**STUBBLE AGGREGATION AND DISPOSAL**

*A*

*Project Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

IN

**INFORMATION TECHNOLOGY**

By

**BASA RITHIKA (1602-19-737-088)**

**BEJUGAM SATHVIKA (1602-19-737-100)**

**VEMULA AKSHITHA (1602-19-737-307)**

*Under guidance of*

**KEZIA RANI AND HASEEBA YASEEN**

**Professor**

Diagram

Description automatically generated

**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University) Ibrahimbagh,**

**Hyderabad-31 2021-2022 Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University) Ibrahimbagh, Hyderabad-31**

**Department of Information Technology**



**DECLARATION BY THE CANDIDATES**

We, **B. RITHIKA, B. SATHVIKA, V. AKSHITHA** bearing hall ticket numbers, **1602-19-737-088, 1602-19-737-100, 1602-19-737-307** hereby declare that the project report entitled “**STUBBLE AGGREGATION AND DISPOSAL**” under the guidance of **KEZIA RANI**, **HASEEBA YASEEN** Professor, Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement of MINI PROJECT of V semester of **Bachelor of Engineering in Information Technology**.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**BASA RITHIKA (1602-19-737-088)**

**BEJUGAM SATHVIKA (1602-19-737-100)**

**VEMULA AKSHITHA (1602-19-737-307)**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University) Ibrahimbagh, Hyderabad-31**

**Department of Information Technology**

Diagram

Description automatically generated

**BONAFIDE CERTIFICATE**

This is to certify that the project entitled " **STUBBLE AGGREGATION AND DISPOSAL**” being submitted by **B. RITHIKA, B. SATHVIKA, V. AKSHITHA** bearing **1602-19-737-088, 1602-19-737-100, 1602-19-737-307** in partial fulfilment of the requirements for the completion of MINI PROJECT of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by them under my guidance.

**KEZIA RANI, HASEEBA YASEEN**

**Dr. K. Ram Mohan Rao**

**Professor HOD, IT**

**Internal Guide**

**ACKNOWLEDGEMENT**

The satisfaction that accompanies that the successful completion of the project would not have been possible without the kind support and help of many individuals. We would like to extend my sincere thanks to all of them. We would like to take the opportunity to express our humble gratitude to **Ms. Kezia Rani (Assistant Professor)** underwhom we executed this project.

We would also like to thank all faculty members and staff of the Department of Information Technology for their generous help in various ways for the completion of this project.

We are thankful to Mrs. Haseeba Yaseen for guiding us throughout.

Contents

[**CHAPTER 1** 8](#_Toc189056)

[**TITLE AND DESCRIPTION** 8](#_Toc189057)

[**1.1 Motivation**: 8](#_Toc189058)

[**CHAPTER 2** 9](#_Toc189059)

[**SOFTWARE REQUIREMENT SPECIFICATIONS** 9](#_Toc189060)

[**2.1 Introduction** 9](#_Toc189061)

[2.1.1 Description of Problem Statement 9](#_Toc189062)

[2.1.2 Scope: 9](#_Toc189063)

[2.1.3 Definitions, Acronyms and Abbreviations: 9](#_Toc189064)

[2.1.4 Overview: 9](#_Toc189065)

[**2.2 General description** 9](#_Toc189066)

[2.2.1 Product Perspective: 10](#_Toc189067)

[2.2.2 Product Functions: 11](#_Toc189068)

[2.2.3 User Characteristics 11](#_Toc189069)

[**2.3 Modules description: 12**](#_Toc189070)

[**2.4 System Requirements** 12](#_Toc189071)

[2.4.1 Hardware Requirements: 12](#_Toc189072)

[2.4.2 Software Requirement Specifications: 13](#_Toc189073)

[**2.5 Design Constraints**: 14](#_Toc189074)

[**CHAPTER 3** 15](#_Toc189079)

[**SYSTEM DESIGN** 15](#_Toc189080)

[3.1 Architecture and Technologies: 15](#_Toc189081)

[**3.2 UML Diagrams:** 16](#_Toc189082)

[**CHAPTER 4** 19](#_Toc189083)

[**IMPLEMENTATION CODE** 19](#_Toc189084)

[**4.1 SYSTEM ARCHITECTURE(DESIGN)** 19](#_Toc189085)

[**4.2 IMPLEMENTATION AND CODE** 23](#_Toc189086)

[4.2.1. PYTHON FILES 23](#_Toc189087)

[**4.2.2 HTML AND CSS** 25](#_Toc189088)

[**CHAPTER 5** 29](#_Toc189089)

[**RESULTS** 29](#_Toc189090)

[**GITHUB LINKS AND FOLDER STRUCTURE:** 35](#_Toc189091)

[**CHAPTER 6** 36](#_Toc189093)

[**TESTING** 36](#_Toc189094)

[**CHAPTER 7** 41](#_Toc189095)

[**CONCLUSION:** 41](#_Toc189096)

[**FUTURE SCOPE:** 42](#_Toc189097)

[**CHAPTER 8** 43](#_Toc189098)

[**REFERENCES** 43](#_Toc189099)

**ABSTRACT**

Stubble management is a web development project that helps farmers to sell their stubble and earn money. While most of the Aggrotech start-ups are focused either on supply chain, focus is now expanding to farm and yield management. The area that remains un-explored is the “stubble left after grain segregation”. As many of us think that stubble which occurs after grain segregation is not useful. So, to dispose the waste people usually burn stubble. But stubble burning is very dangerous as it releases lot of harmful gases which effects environment by causing pollution. Stubble burning is an alarming issue today in North India.

To overcome that situation, we came with an idea of automating the process of stubble collection and reuse of the stubble. For example, the stubble that come from rice grain is rich in fibre. So, by mixing a chemical with that stubble we can produce an item which is rich in fibre.

Stubble management is designed to make your cropping cycle more efficient and less costly. The benefits of this project are reducing harvest cost, reducing fuel consumption, eco-friendly environment.

7

# CHAPTER 1

## TITLE AND DESCRIPTION: -

“**Stubble Aggregation and Disposal**” is a web-based application where a farmer has stubble which gets after harvesting. Here, a farmer can sell his stubble by choosing the most suitable industry and near place where he want to sell with the objective of ‘no burning’. Firstly the farmer need to register and then he gets login access. Then he logs in and then searches the industries which are near to his place. Based on his farmers wish in all aspects i.e. ,price ,area the farmer going to sell his stubble to the respective industry. The website has photos, videos and all the information regarding the stubble and its disposal.

### 1.1 Motivation:

Many farmers think that stubble is not of use and proceeds to burn, so through our project we let them know that stubble which they burn can be used to make a product instead of causing pollution by burning.

# CHAPTER 2

## SOFTWARE REQUIREMENT SPECIFICATIONS:

### **2.1 Introduction**

#### 2.1.1 Description of Problem Statement**:**

Stubble management is a web development project that helps farmers to sell their stubble and earn money. The area that remains un-explored is the “stubble left after grain segregation”. As many of us think that stubble which occurs after grain segregation is not useful. So, to dispose the waste people usually burn stubble. But stubble burning is very dangerous as it releases lot of harmful gases which effects environment by causing pollution. Stubble burning is an alarming issue today in North India.

#### 2.1.2 Scope:

This idea of this web application is to automate the process of stubble collection and reuse of the stubble. The scope of this project is to gain a lot of help for farmers, by collecting all types of stubble. We then display of list of industries who are ready to buy the stubble. The motive of developing this application is to design make your cropping cycle more efficient and less costly and save environment from pollution.

#### 2.1.3 Definitions, Acronyms and Abbreviations:

HTML- Hyper Text Markup Language

CSS - Cascading Style Sheets

JS-JavaScript

2.1.4 Overview:

This document includes a brief description of our project. This chapter provides the requirements specification in detailed terms and a description of the different system interfaces. Description of the modules are also included.

The third chapter provides information about the work related to our project with examples.

The fourth chapter has the information of the technologies used and UML diagrams.

The fifth chapter includes implementation part with the coding part

The sixth chapter is included with the result screenshots of our project.

## 2.2 General description

#### 2.2.1 Product Perspective:

This system is a web application which will be used to find properties and view information about them and also to manage the new information.

This application needs the details of the user’s name, phone number, email-id for registration.

And name of stubble, quantity, date, industry name, address of industry and pickup point for stubble registration.

Since this is a data centric product it will need somewhere to store the data. For that a database is used. The web application will use the database to get data and to modify the data. But the modifications of the data by the user are restricted only to their properties. All the database communication will go over the Internet.

#### 2.2.2 Product Functions:

With the web application, the users will be able to search nearest industry for his stubble to sell. If a user is searching for industry, depending on what place included in the search a list of places along with industry names will be displayed. Each industry will have the detailed information. Any kind of modifications in the website are done only by the admin.

#### 2.2.3 User Characteristics**:**

There are users that interact with this system: users i.e., users and administrators.

**Users:**

* Users must register into the website and login if account already exists.
* In the search bar the user will enter the type of stubble he has and the place where he is.
* Then list of industries will display to him based on his requirements he go for stubble registration.
* If he is satisfied with certain industry then he goes for stubble registration by filling the columns of name of stubble, quantity, date, address to pick up and address of industry.

**Admin/Administrator:**

* The admin has to login to the website.
* The admin will verify the details of the industry posted by the user
* Add the verified industry into the database by which it will get available on the website.
* Has access to database and makes necessary modifications as requested by the user.
* Checks for proper working of the web application.

## 

## 2.3 Modules description:

* **Registration Module:** This module will help the user to register to the website.
* **Login Module:** The users can login to the website once they have successfully registered.
* **Type of Stubble Module:** The type of stubble that the person wants to sell.
* **Place:** This is the place of person from where he wants to sell.
* **Search Module:** The specifications applied while searching a property by the tenant will be applied and respective results are shown in a list.
* **Date**: displays the dates when the person wants to sell.
* **Quantity Module:** The amount of stubble that the person to sell.
* **Address to Pick up:** The place where stubble is available.
* **Address of industry:** The address of or place of industry where the person to sell.
* **Industry name:** The name of the industry to which a person wants to sell his stubble.

## 2.4 System Requirements

#### 2.4.1 Hardware Requirements:

* Computers with 1 GB RAM is required.
* A Computer with an internet connection and any internet browser is required for the client to run the web application.

#### 

#### 2.4.2 Software Requirement Specifications:

**Python 3.7: -**

Python 3.8 benefited from both new functionality and optimizations.

Python is the language used to build the flask framework. It is a dynamic scripting language like Perl and Ruby. The principal author of Python is Guido van Rossum. Python supports dynamic typing and has a garbage collector for automatic memory management. Another important feature of Python is dynamic name solution which binds the names of functions and variables during execution.

**Interpreter:**

Visual Studio code: It features a lightning-fast source code editor, perfect for day today use with support of hundreds of languages. It also helps to be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-section, snippets etc.

**Python-Flask:**

**Flask Framework:**

Django is a free and open-source web framework, written in Python, t’s has a small and easy-to-extend core: it’s a microframework that doesn’t include an ORM (Object Relational Manager) or such features. It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Poocco. Flask is based on the Werkzeg WSGI toolkit and the Jinja2 template engine.Both are Pocco projects.

The primary goal of flask to make the development of complex, data-based websites easier.

### WSGI

The Web Server Gateway Interface (Web Server Gateway Interface, WSGI) has been used as a standard for Python web application development. WSGI is the specification of a common interface between web servers and web applications.

