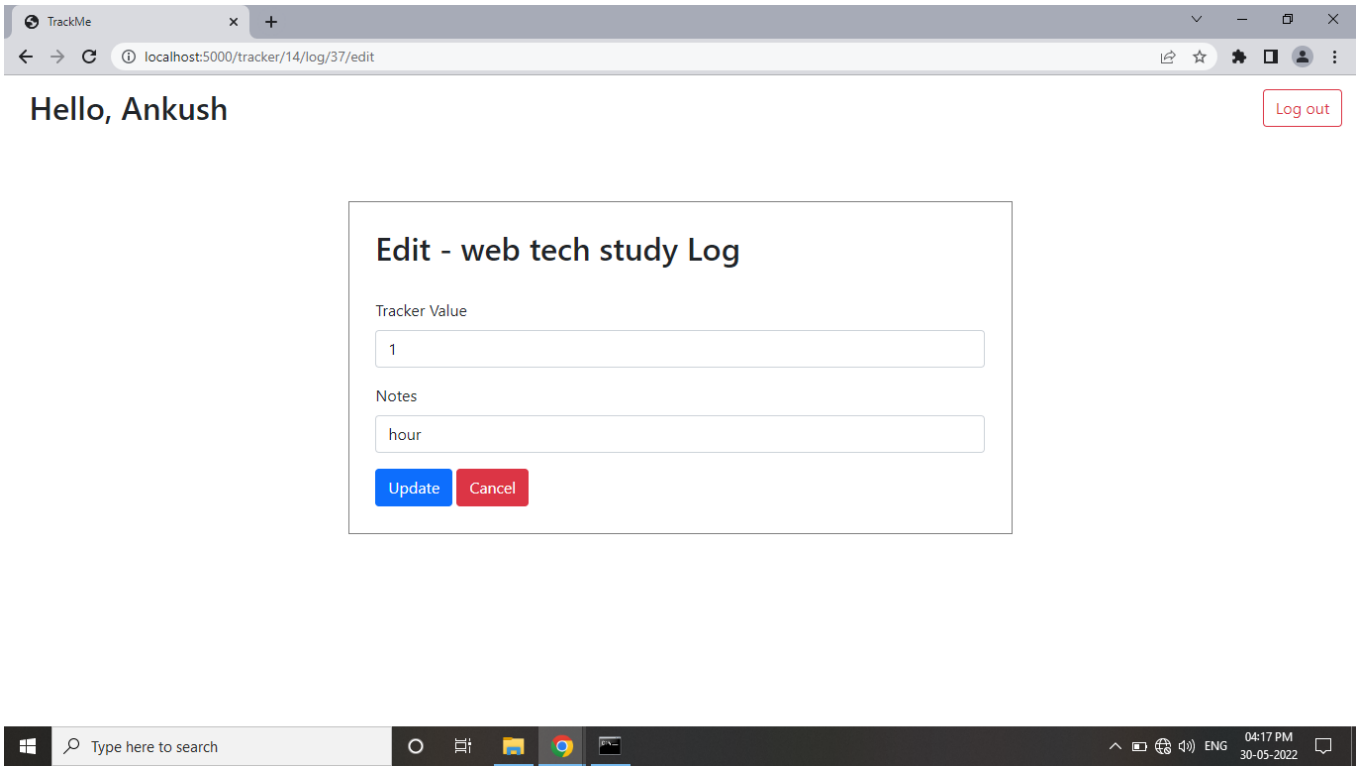

TrackMe
localhost:5000/tracker/1
Hello, Ankush
Log out
Mood - Tracker
On
Value
Notes
Actions
2022-03-10 07:29:33
Happy
Very Happy Edited
2022-03-10 06:47:09
Angry
too much angry
2022-03-10 07:24:50
Sad
So sad now
2022-03-17 08:33:32
Happy
Quite happy
2022-04-20 11:51:22
Happy
Log
Go to Index
Mood
count
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Angry
Sad
Happy
value
Type here to search
ENG
04:15 PM
30-05-2022


TrackMe
localhost:5000/tracker/2
Hello, Ankush
Log out
running - Tracker
On
Value
Notes
Actions
2022-03-10 07:47:05
22
very long time ago
2022-03-10 13:45:41
13
again went down
2022-05-30 10:45:53
23
quite good
2022-05-30 10:46:08
10
too avg
Log
Go to Index
running
Value
Time
2022-03-15
2022-04-01
2022-04-15
2022-05-01
2022-05-15
2022-06-01
Type here to search
ENG
04:16 PM
30-05-2022


TrackMe
localhost:5000/tracker/2/edit
Hello, Ankush
Log out
Edit Tracker
Name
running
Description
The running tracker
Tracker Type
Numeric
Settings
Enter comma seperated values for Multiple Choice tracker type
Update
Cancel
Type here to search
ENG
04:16 PM
30-05-2022


TrackMe
localhost:5000/tracker/create
Hello, Ankush
Log out
Add Tracker
Name
Description
Tracker Type
Numeric
Settings
Enter comma seperated values for Multiple Choice tracker type
Add
Cancel
Type here to search
ENG
04:16 PM
30-05-2022


TrackMe
localhost:5000/tracker/14/log/create/
Hello, Ankush
Log out
Log - web tech study
Tracker Value
Notes
Log It
Cancel
Type here to search
ENG
04:16 PM
30-05-2022


TrackMe
localhost:5000/tracker/14/log/37/edit
Hello, Ankush
Log out
Edit - web tech study Log
Tracker Value
1
Notes
hour
Update
Cancel
Type here to search
ENG
04:17 PM
30-05-2022

4.7 Coding:

Login

```
{% extends 'base.html' %}


{% block content %}


<div class="d-flex justify-content-center" style="margin-top:9%; position: relative;">
   <br>
   <div class="row" style="border:1px solid grey; padding: 2%;">
      <div>
         <h2>Login</h2>
         <br>
         <form action="/login" method="post">
         <div class="mb-3">
            <label for="username" class="form-label">Username </label>
            <input type="text" class="form-control" id="username" name="username"
required>
         </div>


         <div class="mb-3">
            <label for="passwd" class="form-label">Password </label>
            <input type="password" class="form-control" id="passwd" name="passwd"
required>
         </div>
            <input type="submit" class="btn btn-primary" value="Login">
         </form>
         <br>
         <p>Don't have an account, <a href="/register">click here</a> to register</p>
      </div>
   </div>
</div>
```

```
{% endblock %}
```

Register

```
{% extends 'base.html' %}

{% block content %}

<div class="d-flex justify-content-center" style="margin-top:3%; position: relative;">
   <br>
   <div class="row" style="border:1px solid grey; padding: 2%;">
      <div>
         <h2>Register</h2>
         <br>
         <form action="/register" method="post">
         <div class="mb-3">
            <label for="fname" class="form-label">First Name </label>
            <input type="text" class="form-control" id="fname" name="fname" required>
         </div>

         <div class="mb-3">
            <label for="lname" class="form-label">Last Name </label>
            <input type="text" class="form-control" id="lname" name="lname" required>
         </div>

         <div class="mb-3">
            <label for="uname" class="form-label">Username </label>
            <input type="text" class="form-control" id="uname" name="uname" required>
         </div>
```

```
        <div class="mb-3">
            <label for="passwd" class="form-label">Password </label>
            <input type="password" class="form-control" id="passwd" name="passwd"
required>
        </div>

            <input type="submit" class="btn btn-primary" value="Register">
        </form>
        <br>
        <p>Already have an account, <a href="/login">click here</a> to login</p>
      </div>
   </div>
</div>

{% endblock %}
```