### Werkzeug

Werkzeug is a WSGI toolkit that implements requests, response objects, and utility functions. This enables a web frame to be built on it. The Flask framework uses Werkzeg as one of its bases.

### jinja2

jinja2 is a popular template engine for Python.A web template system combines a template with a specific data source to render a dynamic web page.

## 2.5 Design Constraints:

**Software Constraints:** User can run the application either on LINUX or Windows with an internet connection and any internet browser.

**Hardware Constraints:** This system will run on a core processor with minimum of 1GB RAM.

**Acceptance criteria:** Before accepting, the developer must check whether the application is running properly or not and should also check whether the data is correctly sorted or not.

CHAPTER 3

## SYSTEM DESIGN

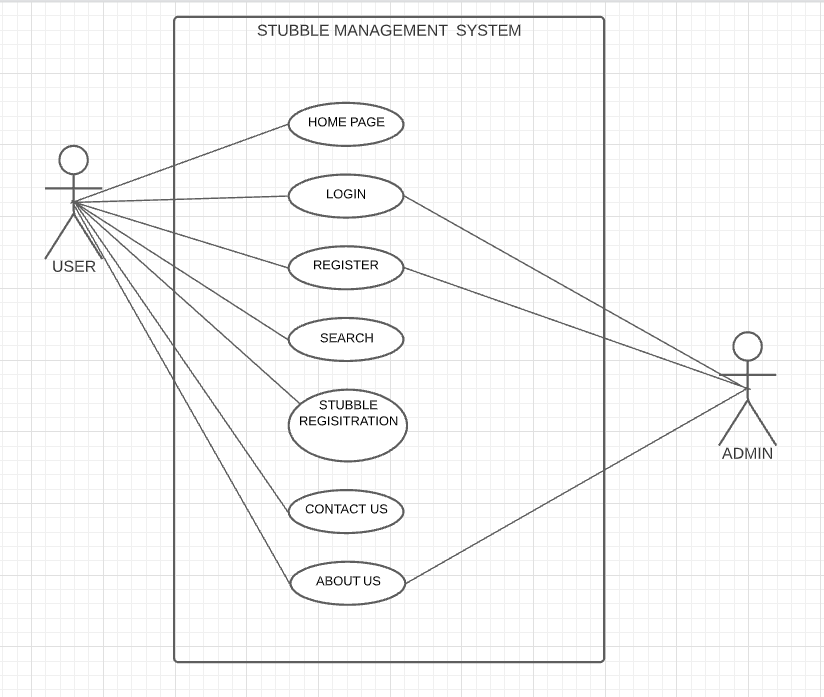
### 3.1 Architecture and Technologies:

**Front-end:**

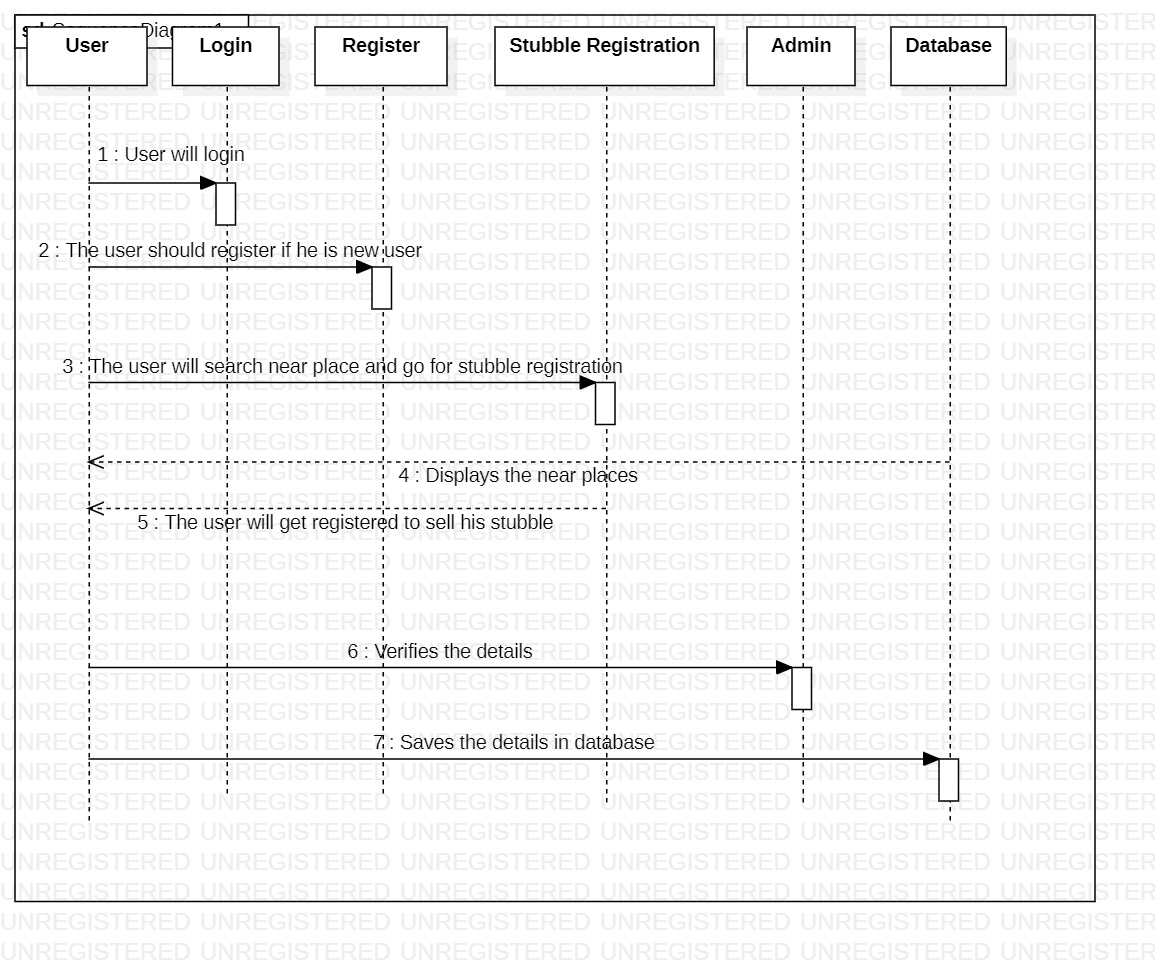
1. HTML
2. CSS
3. JavaScript
4. MySQL Backend**:** SqlLite3(For database)