Index

```
{% extends 'base.html' %}

{% block content %}

<div class="d-flex justify-content-center" style="margin-top:7%; position: relative;">
   <div class="row" style="padding: 2%;">
      <div>
         <table class="table text-center">
            <thead>
            <th>Tracker</th>
            <th>Last tracked</th>
            <th></th>
            <th>Actions</th>
            </thead>
            <tbody>
```

```
{% for t in tdata %}
<tr>
  <td>
    <a href="{{ url_for('tracker_details', tid=t[0].tid) }}">{{ t[0].name }}</a>
  </td>

  <td>{{ t[1] }}</td>

  <td>
    <a href="{{ url_for('create_tracker_log', t_id=t[0].tid) }}">
      <!-- svg image for log -->
      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor"
        class="bi bi-plus-lg" viewBox="0 0 16 16">
        <path fill-rule="evenodd"
          d="M8 2a.5.5 0 0 1 .5.5v5h5a.5.5 0 0 1 0 1h-5v5a.5.5 0 0 1-1 0v-5h-
5a.5.5 0 0 1 0-1h5v-5A.5.5 0 0 1 8 2Z"/>
      </svg>
    </a>
  </td>
  <td>
    <a href="{{ url_for('tracker_edit', tid=t[0].tid) }}" class="link-success">
      <!-- svg image for edit -->
      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor"
        class="bi bi-pencil-square" viewBox="0 0 16 16">
        <path d="M15.502 1.94a.5.5 0 0 1 0 .706L14.459 3.69l-2-
2L13.502.646a.5.5 0 0 1 .707 0l1.293 1.293zm-1.75 2.456-2-2L4.939 9.21a.5.5 0 0 0-
.121.196l-.805 2.414a.25.25 0 0 0 .316.316l2.414-.805a.5.5 0 0 0 .196-.12l6.813-6.814z"/>
        <path fill-rule="evenodd"
          d="M1 13.5A1.5 1.5 0 0 0 2.5 15h11a1.5 1.5 0 0 0 1.5-1.5v-6a.5.5
0 0-1 0v6a.5.5 0 0 1-.5.5h-11a.5.5 0 0 1-.5-.5v-11a.5.5 0 0 1 .5-.5H9a.5.5 0 0 0 0-1H2.5A1.5
1.5 0 0 0 1 2.5v11z"/>
```

```
                    </svg>
                </a>
                   
                <a href="{{ url_for('tracker_delete', tid=t[0].tid) }}" class="link-danger">
                    <!-- svg image for delete -->
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor"
                         class="bi bi-trash" viewBox="0 0 16 16">
                        <path d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-
.5zm2.5 0a.5.5 0 0 1 .5.5v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-.5zm3 .5a.5.5 0 0 0-1 0v6a.5.5 0 0
0 1 0V6z"/>
                        <path fill-rule="evenodd"
                            d="M14.5 3a1 1 0 0 1-1 1H13v9a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2V4h-
.5a1 1 0 0 1-1-1V2a1 1 0 0 1 1-1H6a1 1 0 0 1 1-1h2a1 1 0 0 1 1 1h3.5a1 1 0 0 1 1
1v1zM4.118 4 4 4.059V13a1 1 0 0 0 1 1h6a1 1 0 0 0 1-1V4.059L11.882 4H4.118zM2.5
3V2h11v1h-11z"/>
                    </svg>
                </a>
            </td>
        </tr>
        {% endfor %}
        </tbody>
    </table>
</div>

<!-- add button -->
<div class="mb-3" style="margin-left: 35%;">
    <a href="{{ url_for('tracker_create') }}" class="btn btn-outline-success btn-sm">
        <!-- svg image for add -->
        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-plus-lg"
             viewBox="0 0 16 16">
            <path fill-rule="evenodd"
```

```
                    d="M8 2a.5.5 0 0 1 .5.5v5h5a.5.5 0 0 1 0 1h-5v5a.5.5 0 0 1-1 0v-5h-5a.5.5 0
0 1 0-1h5v-5A.5.5 0 0 1 8 2Z"/>
                </svg>
                Add Tracker
            </a>
        </div>

    </div>

</div>

{% endblock %}
Track details
{% extends 'base.html' %}

{% block content %}

<div class="container" style="margin-top:2%;">
    <div class="row">
        <div class="text-center" style="margin-bottom: 1%;">
            <h2>{{ tracker.name }} - Tracker</h2>
        </div>
        <div class="col-md-6">
            <table class="table text-center">
                <thead>
                    <th>On</th>
                    <th>Value</th>
                    <th>Notes</th>
                    <th>Actions</th>
                </thead>
                <tbody>
```

```
{% for log in logs %}
<tr>
  <td>{{ log.log_time }}</td>
  <td>{{ log.value }}</td>
  <td>{{ log.notes }}</td>
  <td>
    <a href="{{ url_for('edit_tracker_log', t_id=tracker.tid, l_id=log.lid) }}"
      class="link-success">
      <!-- svg image for edit -->
      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor"
        class="bi bi-pencil-square" viewBox="0 0 16 16">
        <path
          d="M15.502 1.94a.5.5 0 0 1 0 .706L14.459 3.69l-2-
2L13.502.646a.5.5 0 0 1 .707 0l1.293 1.293zm-1.75 2.456-2-2L4.939 9.21a.5.5 0 0 0-
.121.196l-.805 2.414a.25.25 0 0 0 .316.316l2.414-.805a.5.5 0 0 0 .196-.12l6.813-6.814z" />
        <path fill-rule="evenodd"
          d="M1 13.5A1.5 1.5 0 0 0 2.5 15h11a1.5 1.5 0 0 0 1.5-1.5v-6a.5.5 0
0 0-1 0v6a.5.5 0 0 1-.5.5h-11a.5.5 0 0 1-.5-.5v-11a.5.5 0 0 1 .5-.5H9a.5.5 0 0 0 0-1H2.5A1.5
1.5 0 0 0 1 2.5v11z" />
      </svg>
    </a>
       
    <a href="{{ url_for('delete_tracker_log', t_id=tracker.tid, l_id=log.lid) }}"
      class="link-danger">
      <!-- svg image for delete -->
      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor"
        class="bi bi-trash" viewBox="0 0 16 16">
        <path
          d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-
.5zm2.5 0a.5.5 0 0 1 .5.5v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1 .5-.5zm3 .5a.5.5 0 0 0-1 0v6a.5.5 0 0
0 1 0V6z" />
```

```
                        <path fill-rule="evenodd"
                            d="M14.5 3a1 1 0 0 1-1 1H13v9a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2V4h-.5a1 1 0 0 1-1-1V2a1 1 0 0 1 1-1H6a1 1 0 0 1 1-1h2a1 1 0 0 1 1 1h3.5a1 1 0 0 1 1 1v1zM4.118 4 4 4.059V13a1 1 0 0 0 1 1h6a1 1 0 0 0 1-1V4.059L11.882 4H4.118zM2.5 3V2h11v1h-11z" />
                    </svg>
                </a>
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
<a href="{{ url_for('create_tracker_log', t_id=tracker.tid) }}" class="btn btn-outline-success">Log</a>
<a href="/" class="btn btn-outline-success">Go to Index</a>
</div>

<div class="col-md-6" style="padding: 1%;">
<img src="{{ url_for('static', filename='img/plot.png') }}" alt="tracker plot" height="480px" width="650px">
</div>

</div>
</div>

{% endblock %}

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>TrackMe</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css.map') }}">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/custom.css') }}">
</head>
<body>

    <div class="container-fluid">

        {% if session['username'] %}
        <div class="title">
            <h2>Hello, {{ session['fname'] }}</h2>
        </div>

        <div class="logout">
            <!-- <a href="/logout" class="btn btn-danger">Logout</a> -->
            <a href="/logout" class="btn btn-outline-danger ">
                <span class="glyphicon glyphicon-log-out"></span> Log out
              </a>
        </div>
        {% endif %}

        <div class="row">
            {% with messages = get_flashed_messages() %}
            {% if messages %}
                {% for message in messages %}
                    <p class="alert alert-warning" role="alert">{{ message }}</p>
                {% endfor %}
            {% endif %}
```

```
        {% endwith %}
        </div>

        {% block content %}

        {% endblock %}

    </div>

    <!-- js section -->
    <script src="{{ url_for('static', filename='js/bootstrap.bundle.min.js') }}"></script>
    <script src="{{ url_for('static', filename='js/bootstrap.bundle.min.js.map') }}"></script>
    <script src="{{ url_for('static', filename='js/custom.js') }}"></script>
</body>
</html>
{% extends 'base.html' %}

{% block content %}

<div class="d-flex justify-content-center" style="margin-top:5%;">
    <div class="col-md-6" style="border:1px solid grey; padding: 2%;">
        <h2>Edit - {{ tracker.name }} Log</h2>
        <br>
        <form action="/tracker/{{ tracker.tid }}/log/{{ log.lid }}/edit" method="post">
            {% if tracker.tracker_type == 'Numeric' %}
            <div class="mb-3">
                <label class="form-label" for="tnumval">Tracker Value</label>
                <input type="text" class="form-control" name="tval" id="tnumval" value="{{
log.value }}" required>
            </div>
            {% endif %}
```

```
{% if tracker.tracker_type == 'Boolean' %}
<div class="mb-3">
  <label class="form-label" for="tval">Tracker Value</label>
  <br>
  <input type="radio" class="form-check-input" id="y" name="tval" value="Yes" {% if log.value== 'Yes' %} checked {% endif %}>
  <label for="y" class="form-label">Yes</label><br>
  <input type="radio" class="form-check-input" id="n" name="tval" value="No" {% if log.value== 'No' %} checked {% endif %}>
  <label for="n" class="form-label">No</label><br>
</div>
{% endif %}
{% if tracker.tracker_type == 'Multiple Choice' %}
<div class="mb-3">
  <label for="tchval" class="form-label">Tracker Value</label>
  <select id="tchval" class="form-select" name="tval" required>
    {% for option in options %}
    <option value="{{ option }}" {% if log.value== option %} selected {% endif %}>{{ option }}</option>
    {% endfor %}
  </select>
</div>
{% endif %}
<div class="mb-3">
  <label for="tnotes" class="form-label">Notes</label>
  <input type="text" class="form-control"  value="{{ log.notes }}" name="tnotes" id="tnotes">
</div>
<input type="submit" class="btn btn-primary" value="Update">
<a href="{{ url_for('tracker_details', tid=tracker.tid) }}" class="btn btn-danger">Cancel</a>
</form>
```

```
    </div>
</div>

{% endblock %}
```