3.2 UML Diagrams:

**USE CASE DIAGRAM:**



**SEQUENCE DIAGRAM:**



**CLASS DIAGRAM:**

Diagram

Description automatically generated

# 

# CHAPTER 4

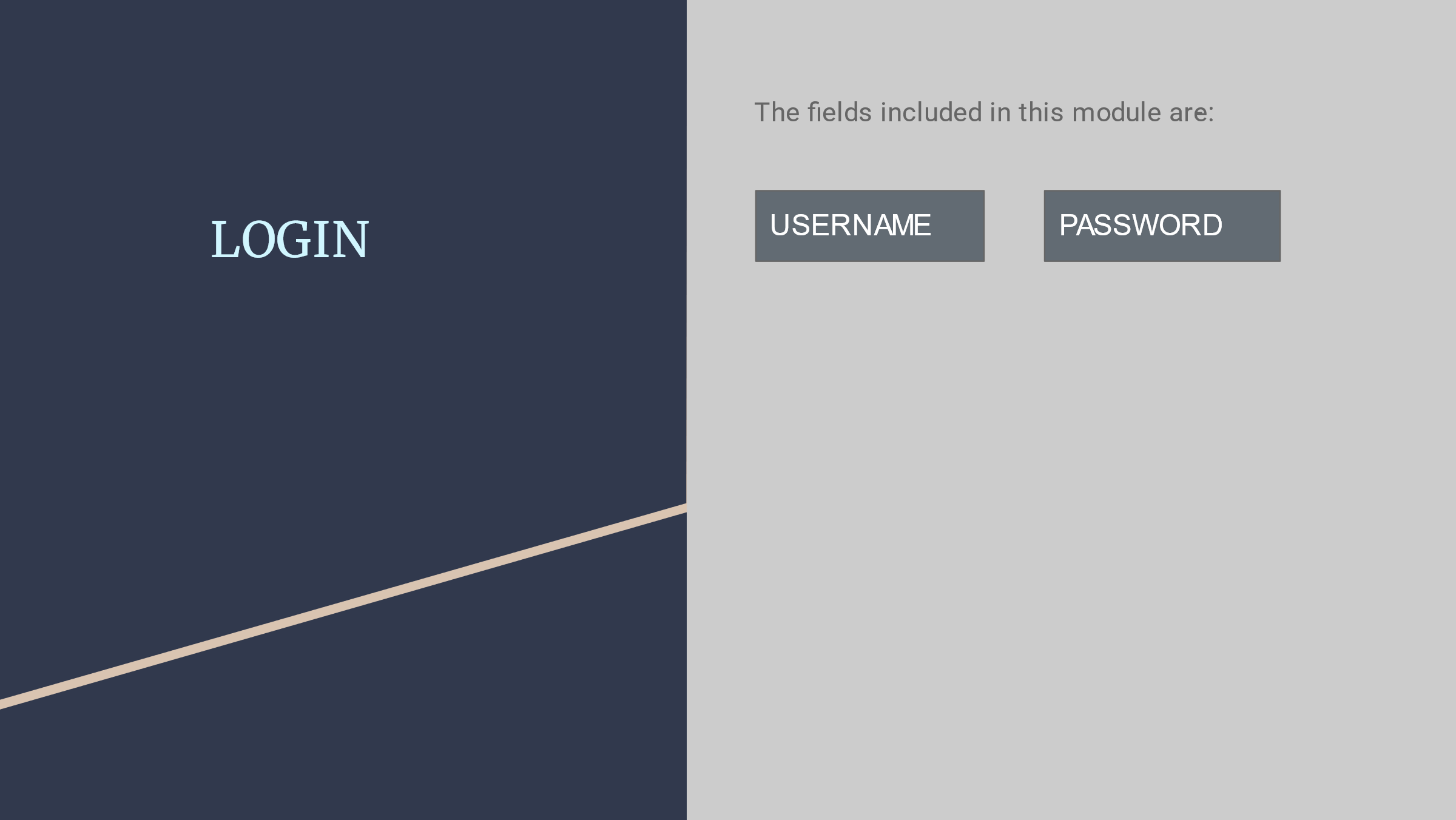
## IMPLEMENTATION OF CODE

### **5.1 SYSTEM ARCHITECTURE(DESIGN)**

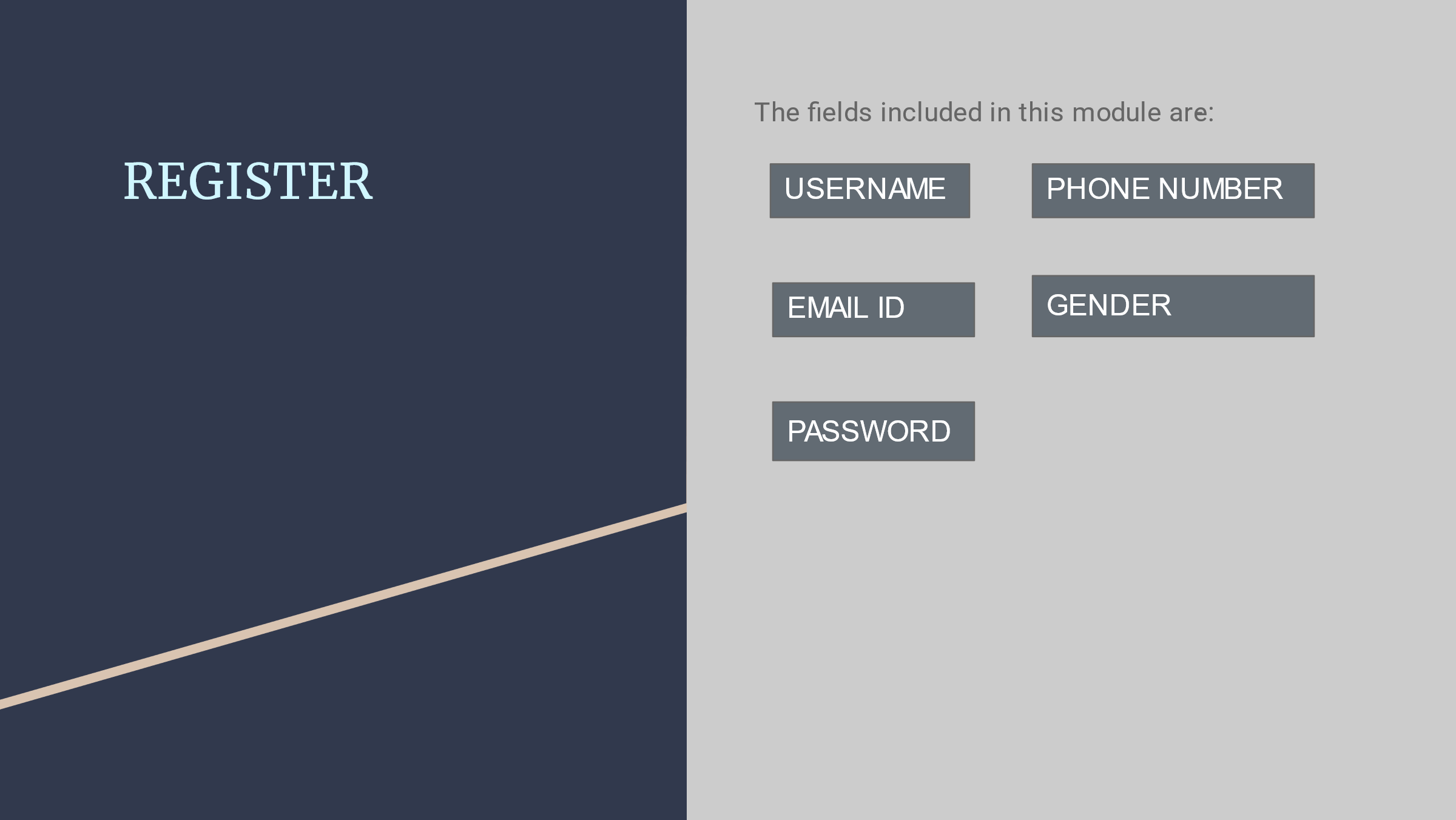
**Home Page:**



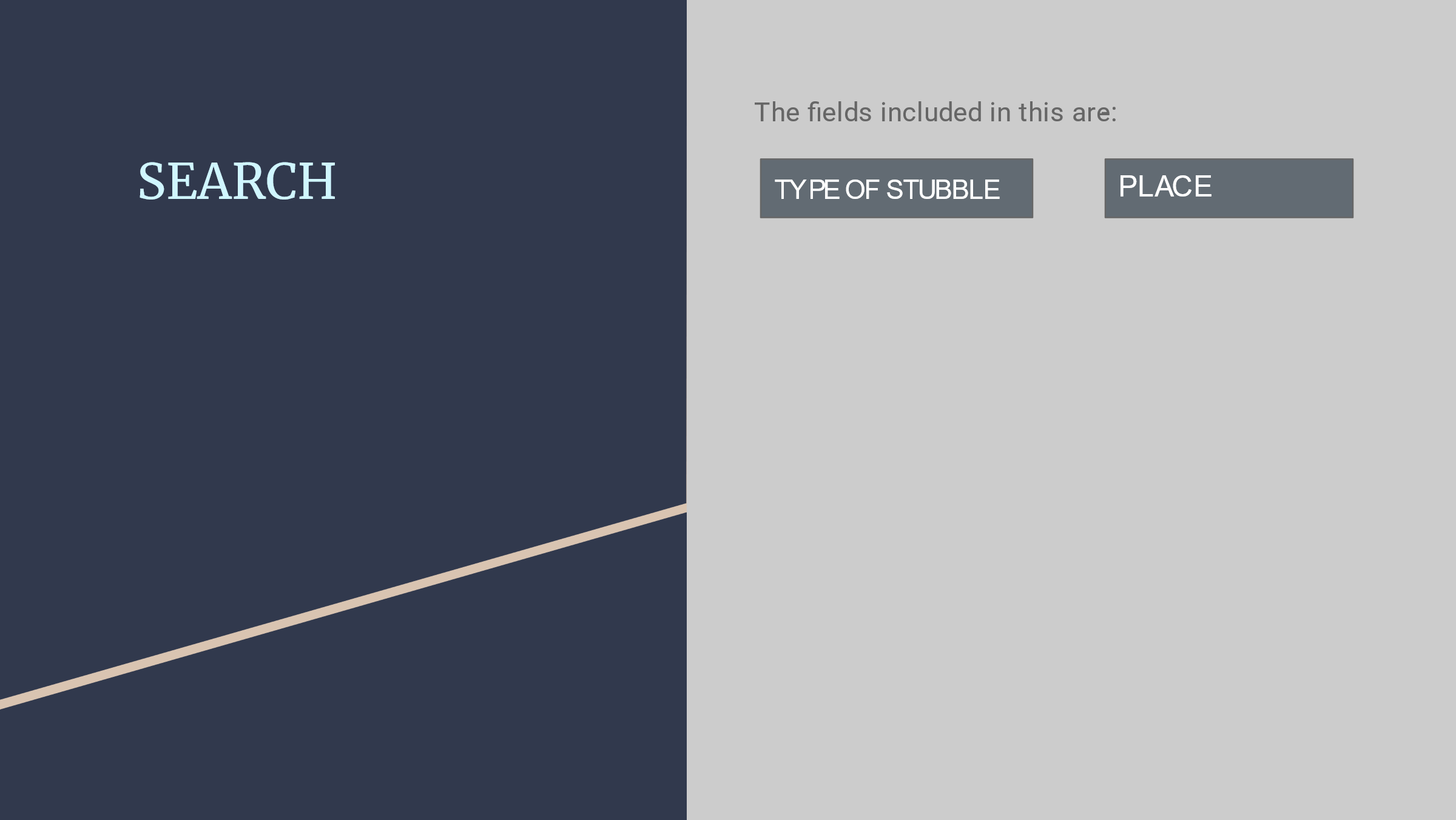
**Login Page:**



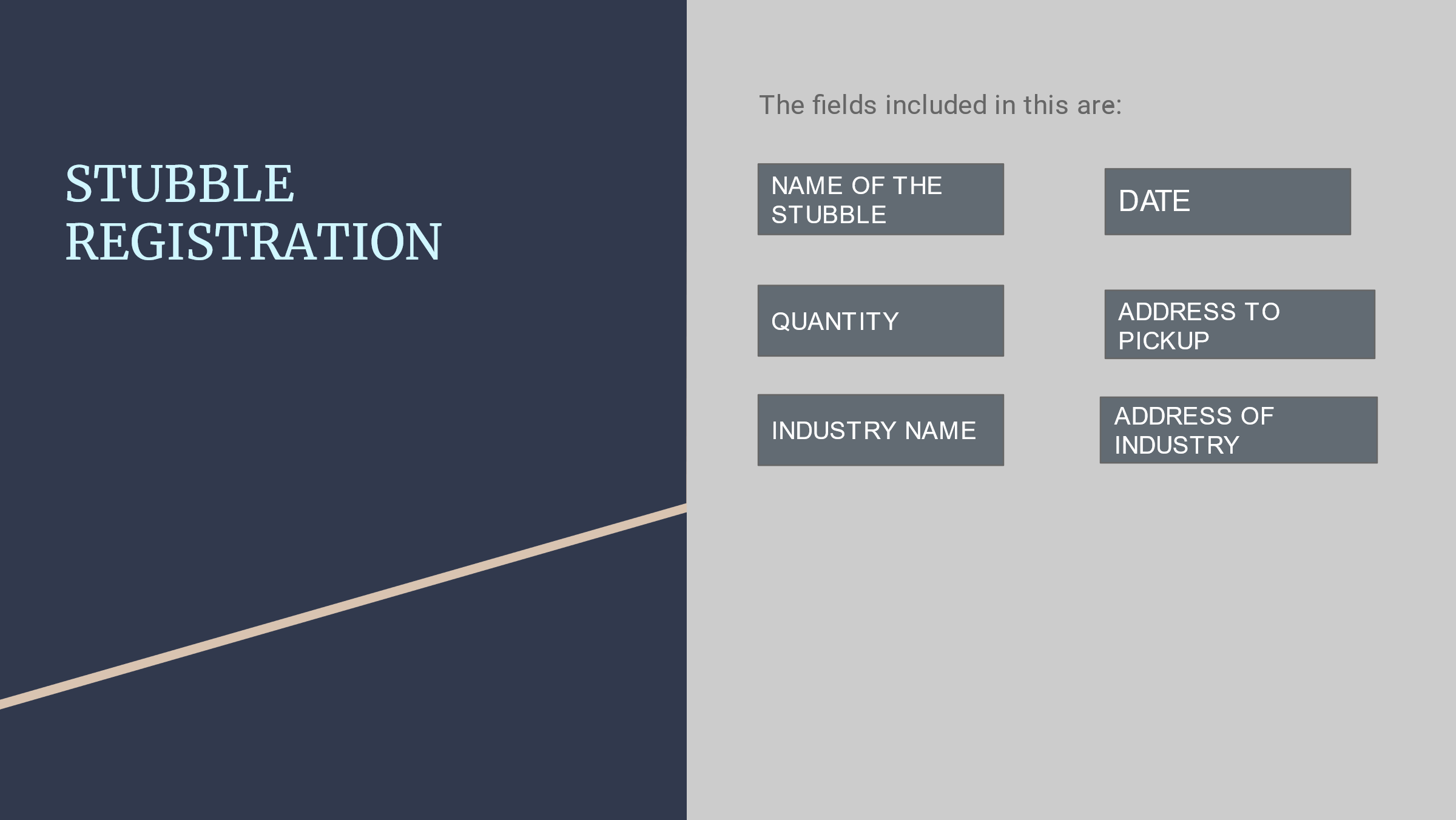
**Register Page:**



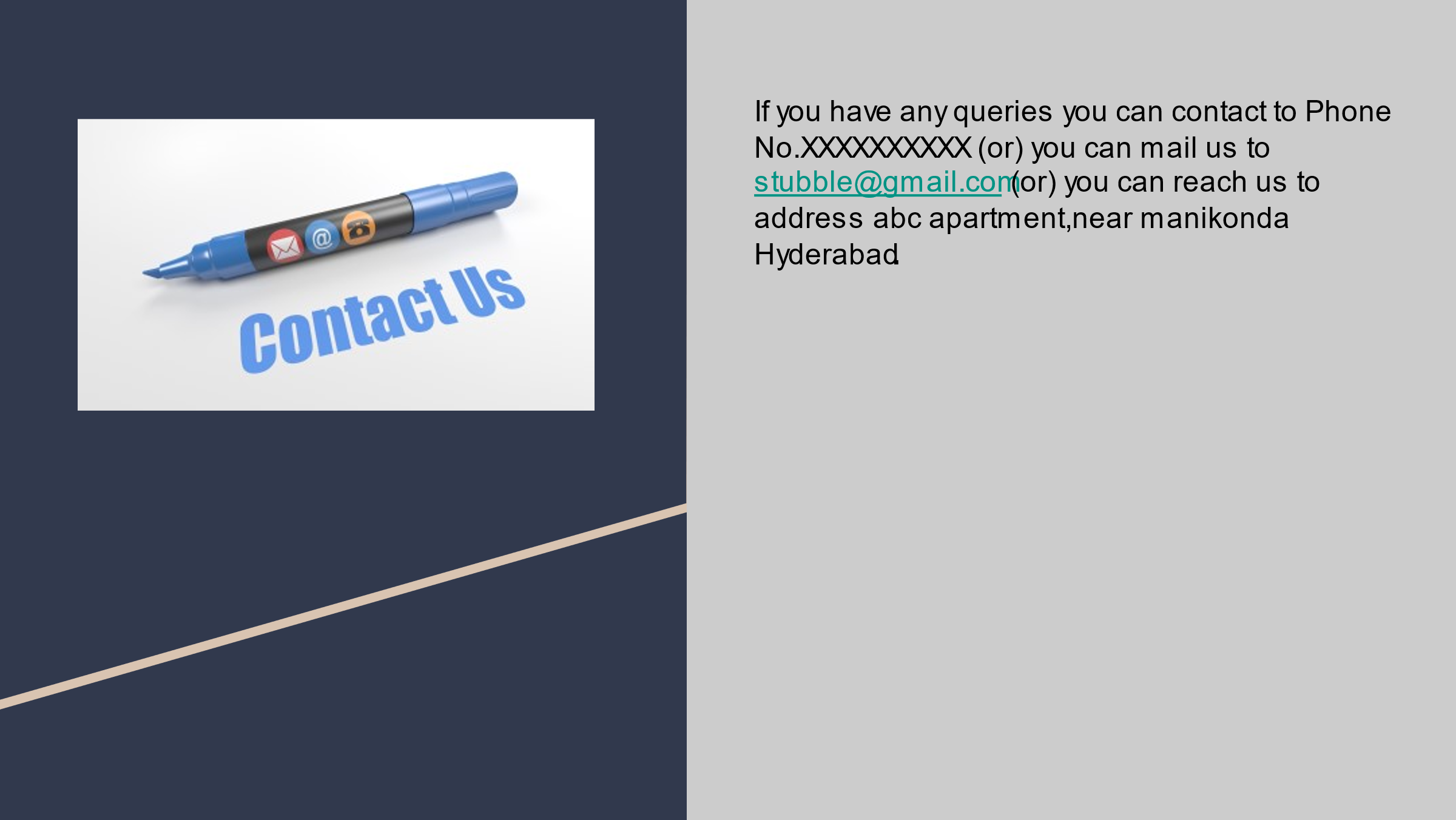
**Search:**



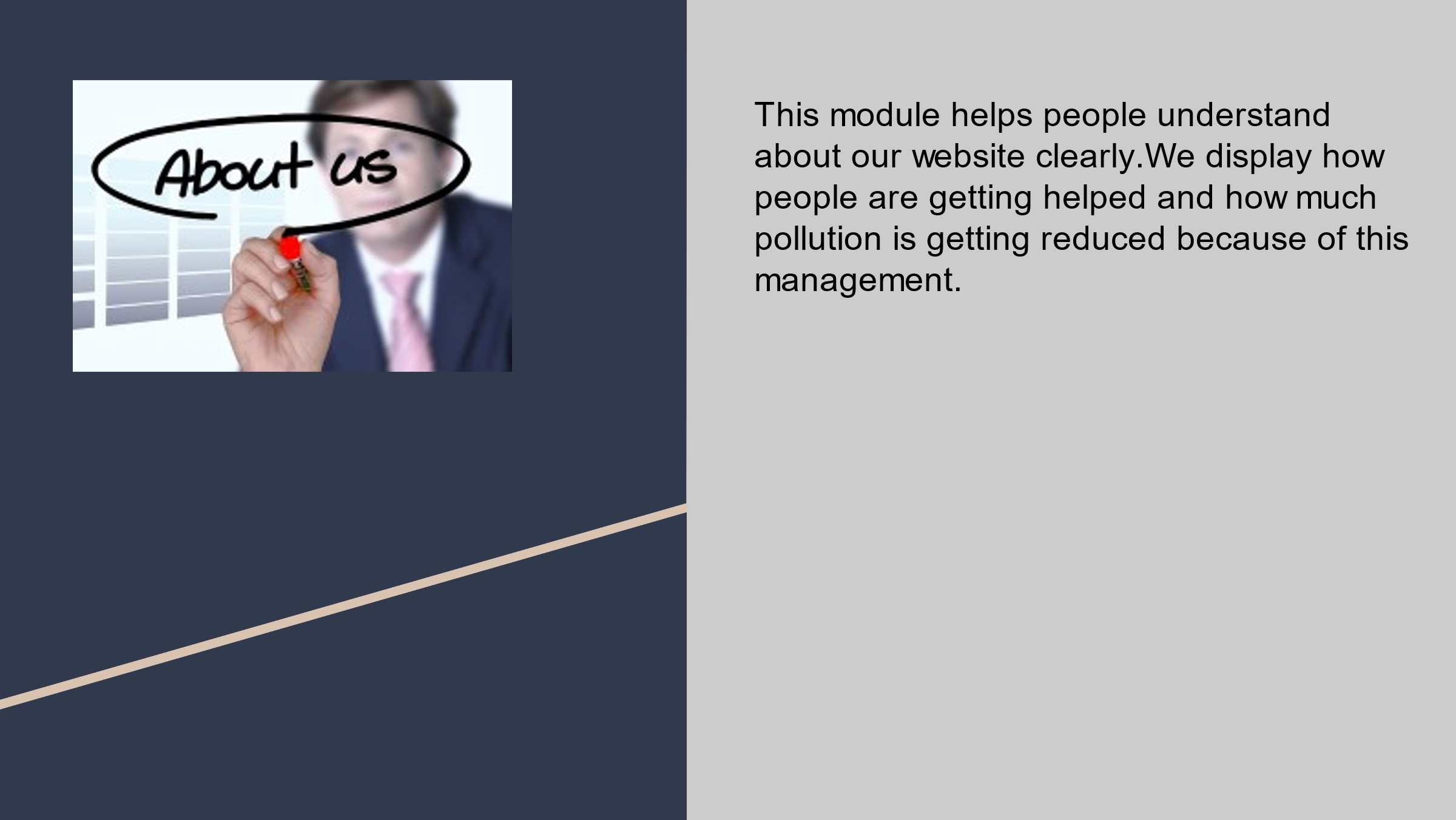
**Stubble Registration:**



**Contact Us:**



**About Us:**



## 

## 4.2 IMPLEMENTATION AND CODE

#### 4.2.1. PYTHON FILES

**main.py:**

from flask import \*  
from database import WorkspaceData  
app = Flask(\_\_name\_\_)  
  
@app.route('/',methods=['GET'])  
def home():  
 return render\_template("home.html")  
@app.route('/login',methods=['POST','GET'])  
def login():  
 return render\_template('login.html',retry=False, errorType='none')  
  
@app.route('/validate\_login', methods=['POST', 'GET'])  
def validate\_login():  
 if request.method == 'POST':  
 if 'login' in request.form:  
 user\_name = request.form.get('username')  
 password = request.form.get('password')  
 db = WorkspaceData()  
 data = db.get('login', user\_name)  
 for s in data:  
 if user\_name == s['username'] and password == s['password']:  
 return redirect('/frontp')  
 return render\_template('login.html', retry=True, errorType='login')  
 elif 'signup' in request.form:  
 user\_name = request.form.get('name')  
 password = request.form.get('password')  
 email = request.form.get('email')  
 db = WorkspaceData()  
 db.add('login', [(user\_name, password, email)])  
 return render\_template('frontpage.html', retry=False, errorType='signup')  
  
@app.route('/validate\_stubble', methods=['POST', 'GET'])  
def validate\_stubble():  
 if request.method == 'POST':  
 user\_name = request.form.get('name')  
 address = request.form.get('address')  
 sname=request.form.get('stubblename');  