Css

```
.title {
    margin-top: 1%;
    margin-left: 1%;
}

.logout{
    position: absolute;
    top: 0;
    right: 0;
    margin-top: 1%;
    margin-right: 1%;
}
```

Models

```
from application.database import db

class User(db.Model):
    uid = db.Column(db.Integer, primary_key=True, autoincrement=True)
    first_name = db.Column(db.String, nullable=False)
    last_name = db.Column(db.String)
    username = db.Column(db.String, unique=True, nullable=False)
    password = db.Column(db.String, nullable=False)
```

```
    trackers = db.relationship('Tracker', backref='usr', lazy='dynamic',
                               cascade='all, delete-orphan', passive_deletes=True)

class Tracker(db.Model):
    tid = db.Column(db.Integer, primary_key=True, autoincrement=True)
    uid = db.Column(db.Integer, db.ForeignKey(
        'user.uid', ondelete='CASCADE'), nullable=False)
    name = db.Column(db.String, nullable=False)
    description = db.Column(db.String)
    tracker_type = db.Column(db.String, nullable=False)
    settings = db.Column(db.String)

    logs = db.relationship('Log', backref='trkr', lazy='dynamic',
                           cascade='all, delete-orphan', passive_deletes=True)

class Log(db.Model):
    lid = db.Column(db.Integer, primary_key=True, autoincrement=True)
    tid = db.Column(db.Integer, db.ForeignKey(
        'tracker.tid', ondelete='CASCADE'), nullable=False)
    log_time = db.Column(db.TIMESTAMP, server_default=db.func.now())
    value = db.Column(db.String)
    notes = db.Column(db.String)
```

App

```
import os
from flask import Flask
from flask_restful import Api
from application.database import db
```

```
from application.config import LocalDevelopmentConfig, ProductionDevelopmentConfig

app = None
api = None

def create_app():
    app = Flask(__name__, template_folder="templates")

    if os.getenv('ENV', 'development') == 'production':
        print("Starting production config...")
        app.config.from_object(ProductionDevelopmentConfig)
    else:
        print("Starting local config...")
        app.config.from_object(LocalDevelopmentConfig)

    db.init_app(app)
    api = Api(app)
    app.app_context().push()
    return app, api

app, api = create_app()
app.secret_key = "my_very_very_secret_key"

# importing all controllers

from application.controllers import *

# import all resources

from application.resources import UserResource, TrackerResource, LogResource
```

```
api.add_resource(UserResource, "/api/user", "/api/user/<int:uid>")
api.add_resource(TrackerResource, "/api/tracker", "/api/tracker/<int:tid>")
api.add_resource(LogResource, "/api/log", "/api/log/<int:lid>")

if __name__ == '__main__':
    db.create_all()
    app.run()

Resources
from flask_restful import Resource, reqparse, marshal_with, fields, abort, inputs
from application.models import User, Tracker, Log
from application.database import db
from hashlib import md5

# User output
user_output = {
    'uid': fields.Integer,
    'username': fields.String,
    'first_name': fields.String,
    'last_name': fields.String,
    'password': fields.String
}

# User request parsers
user_create_req = reqparse.RequestParser()
user_create_req.add_argument(
    'first_name', type=str, help="First name is required and is string", required=True)
user_create_req.add_argument('last_name', type=str, help="Last name is string")
user_create_req.add_argument(
```

```
    'username', type=str, help="Username is required and is string", required=True)
user_create_req.add_argument(
    'password', type=str, help="email is required and is string", required=True)

user_update_req = reqparse.RequestParser()
user_update_req.add_argument(
    'first_name', type=str, help="First name is required and is string", required=True)
user_update_req.add_argument(
    'last_name', type=str, help="Last name is required and is string", required=True)

# User resource
class UserResource(Resource):

    @marshal_with(user_output)
    def get(self, uid):
        user = User.query.get(uid)
        if user:
            return user
        else:
            abort(404, message="User not found!")

    @marshal_with(user_output)
    def post(self):
        data = user_create_req.parse_args()
        password = data.password
        p = md5(password.encode())
        passwd = p.hexdigest()
        user = User(first_name=data.first_name, last_name=data.last_name,
                    username=data.username, password=passwd)
        if user:
```

```
        db.session.add(user)
        db.session.commit()
        return user
    else:
        abort(500, message="Something went wrong!")

def delete(self, uid):
    user = User.query.get(uid)
    if user:
        trackers = user.trackers.all()
        if trackers:
            for tracker in trackers:
                Log.query.filter(Log.tid == tracker.tid).delete()
                db.session.delete(tracker)
        db.session.delete(user)
        db.session.commit()
        return "User deleted successfully!"
    else:
        abort(404, message="User not found!")

@marshal_with(user_output)
def put(self, uid):
    user = User.query.get(uid)
    if user:
        data = user_update_req.parse_args()
        user.first_name = data.first_name
        user.last_name = data.last_name
        db.session.add(user)
        db.session.commit()
        return user
```

```
        else:
            abort(404, message="User not found!")


# Tracker output
tracker_output = {
    'tid': fields.Integer,
    'uid': fields.Integer,
    'name': fields.String,
    'description': fields.String,
    'tracker_type': fields.String,
    'settings': fields.String
}


# Tracker request parsers
tracker_create_req = reqparse.RequestParser()
tracker_create_req.add_argument(
    'uid', type=int, help="uid required and is Integer", required=True)
tracker_create_req.add_argument(
    'name', type=str, help="name required and is string", required=True)
tracker_create_req.add_argument(
    'description', type=str, help="description is string")
tracker_create_req.add_argument(
    'tracker_type', type=str, help="tracker_type required and is string", required=True)
tracker_create_req.add_argument(
    'settings', type=str, help="settings and is string")


tracker_update_req = reqparse.RequestParser()
# tracker_update_req.add_argument('name', type=str, help="name required and is string",
required=True)
tracker_update_req.add_argument(
    'description', type=str, help="description is string")
```

```
tracker_update_req.add_argument(
    'tracker_type', type=str, help="tracker_type required and is string", required=True)
tracker_update_req.add_argument(
    'settings', type=str, help="settings and is string")

# Tracker resource
class TrackerResource(Resource):

    @marshal_with(tracker_output)
    def get(self, tid):
        tracker = Tracker.query.get(tid)
        if tracker:
            return tracker
        else:
            abort(404, message="Tracker not found!")

    @marshal_with(tracker_output)
    def post(self):
        data = tracker_create_req.parse_args()
        u = User.query.get(data.uid)
        if u:
            tracker = Tracker(uid=data.uid, name=data.name, description=data.description,
                              tracker_type=data.tracker_type, settings=data.settings)
            db.session.add(tracker)
            db.session.commit()
            return tracker
        else:
            abort(400, message='Enter correct user id')

    def delete(self, tid):
```

```
        tracker = Tracker.query.get(tid)
        if tracker:
            Log.query.filter(Log.tid == tracker.tid).delete()
            db.session.delete(tracker)
            db.session.commit()
            return "Tracker deleted successfully!"
        else:
            abort(404, message="Tracker not found!")

    @marshal_with(tracker_output)
    def put(self, tid):
        tracker = Tracker.query.get(tid)
        if tracker:
            data = tracker_update_req.parse_args()
            tracker.tracker_type = data.tracker_type
            tracker.description = data.description
            tracker.settings = data.settings
            db.session.add(tracker)
            db.session.commit()
            return tracker
        else:
            abort(404, message="Tracker not found!")

# Tracker output
log_output = {
    'lid': fields.Integer,
    'tid': fields.Integer,
    'log_time': fields.DateTime,
    'value': fields.String,
    'notes': fields.String,
```

```
}


# Tracker request parsers
log_create_req = reqparse.RequestParser()
log_create_req.add_argument(
    'tid', type=int, help="tid required and is string", required=True)
# log_create_req.add_argument('log_time', type=inputs.datetime_from_iso8601,
help="log_time required and is datetime", required=True)
log_create_req.add_argument(
    'value', type=str, help="value is required is string", required=True)
log_create_req.add_argument('notes', type=str, help="notes is string")


log_update_req = reqparse.RequestParser()
# log_update_req.add_argument('log_time', type=inputs.datetime_from_iso8601,
help="log_time required and is datetime", required=True)
log_update_req.add_argument(
    'value', type=str, help="alue is required is string", required=True)
log_update_req.add_argument('notes', type=str, help="notes is string")


# Log resource
class LogResource(Resource):


    @marshal_with(log_output)
    def get(self, lid):
        log = Log.query.get(lid)
        if log:
            return log
        else:
            abort(404, message="Log not found!")


    @marshal_with(log_output)
```

```
def post(self):
    data = log_create_req.parse_args()
    t = Tracker.query.get(data.tid)
    if t:
        if t.tracker_type == "Numeric":
            try:
                temp = float(data.value)
            except:
                abort(400, message="Invalid tracker value")

        if t.tracker_type == "Multiple Choice":
            options = [i.strip() for i in t.settings.split(",")]
            if data.value not in options:
                abort(400, message="Invalid tracker value")

        if t.tracker_type == "Boolean":
            if data.value not in ["Yes", "No"]:
                abort(400, message="Invalid tracker value")

        log = Log(tid=data.tid, value=data.value, notes=data.notes)
        db.session.add(log)
        db.session.commit()
        return log
    else:
        abort(400, message="Enter correct tracker id")

def delete(self, lid):
    log = Log.query.get(lid)
    if log:
        db.session.delete(log)
```

```
            db.session.commit()
            return "log deleted successfully!"
        else:
            abort(404, message="Log not found!")

    @marshal_with(log_output)
    def put(self, lid):
        log = Log.query.get(lid)
        if log:
            data = log_update_req.parse_args()
            log.value = data.value
            log.notes = data.notes
            db.session.add(log)
            db.session.commit()
            return log
        else:
            abort(404, message="Log not found!")
```