 dmy=request.form.get('date');  
 quan = request.form.get('quantity');  
 mobile = request.form.get('phone');  
 iname = request.form.get('industryname');  
 iaddress = request.form.get('industryaddress');  
 db = WorkspaceData()  
 db.add('stubble', [(user\_name, address,sname,dmy,quan,mobile,iname,iaddress)])  
 return render\_template('registartion.html')  
  
@app.route('/signup',methods=['POST','GET'])  
def signup():  
 return render\_template('signup.html')  
  
@app.route('/frontp',methods=['POST','GET'])  
def frontp():  
 return render\_template("frontpage.html")  
  
@app.route('/contactus')  
def contactus():  
 return render\_template('ContactUs.html')  
  
@app.route('/aboutus')  
def aboutus():  
 return render\_template('Aboutus.html')  
@app.route('/search',methods=['POST','GET'])  
def search():  
 return render\_template('search.html',info\_data=0,length=0)  
  
@app.route('/registartion',methods=['POST','GET'])  
def registartion():  
 return render\_template('registartion.html')  
  
@app.route('/thankyou',methods=['POST','GET'])  
def thankyou():  
 return render\_template('thankyou.html')  
  
@app.route('/validate\_search', methods=['POST', 'GET'])  
def validate\_search():  
 if request.method == 'POST':  
 name = request.form.get('name')  
 place = request.form.get('place')  
 db = WorkspaceData()  
 data = db.get('industries', name)  
 info\_data = dict()  
 info\_data['industryname']=[]  
 info\_data['industryplace']=[]  
 info\_data['quantity']=[]  
 info\_data['price']=[]  
 print(place)  
 for s in data:  
 print(s[1])  
 if str(name) == str(s[0]) and str(place) == str(s[1]):  
 print(place)  
 info\_data['industryname'].append(s[2])  
 info\_data['industryplace'].append(s[3])  
 info\_data['quantity'].append(s[4])  
 info\_data['price'].append(s[5])  
 print(info\_data)  
 l=len(info\_data['industryname'])  
 return render\_template('search.html', info\_data=info\_data,length=l)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

**database.py**

import sqlite3  
import typing  
class WorkspaceData:  
 def \_\_init\_\_(self):  
 self.conn = sqlite3.connect('database.db')  
 self.conn.row\_factory = sqlite3.Row  
 self.cursor = self.conn.cursor()  
 self.cursor.execute("CREATE TABLE IF NOT EXISTS login(username TEXT PRIMARY KEY, password TEXT, email TEXT)")  
 self.cursor.execute("CREATE TABLE IF NOT EXISTS industries(username TEXT ,place TEXT,industryname TEXT,industryaddress TEXT,quantity TEXT,price REAl)")  
 self.cursor.execute("CREATE TABLE IF NOT EXISTS stubble(username TEXT,address TEXT,stubblename TEXT,date TEXT,quan TEXT,phonenumber REAL,industryname TEXT,industryaddress TEXT)")  
 self.conn.commit()  
 def add(self, table:str, data:typing.List[typing.Tuple]):  
 table\_data = self.cursor.execute(f"SELECT \* FROM {table}")  
 columns = [description[0] for description in table\_data.description]  
 sql\_statement = f"INSERT INTO {table} ({', '.join(columns)}) VALUES ({', '.join(['?']\*len(columns))})"  
 self.cursor.executemany(sql\_statement, data)  
 self.conn.commit()  
 def save(self, table: str, data: typing.List[typing.Tuple], username):  
 self.cursor.execute(f"DELETE FROM {table} WHERE USERNAME = '{username}'")  
  
 table\_data = self.cursor.execute(f"SELECT \* FROM {table}")  
  
 columns = [description[0] for description in table\_data.description]  
 sql\_statement = f"INSERT INTO {table} ({', '.join(columns)}) VALUES ({', '.join(['?']\*len(columns))})"  
  
 self.cursor.executemany(sql\_statement, data)  
 self.conn.commit()  
  
 def get(self, table:str,username:str) -> typing.List[sqlite3.Row]:  
 self.cursor.execute(f"SELECT \* FROM {table} WHERE USERNAME='{username}'")  
 data = self.cursor.fetchall()  
  
 return data  
  
#c = WorkspaceData()

#### **4.2.2 HTML AND CSS**

**login.html:**

1. <!DOCTYPE html>  
   <html style="font-size: 16px;">  
    <head>  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <meta charset="utf-8">  
    <meta name="keywords" content="Contact Form">  
    <meta name="description" content="">  
    <meta name="page\_type" content="np-template-header-footer-from-plugin">  
    <title>login</title>  
    <link rel="stylesheet" href="nicepage.css" media="screen">  
   <link rel="stylesheet" href="{{ url\_for('static',filename='nicepage.css')}}" media="screen">  
   <link rel="stylesheet" href="{{ url\_for('static',filename='login.css')}}" media="screen">  
    <script class="u-script" type="text/javascript" src="{{ url\_for('static',filename='jquery.js')}}" defer=""></script>  
    <script class="u-script" type="text/javascript" src="{{ url\_for('static',filename='nicepage.js')}}" defer=""></script>  
    <meta name="generator" content="Nicepage 4.0.3, nicepage.com">  
    <link id="u-theme-google-font" rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:100,100i,300,300i,400,400i,500,500i,700,700i,900,900i|Open+Sans:300,300i,400,400i,600,600i,700,700i,800,800i">  
     
     
     
    <script type="application/ld+json">{  
    "@context": "http://schema.org",  
    "@type": "Organization",  
    "name": ""  
   }</script>  
    <meta name="theme-color" content="#478ac9">  
    <meta property="og:title" content="login">  
    <meta property="og:type" content="website">  
    </head>  
    <body class="u-body">  
    <section class="u-clearfix u-section-1" id="sec-6a6c">  
    <div class="u-absolute-hcenter u-expanded u-palette-5-light-1 u-shape u-shape-rectangle"></div>  
    <div class="u-form u-login-control u-form-1">  
    <form action="/validate\_login" method="POST" class="u-clearfix u-form-custom-backend u-form-spacing-10 u-form-vertical u-inner-form" source="custom" name="form" style="padding: 10px;">  
    <div class="u-form-group u-form-name">  
    <label for="username-a30d" class="u-label">Username \*</label>  
    <input type="text" placeholder="Enter your Username" id="username-a30d" name="username" class="u-grey-5 u-input u-input-rectangle u-input-1" required="">  
    </div>  
    <div class="u-form-group u-form-password">  
    <label for="password-a30d" class="u-label">Password \*</label>  
    <input type="password" placeholder="Enter your Password" id="password-a30d" name="password" class="u-grey-5 u-input u-input-rectangle u-input-2" required="">  
    </div>  
    <div class="u-form-checkbox u-form-group">  
    <input type="checkbox" id="checkbox-a30d" name="remember" value="On">  
    <label for="checkbox-a30d" class="u-label">Remember me</label>  
    </div>  
    <div class="u-align-left u-form-group u-form-submit">  
    <input class="u-btn u-btn-submit u-button-style u-btn-1" type="submit" value="submit" >  
    </div>  
    <input type="hidden" value="" name="recaptchaResponse">  
    <input type="hidden" value="" name="login" id="login">  
    </form>  
    </div>  
    <a href="#" class="u-border-1 u-border-active-palette-2-base u-border-hover-palette-1-base u-btn u-button-style u-login-control u-login-forgot-password u-none u-text-palette-1-base u-btn-2">Forgot password?</a>  
    <a href="{{ url\_for('signup')}}" class="u-border-1 u-border-active-palette-2-base u-border-hover-palette-1-base u-btn u-button-style u-login-control u-login-create-account u-none u-text-palette-1-base u-btn-3">Don't have an account?</a>  
    </section>  
    <footer class="u-align-center u-clearfix u-footer u-grey-80 u-footer" id="sec-213c"><div class="u-clearfix u-sheet u-sheet-1">  
    <p class="u-small-text u-text u-text-variant u-text-1">Copyright 2021 Stubble Aggregation and Disposal.All rights reserved.</p>  
    </div>  
    </footer>  
    </body>  
   </html>

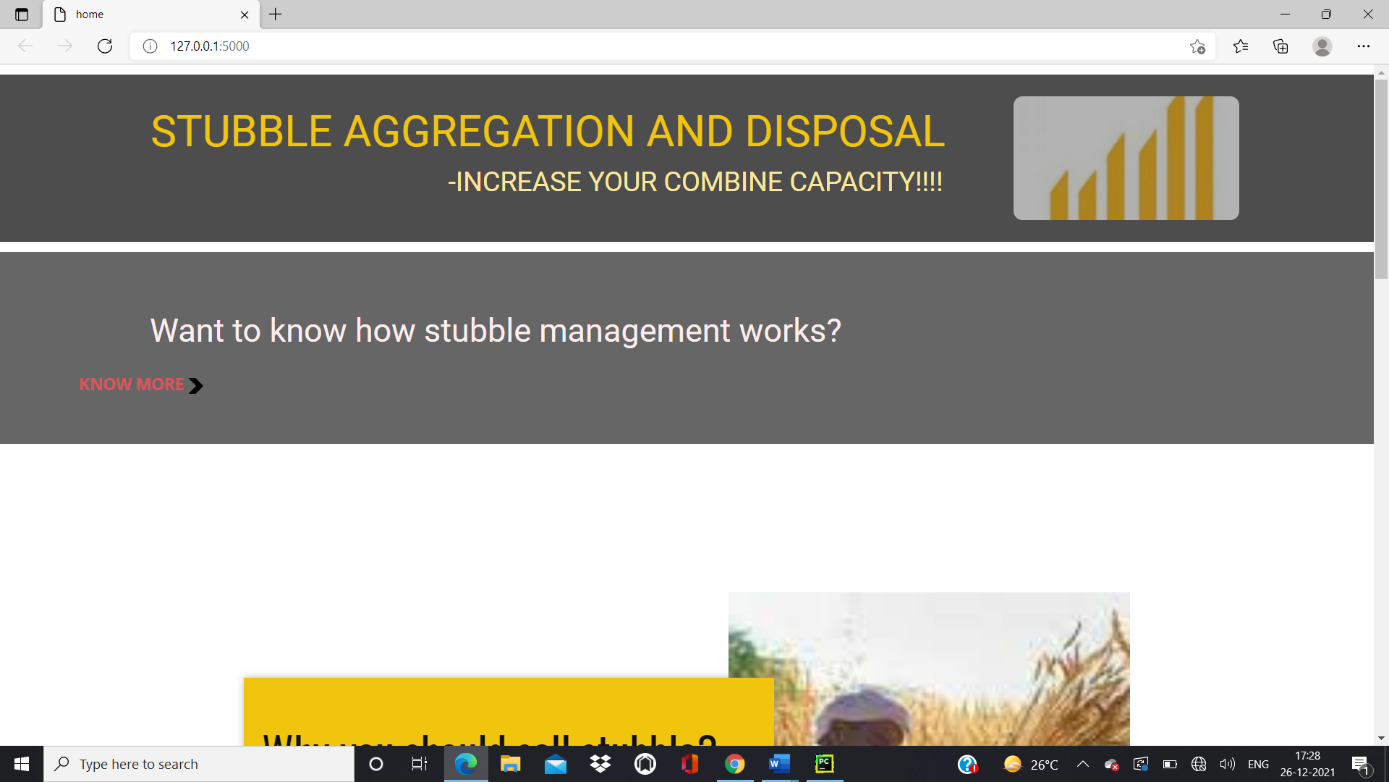
**Login.css:**

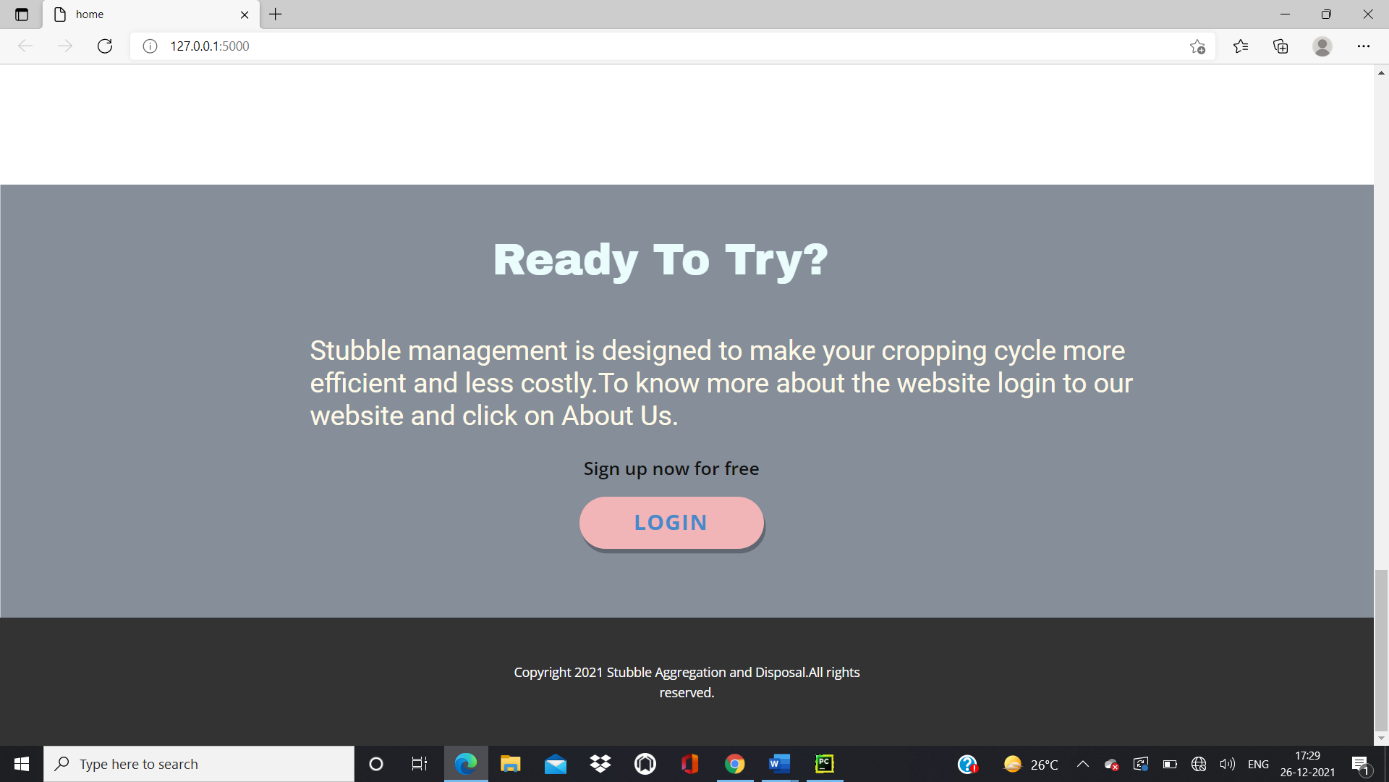
.u-section-1 {  
 min-height: 500px;  
}  
  