Database

```
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy.ext.declarative import declarative_base

engine = None
Base = declarative_base()
db = SQLAlchemy()
```

Controls

```
from flask import request, render_template, redirect, session, flash, url_for
from flask import current_app as app
from application.models import User, Tracker, Log
```

```
from application.database import db
from hashlib import md5
import matplotlib

matplotlib.use('Agg')
import matplotlib.pyplot as plt
from sqlalchemy import desc

ttypes = ('Multiple Choice', 'Numeric', 'Boolean')

@app.route("/")
def index():
    if "username" in session:
        user = User.query.get(session['user_id'])

        if not user:
            return redirect(url_for('logout'))

        tdata = []
        for tracker in user.trackers.all():
            logs = tracker.logs.all()
            last_log = None
            if logs:
                last_log = tracker.logs.order_by(desc(Log.log_time)).first().log_time
            tdata.append([tracker, last_log])

        return render_template("index.html", tdata=tdata)
    else:
        return redirect("/login")
```

```
@app.route("/register", methods=["GET", "POST"])
def register():
    if "username" in session:
        return redirect("/")

    if request.method == "POST":
        fname = request.form.get("fname")
        lname = request.form.get("lname")
        uname = request.form.get("uname")
        password = request.form.get("passwd")

        if uname and fname and password:

            p = md5(password.encode())
            passwd = p.hexdigest()

            u = User(first_name=fname, last_name=lname,
                    username=uname, password=passwd)
            db.session.add(u)
            db.session.commit()
            return redirect("/")
        else:
            flash("Enter correct data!!!")

    return render_template("register.html")

@app.route("/login", methods=["GET", "POST"])
def login():
    if "username" in session:
        return redirect("/")
```

```
    if request.method == "POST":

        username = request.form.get('username')
        password = request.form.get('passwd')

        if username and password:

            u = User.query.filter(User.username == username).first()

            p = md5(password.encode())

            passwd = p.hexdigest()

            if u and (u.password == passwd):
                session["username"] = username
                session["user_id"] = u.uid
                session["fname"] = u.first_name
                return redirect("/")
            else:
                flash("Something went wrong!!")

    return render_template("login.html")

@app.route("/logout")
def logout():
    if "username" in session:
        session.pop("username")
        session.pop("user_id", "username")
        session.pop("fname", "username")
```

```
    return redirect("/")

def get_tracker_plot(tracker_id):
    t = Tracker.query.get(tracker_id)
    logs = t.logs.order_by(Log.log_time).all()
    x = []
    y = []
    if t.tracker_type == 'Multiple Choice':
        temp = []
        for l in logs:
            temp.append(l.value)
        x = list(set(temp))

        for i in x:
            y.append(temp.count(i))

        plt.bar(x, y)
        plt.xlabel('value')
        plt.ylabel('count')
        plt.title(t.name)
        plt.savefig('static/img/plot.png')
        plt.clf()

    elif t.tracker_type == 'Numeric':
        for l in logs:
            x.append(l.log_time)
            try:
                y.append(float(l.value))
            except:
```

```
                print("Tracker has few invalid logs")
                x.remove(l.log_time)
                continue

        plt.plot(x, y, linestyle='dotted')
        plt.xlabel('Time')
        plt.ylabel('Value')
        plt.title(t.name)
        plt.savefig('static/img/plot.png')
        plt.clf()

    elif t.tracker_type == 'Boolean':
        temp = []
        for l in logs:
            temp.append(l.value)

        x = ['Yes', 'No']
        y = [temp.count(x[0]), temp.count(x[1])]

        plt.bar(x, y)
        plt.xlabel('value')
        plt.ylabel('count')
        plt.title(t.name)
        plt.savefig('static/img/plot.png')
        plt.clf()

    else:
        print("Wrong tracker type!")

@app.route("/tracker/<int:tid>")
```

```
def tracker_details(tid):
    if "username" in session:
        tracker = Tracker.query.get(tid)
        logs = Log.query.filter(Log.tid == tracker.tid)
        if (tracker.tracker_type in ttypes):
            if logs:
                get_tracker_plot(tracker.tid)
            return render_template("tracker_details.html", tracker=tracker, logs=logs)
        else:
            flash("tracker not found!")
            return redirect("/")
    else:
        return redirect("/login")


@app.route("/tracker/create", methods=["GET", 'POST'])
def tracker_create():
    if "username" in session:
        if request.method == 'POST':
            tname = request.form.get("tname")
            desc = request.form.get("desc")
            ttype = request.form.get("ttype")
            settings = request.form.get("settings")


            if tname and (ttype in ttypes):
                t = Tracker(uid=session['user_id'], tracker_type=ttype,
                        name=tname, description=desc, settings=settings)
                db.session.add(t)
                db.session.commit()
                return redirect("/")
        return render_template("add_tracker.html")
```

```
    else:
        return redirect("/login")

@app.route("/tracker/<int:tid>/delete")
def tracker_delete(tid):
    if "username" in session:
        tracker = Tracker.query.get(tid)
        if tracker:
            Log.query.filter(Log.tid == tracker.tid).delete()
            db.session.delete(tracker)
            db.session.commit()
        else:
            flash("Something went wrong!")
        return redirect("/")
    else:
        return redirect("/login")

@app.route("/tracker/<int:tid>/edit", methods=['GET', 'POST'])
def tracker_edit(tid):
    if "username" in session:
        tracker = Tracker.query.get(tid)
        if tracker:
            if request.method == 'POST':
                desc = request.form['desc']
                ttype = request.form['ttype']
                settings = request.form['settings']

                if ttype in ttypes:
                    tracker.description = desc
                    tracker.tracker_type = ttype
```

```
                tracker.settings = settings

                db.session.add(tracker)
                db.session.commit()
                return redirect("/")

        return render_template("edit_tracker.html", tracker=tracker)
    else:
        flash("Something went wrong!")
        return redirect("/")
else:
    return redirect("/login")

@app.route("/tracker/<int:t_id>/log/create/", methods=['GET', 'POST'])
def create_tracker_log(t_id):
    if "username" in session:
        tracker = Tracker.query.get(t_id)
        if tracker:
            options = None
            if tracker.tracker_type == "Multiple Choice":
                options = [i.strip() for i in tracker.settings.split(",")]
            if request.method == 'POST':
                tval = request.form['tval']
                notes = request.form['tnotes']

                if tracker.tracker_type == "Numeric":
                    try:
                        temp = float(tval)
                        print("Unknown numeric value", temp)
                    except:
```

```
                flash("Please enter numeric value")
                return render_template("add_tracker_log.html", tracker=tracker,
options=options)
            if tracker.tracker_type == "Multiple Choice":
                if tval not in options:
                    flash("Please enter appropriate option")
                    return render_template("add_tracker_log.html", tracker=tracker,
options=options)
            if tracker.tracker_type == "Boolean":
                if tval not in ["Yes", "No"]:
                    flash("Please enter appropriate option")
                    return render_template("add_tracker_log.html", tracker=tracker,
options=options)

            lg = Log(tid=tracker.tid, value=tval, notes=notes)
            db.session.add(lg)
            db.session.commit()
            return redirect(url_for("tracker_details", tid=tracker.tid))
        return render_template("add_tracker_log.html", tracker=tracker, options=options)
    else:
        flash("tracker not found!")
        return redirect("/")
else:
    return redirect("/login")


@app.route("/tracker/<int:t_id>/logs")
def get_all_tracker_logs(t_id):
    if "username" in session:
        tracker = Tracker.query.get(t_id)
        logs = Log.query.filter(Log.tid == tracker.tid).all()
        if tracker and logs:
```

```
            return render_template("get_all_tracker_logs.html", tracker=tracker, logs=logs)
        else:
            flash("something went wrong!")
            return redirect("/")
    else:
        return redirect("/login")

@app.route("/tracker/<int:t_id>/log/<int:l_id>")
def get_tracker_log(t_id, l_id):
    if "username" in session:
        tracker = Tracker.query.get(t_id)
        log = Log.query.get(l_id)
        if tracker and log:
            return render_template("get_tracker_log.html", tracker=tracker, log=log)
        else:
            flash("something went wrong!")
            return redirect("/")
    else:
        return redirect("/login")

@app.route("/tracker/<int:t_id>/log/<int:l_id>/edit", methods=['GET', 'POST'])
def edit_tracker_log(t_id, l_id):
    if "username" in session:
        tracker = Tracker.query.get(t_id)
        log = Log.query.get(l_id)
        if tracker and log:
            options = None
            if tracker.tracker_type == "Multiple Choice":
                options = [i.strip() for i in tracker.settings.split(",")]
```

```
        if request.method == 'POST':
            tval = request.form['tval']
            notes = request.form['tnotes']

            log.value = tval
            log.notes = notes
            db.session.add(log)
            db.session.commit()
            return redirect(url_for("tracker_details", tid=tracker.tid))
        return render_template("edit_tracker_log.html", tracker=tracker, log=log,
options=options)
    else:
        flash("something went wrong!")
        return redirect("/")
else:
    return redirect("/login")


@app.route("/tracker/<int:t_id>/log/<int:l_id>/delete")
def delete_tracker_log(t_id, l_id):
    if "username" in session:
        tracker = Tracker.query.get(t_id)
        log = Log.query.get(l_id)
        if tracker and log:
            db.session.delete(log)
            db.session.commit()
            return redirect(url_for('tracker_details', tid=tracker.tid))
        else:
            flash("something went wrong!")
            return redirect("/")
    else:
        return redirect("/login")
```