.u-section-1 .u-form-1 {  
 width: 570px;  
 height: 281px;  
 margin: 103px auto 0;  
}  
  
.u-section-1 .u-input-1 {  
 background-image: none;  
}  
  
.u-section-1 .u-input-2 {  
 background-image: none;  
}  
  
.u-section-1 .u-btn-1 {  
 width: 100%;  
 padding: 10px 30px;  
}  
  
.u-section-1 .u-btn-2 {  
 border-style: none none solid;  
 align-self: center;  
 margin: 1px auto 0 calc(((100% - 1140px) / 2) + 300px);  
 padding: 0;  
}  
  
.u-section-1 .u-btn-3 {  
 border-style: none none solid;  
 margin: -25px calc(((100% - 1140px) / 2) + 304px) 60px auto;  
 padding: 0;  
}  
  
@media (max-width: 1199px) {  
 .u-section-1 .u-btn-2 {  
 margin-left: calc(((100% - 940px) / 2) + 100px);  
 }  
  
 .u-section-1 .u-btn-3 {  
 margin-right: calc(((100% - 940px) / 2) + 104px);  
 }  
}  
  
@media (max-width: 991px) {  
 .u-section-1 .u-btn-2 {  
 margin-left: calc(((100% - 720px) / 2));  
 }  
  