Config

```
import os

basedir = os.path.abspath(os.path.dirname(__file__))

class Config:
    DEBUG = False
    SQLITE_DB_DIR = None
    SQLALCHEMY_DATABASE_URI = None
    SQLALCHEMY_TRACK_MODIFICATIONS = False

class LocalDevelopmentConfig(Config):
    DEBUG = True
    SQLITE_DB_DIR = os.path.join(basedir, '../db_dir')
    SQLALCHEMY_DATABASE_URI = "sqlite:///" + os.path.join(SQLITE_DB_DIR,
'mydb.sqlite3')
    SQLALCHEMY_TRACK_MODIFICATIONS = False

class ProductionDevelopmentConfig(Config):
    DEBUG = False
    SQLITE_DB_DIR = os.path.join(basedir, '../db_dir')
    SQLALCHEMY_DATABASE_URI = "sqlite:///" + os.path.join(SQLITE_DB_DIR,
'mydb.sqlite3')
    SQLALCHEMY_TRACK_MODIFICATIONS = False
```

API Specification

```
import os

basedir = os.path.abspath(os.path.dirname(__file__))
```

```
class Config:
    DEBUG = False
    SQLITE_DB_DIR = None
    SQLALCHEMY_DATABASE_URI = None
    SQLALCHEMY_TRACK_MODIFICATIONS = False


class LocalDevelopmentConfig(Config):
    DEBUG = True
    SQLITE_DB_DIR = os.path.join(basedir, '../db_dir')
    SQLALCHEMY_DATABASE_URI = "sqlite:///" + os.path.join(SQLITE_DB_DIR,
'mydb.sqlite3')
    SQLALCHEMY_TRACK_MODIFICATIONS = False


class ProductionDevelopmentConfig(Config):
    DEBUG = False
    SQLITE_DB_DIR = os.path.join(basedir, '../db_dir')
    SQLALCHEMY_DATABASE_URI = "sqlite:///" + os.path.join(SQLITE_DB_DIR,
'mydb.sqlite3')
    SQLALCHEMY_TRACK_MODIFICATIONS = False
```

5. PERFORMANCE ANALYSIS

5.1 Testing and Implementing Testing

The manual testing has been done in the initial phases of the development. The software is working properly as it should be.

An Application testing tool is any program that helps QAs manage and regulate the test process. Deciding which application testing software or framework is to be used varies according to the nature of the application to be tested. Now, let's explore some of the most popular test automation frameworks used for application testing.

Selenium is the most popular tool suite for automating web application testing. It enables QAs to verify the cross browser compatibility of a web application using Selenium WebDriver. Rational Functional Tester (RFT) can be used as an alternative to Selenium.

Following are the fundamental steps involved in testing this application:

1. We created a test plan according to the application requirements
2. Developed manual test case scenarios from the end-users perspective
3. Automated the test scenarios using scripts
4. Performed functional tests and validated and everything worked according to requirements

Application testing is a significant stage in the software development life cycle. Consequently, it becomes necessary for every QA to understand the basics of application testing. This article attempts to foster this understanding so that QAs can do their job in the best possible way.

5.2 Testing Methods

There are two major type of testing they are

1) White Box Testing.
2) Black Box Testing.

White Box Testing

White box sometimes called “Glass box testing” is a test case design uses the control structure of the procedural design to drive test case.

Using white box testing methods, the following tests were made on the system

a) All independent paths within a module have been exercised once. In our system, ensuring that case was selected and executed checked all case structures. The bugs that were prevailing in some part of the code where fixed

b) All logical decisions were checked for the truth and falsity of the values.

Black box Testing

Black box testing focuses on the functional requirements of the software. This is black box testing enables the software engineering to derive a set of input conditions that will fully exercise all functional requirements for a program. Black box testing is not an alternative to white box testing rather it is complementary approach that is likely to uncover a different class of errors that white box methods like..

1) Interface errors

2) Performance in data structure

3) Performance errors

4) Initializing and termination errors

Unit testing

Unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. Ideally, each test case is independent from the others: substitutes like method stubs, objects, fakes and test harnesses can be used to assist testing a module in isolation.

Integration Testing:

This testing is sometimes called Integration and Testing. Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates and delivers as its output the integrated system ready for system testing.

Validation Testing:

Validation Testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that can reasonably expected by a customer. After validation test has been conducted, one of the following two possible conditions exists. The functions or performance characteristics confirm to specification and are accepted.

• In the administrator and marks modules, all the fields must be filled.

• In the student registration, mobile number should contain exactly 10 numbers.

User Acceptance Testing:

User acceptance of a system is a key factor of any system. The system under consideration is tested for the acceptance by constantly keeping in touch with the prospective system users at the same time of developing and marketing changes whenever required. This is done in regard to the following points:

• Input Screen Design

• Output Screen Design

6. CONCLUSION

6.1 Conclusion

Habit tracking webapp is a web app which helps to track users their daily things. This is a great application to the people who want to change their life by analysing it using track me. Track is user friendly, great interface, and safe to use application.

6.2 Future Scope

Moreover, the report gives detailed data about the major factors influencing the growth of the "track me" market at the national and local level forecast of the market size, in terms of value, market share by region, and segment, regional market positions, segment and country opportunities for growth, Key company profiles, SWOT, product portfolio and growth strategies. The future of this application will focus on the more detailed analysis of users daily life. It will help them a lot to improve their good habits and eliminate bad ones.

6.3 User Manual

i. Create account
ii. Login to the site
iii. Click on add tracker
iv. Fill the name, description, tracker type information in their sections and click add button.
v. Check the tracking history in 'tracker' section.
vi. Edit or delete the tracked information in 'action' section.
vii. Click log out when your work is done.

6.4 Proposed Enhancements

i. Graphics
ii. More features like voice notes recorder.

REFERENCES

We would like to mention some sources which proud to be helpful in making this presentation some of them are as follows:

Book-

1) Head first HTML and CSS

- Elisabeth Robson and Eric Freeman

2) "Effective PyCharm" by Michael Kennedy and Matt Harrison

Websites-

1) http://www.w3school.com
2) http://www.javatpont.com
3) http://www.geeksforgeeks.org

ANNEXURES:

ACKNOWLEDGEMENT

I like to share our sincere gratitude to all those who help us in completion of this project. During the work faced many challenges due to my lack of knowledge and experience but these people help us to get over from all the difficulties and in final compilation of our idea to a shaped sculpture.

I wish to express our deep sense of gratitude to our guide, Prof. M.D.Wangikar, Dir. S.D.Khamitkar (H.O.D of computational department SRTM University, Nanded) for his valuable guidance, keen interest, constructive suggestions and sustained encouragement throughout the project work.

I am also thankful to my whole class and to my parents who have inspired me to face all challenges and win all hurdles in life.

Potgante Ankush Baburao