 .u-section-1 .u-btn-3 {  
 margin-right: calc(((100% - 720px) / 2));  
 }  
}  
  
@media (max-width: 767px) {  
 .u-section-1 .u-form-1 {  
 width: 540px;  
 }  
  
 .u-section-1 .u-btn-2 {  
 margin-left: calc(((100% - 540px) / 2));  
 }  
  
 .u-section-1 .u-btn-3 {  
 margin-right: calc(((100% - 540px) / 2));  
 }  
}  
  
@media (max-width: 575px) {  
 .u-section-1 .u-form-1 {  
 width: 340px;  
 }  
  
 .u-section-1 .u-btn-2 {  
 margin-left: calc(((100% - 340px) / 2));  
 }  
  
 .u-section-1 .u-btn-3 {  
 margin-right: calc(((100% - 340px) / 2));  
 }  
}

# CHAPTER 5

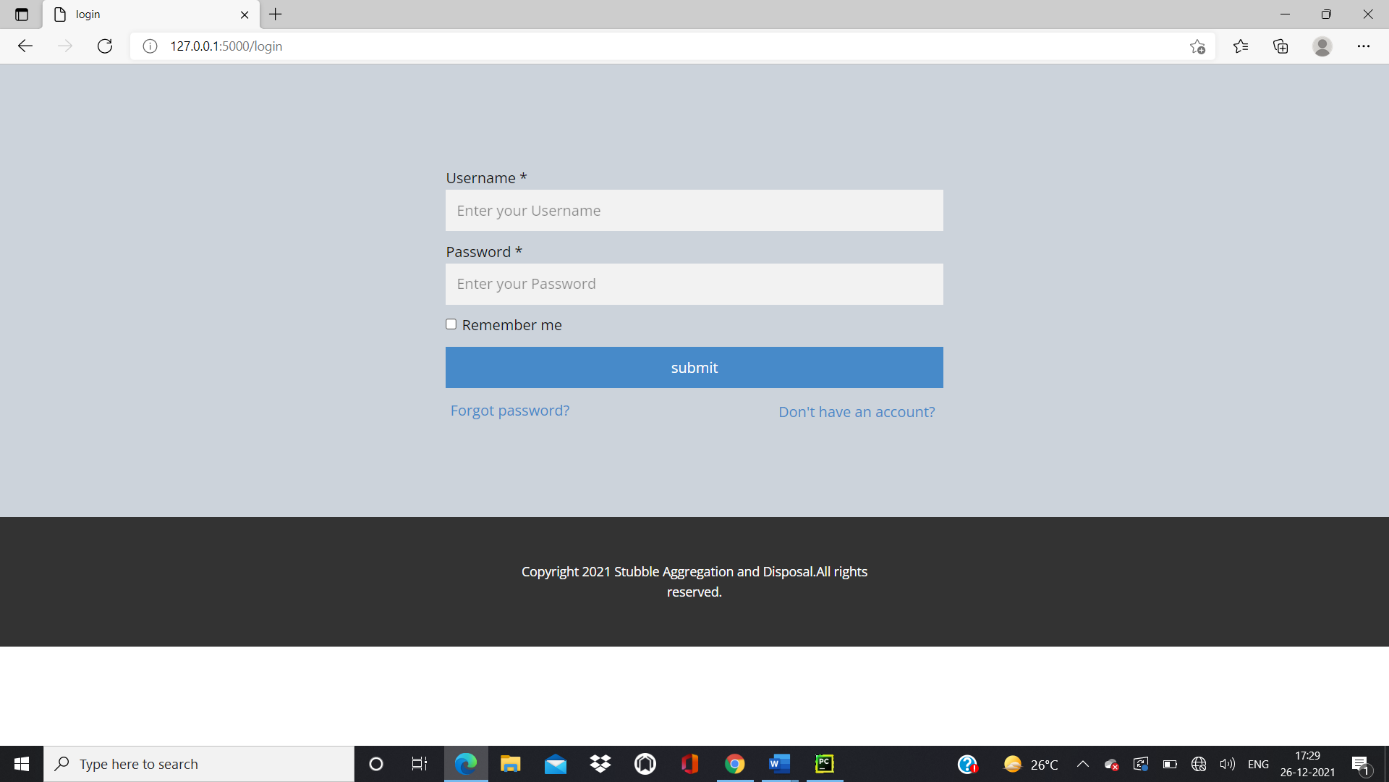
## RESULTS

HOME PAGE: -

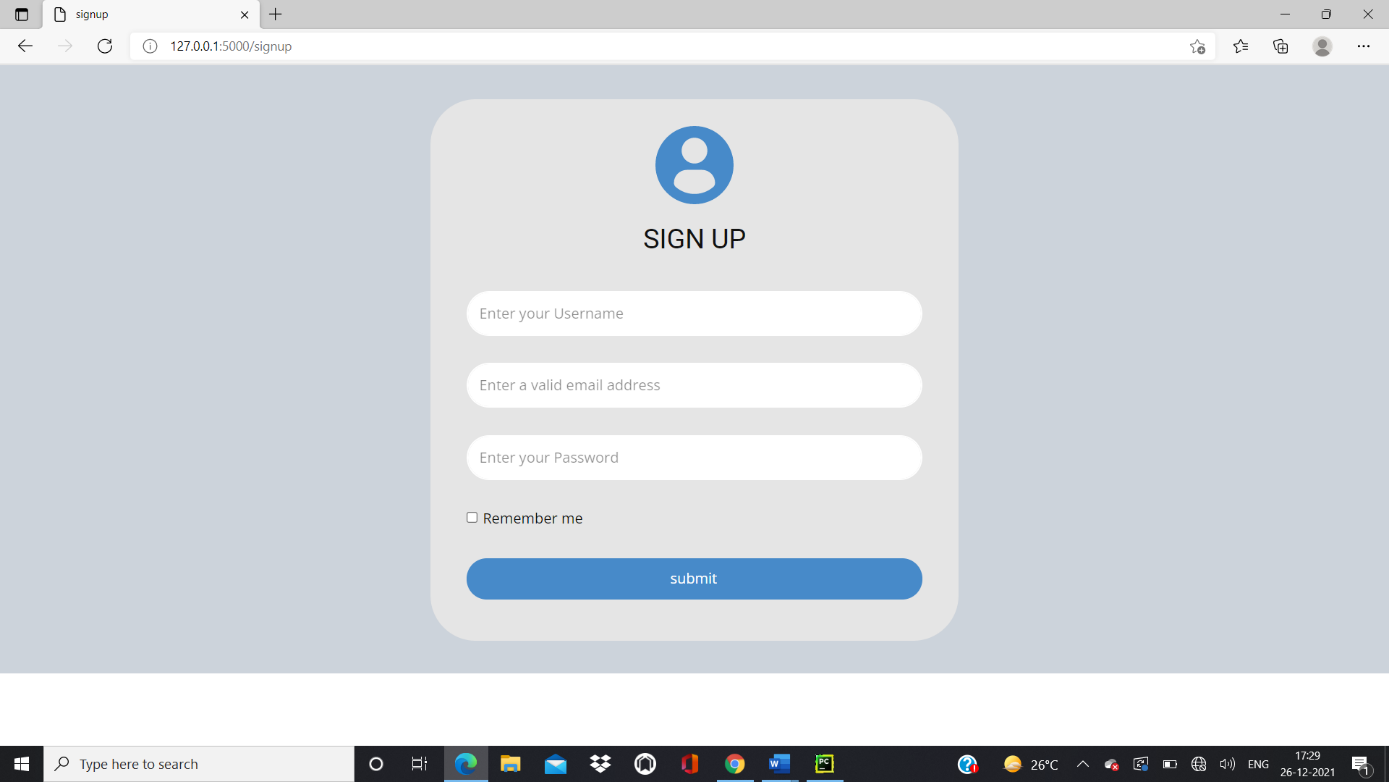




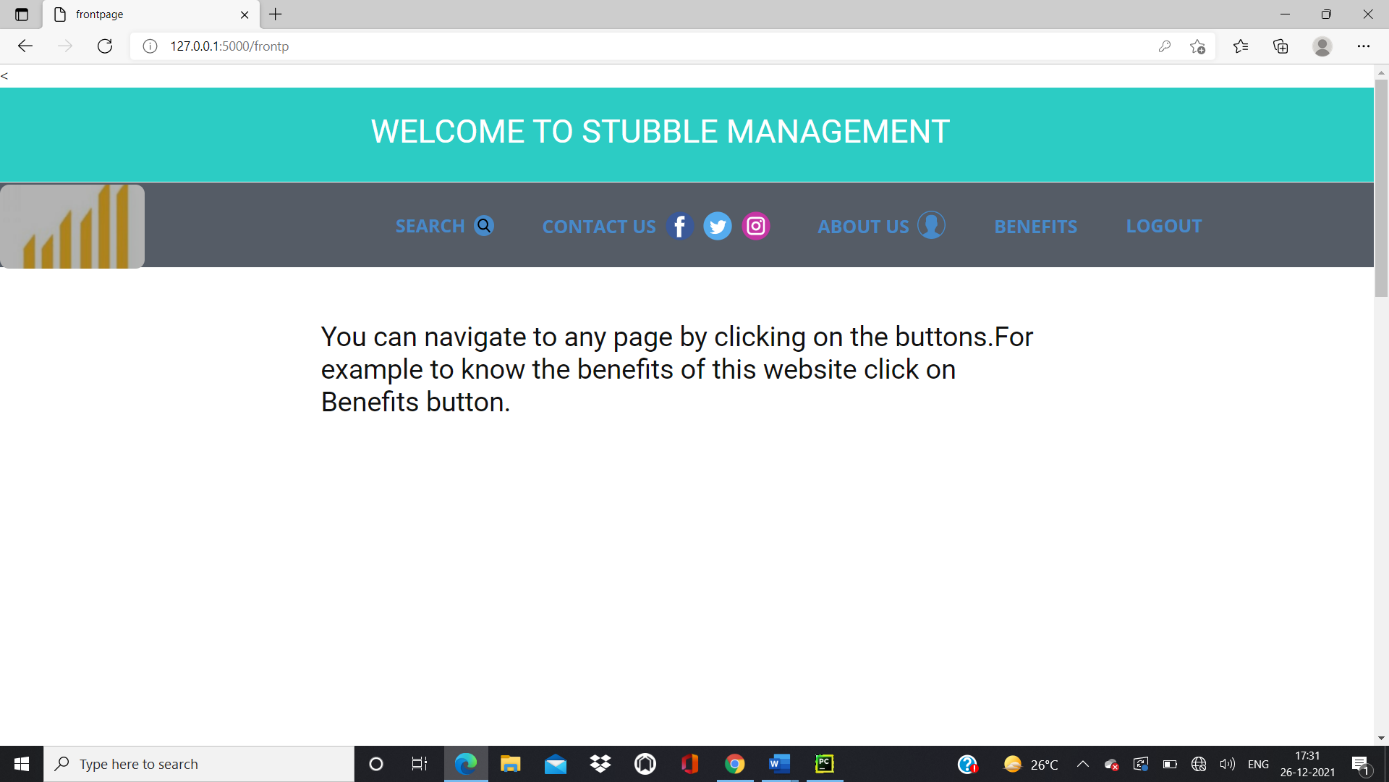
LOGIN PAGE: -



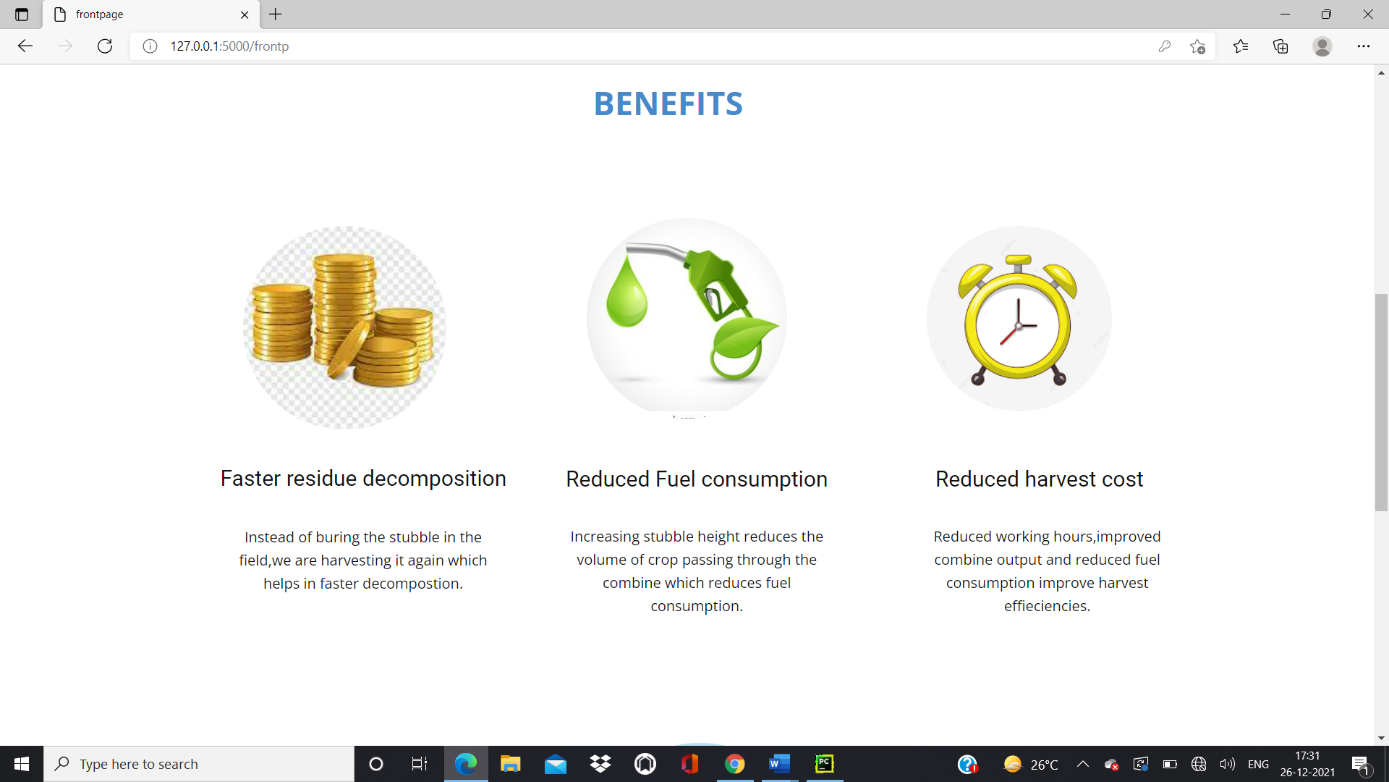
SIGNUP PAGE: -



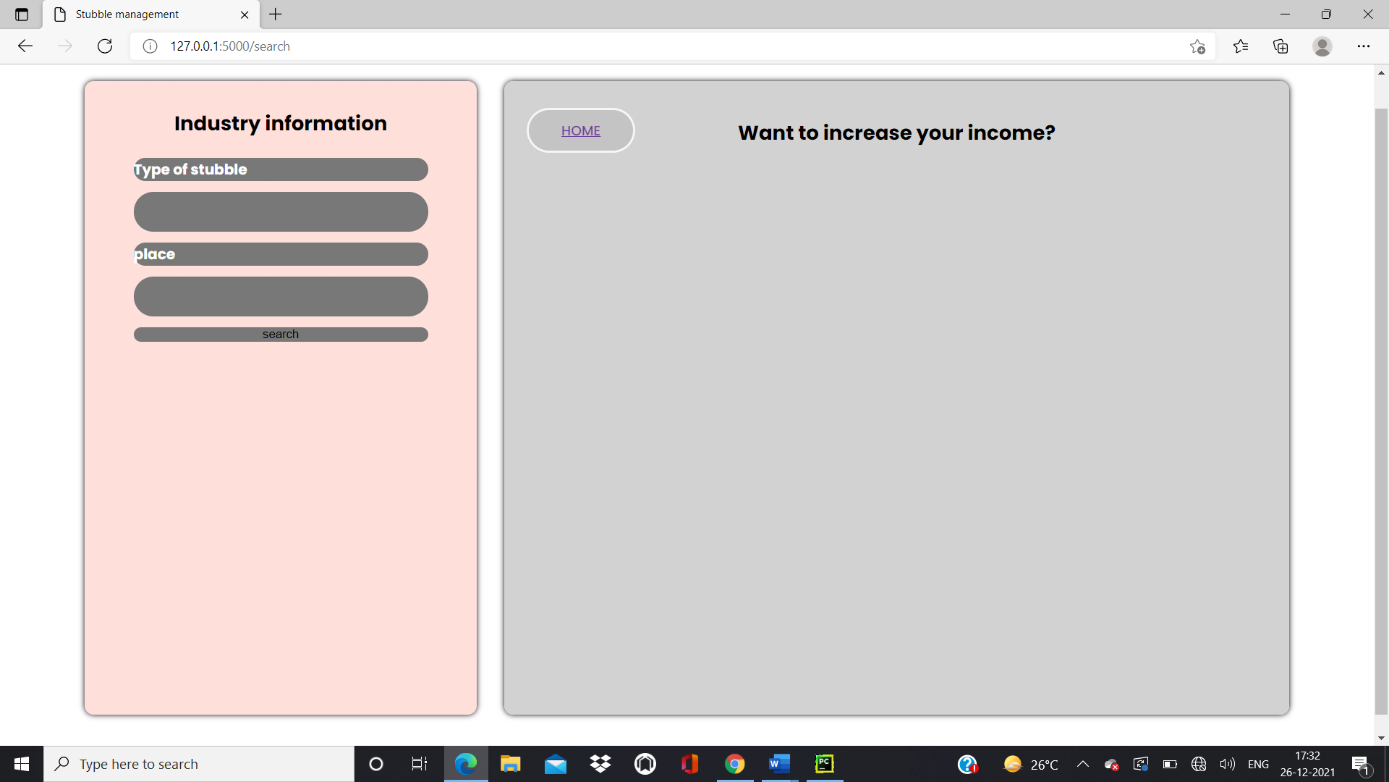
FRONTPAGE: -



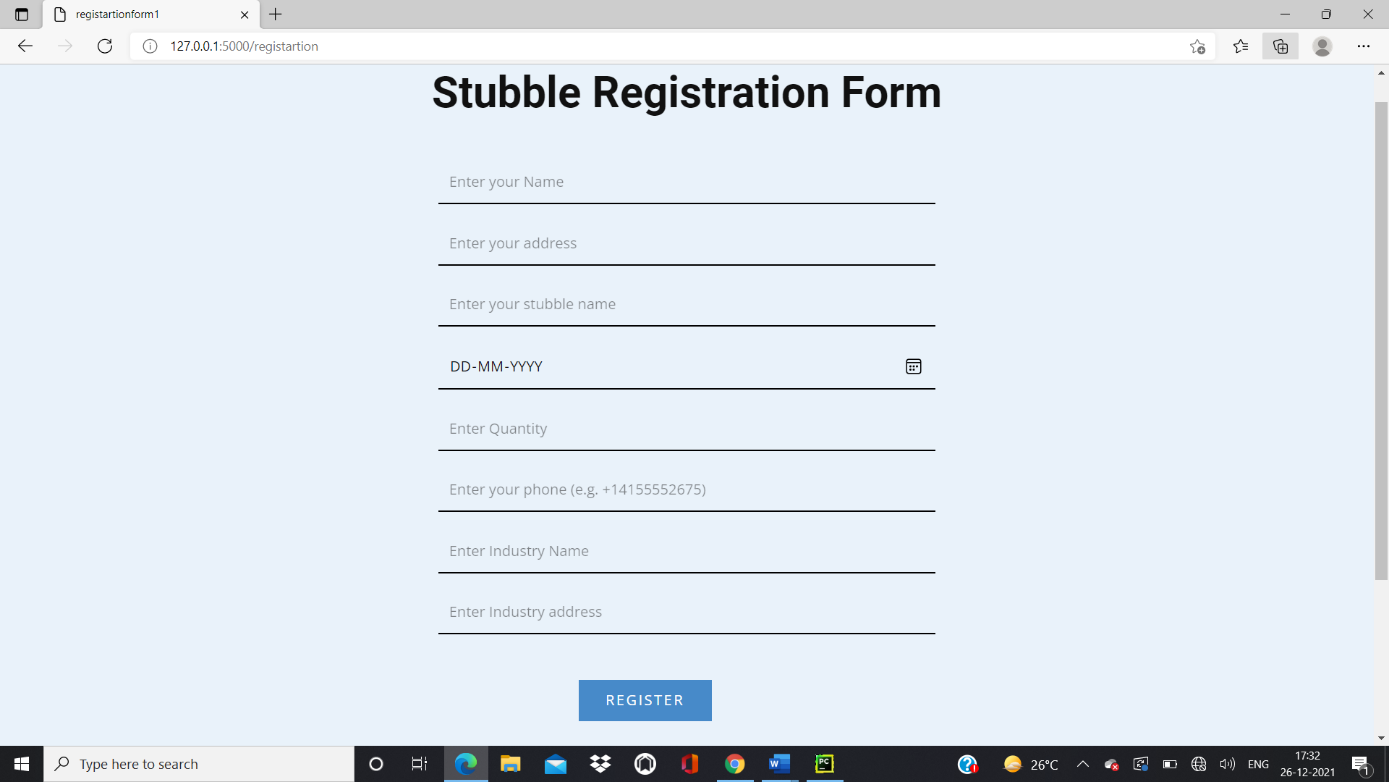
BENEFITS: -



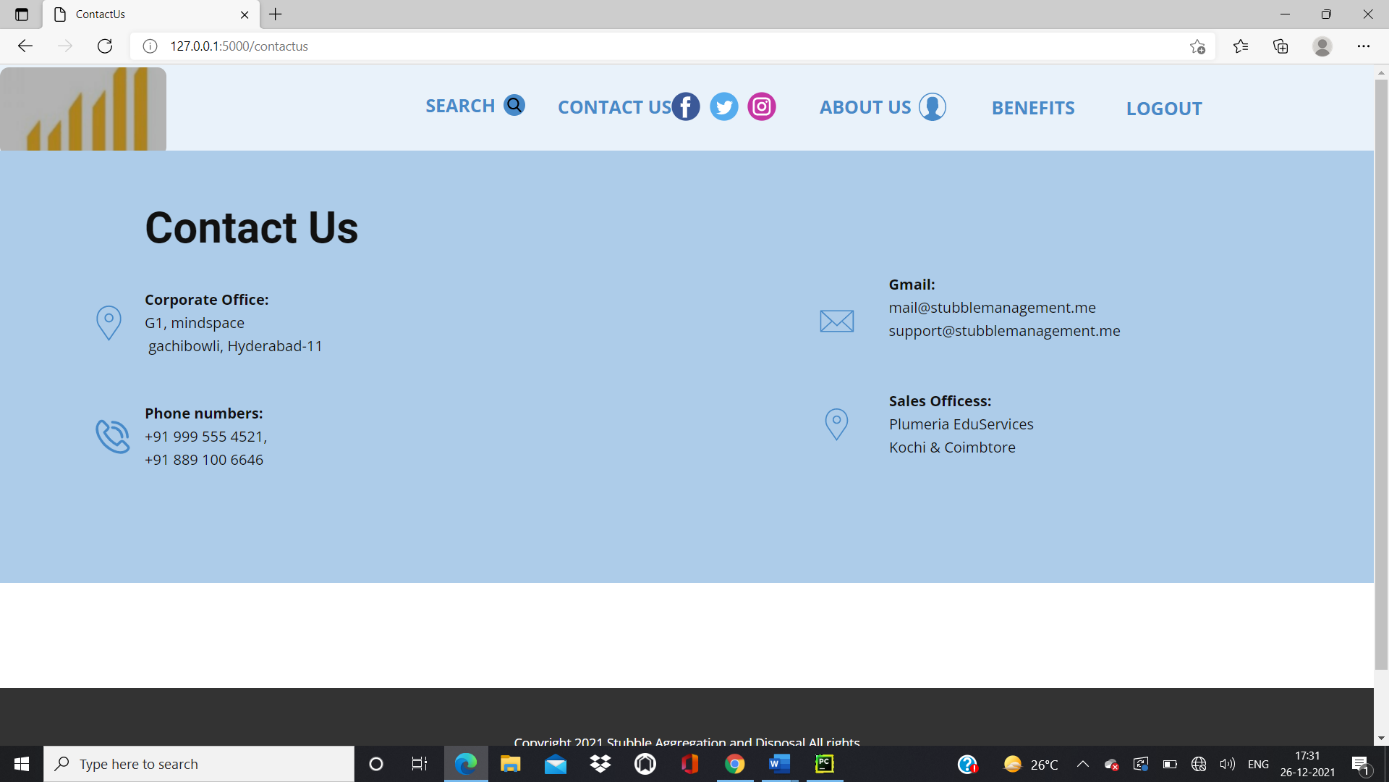
SEARCH: -



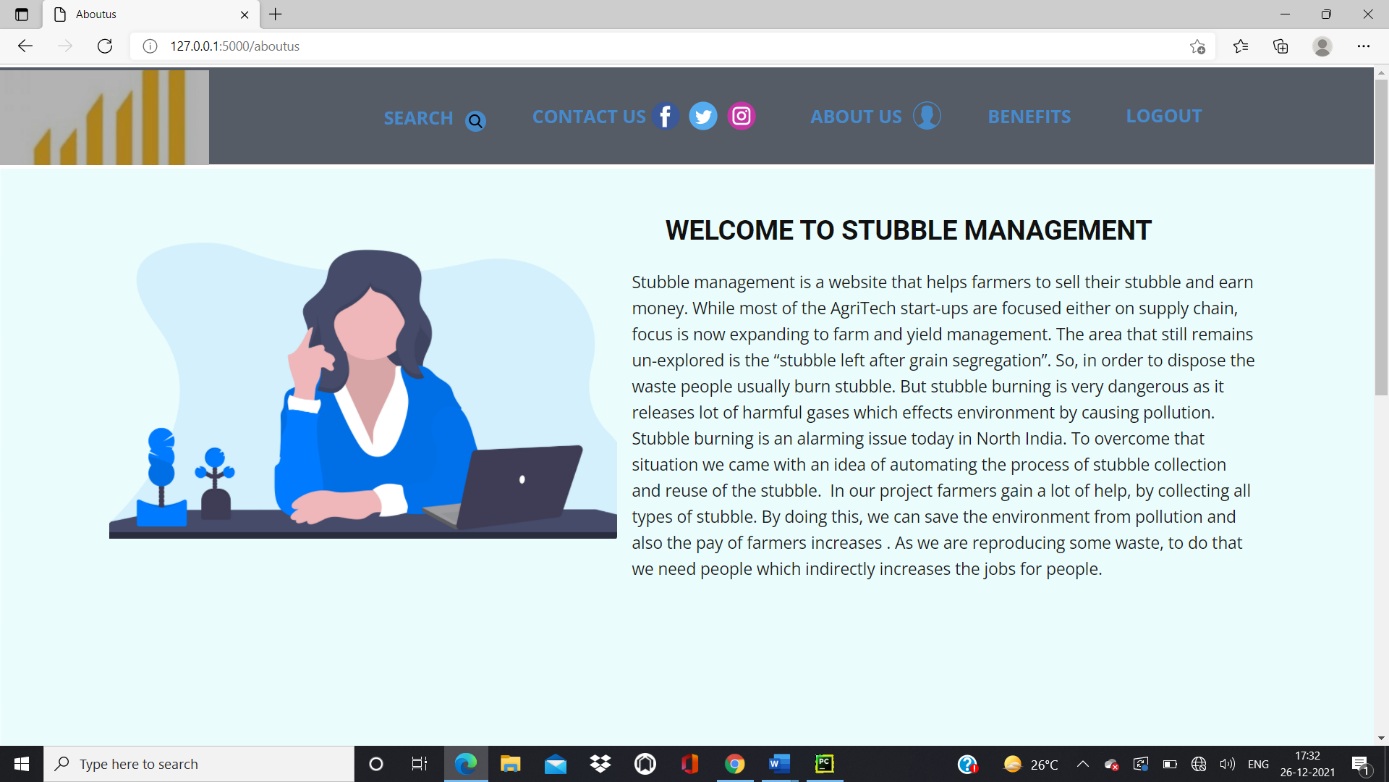
STUBBLE REGISTRATION: -

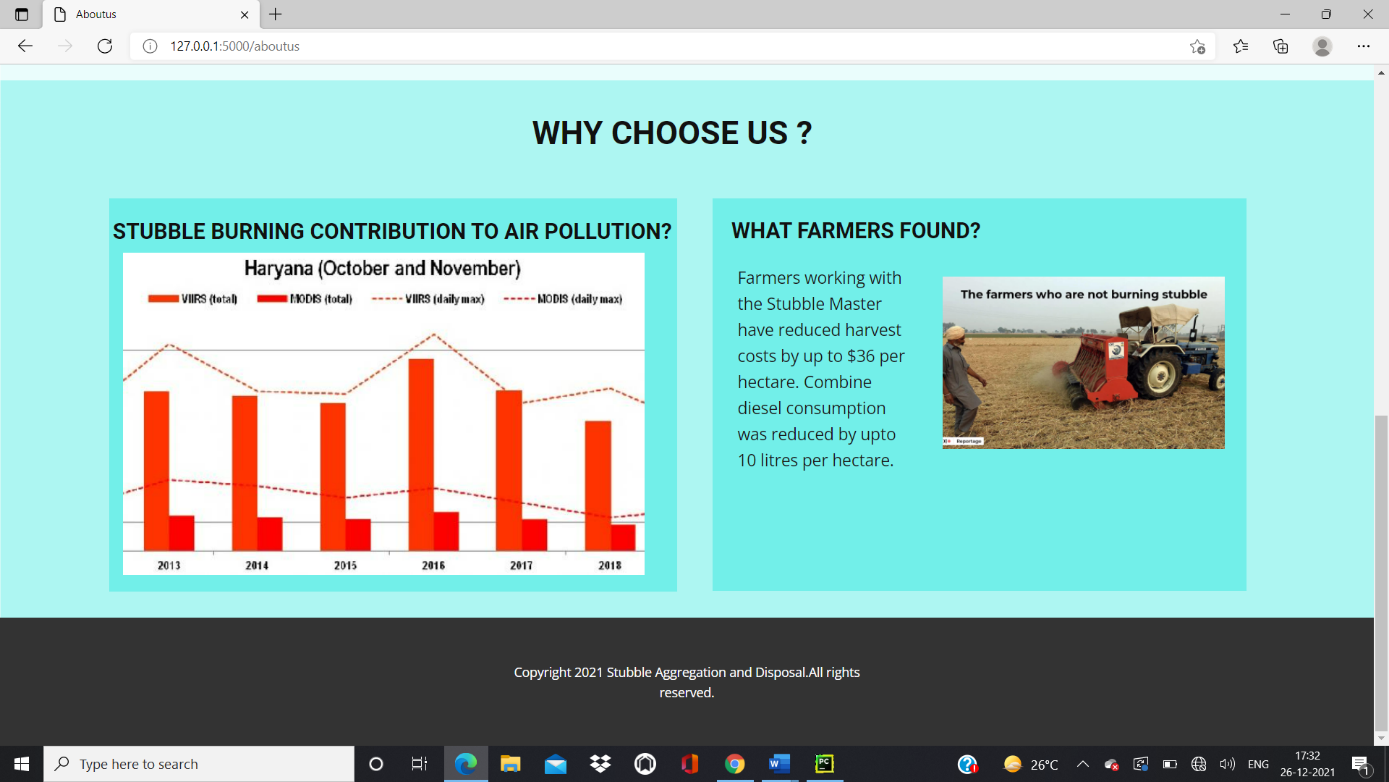


CONTACT US: -

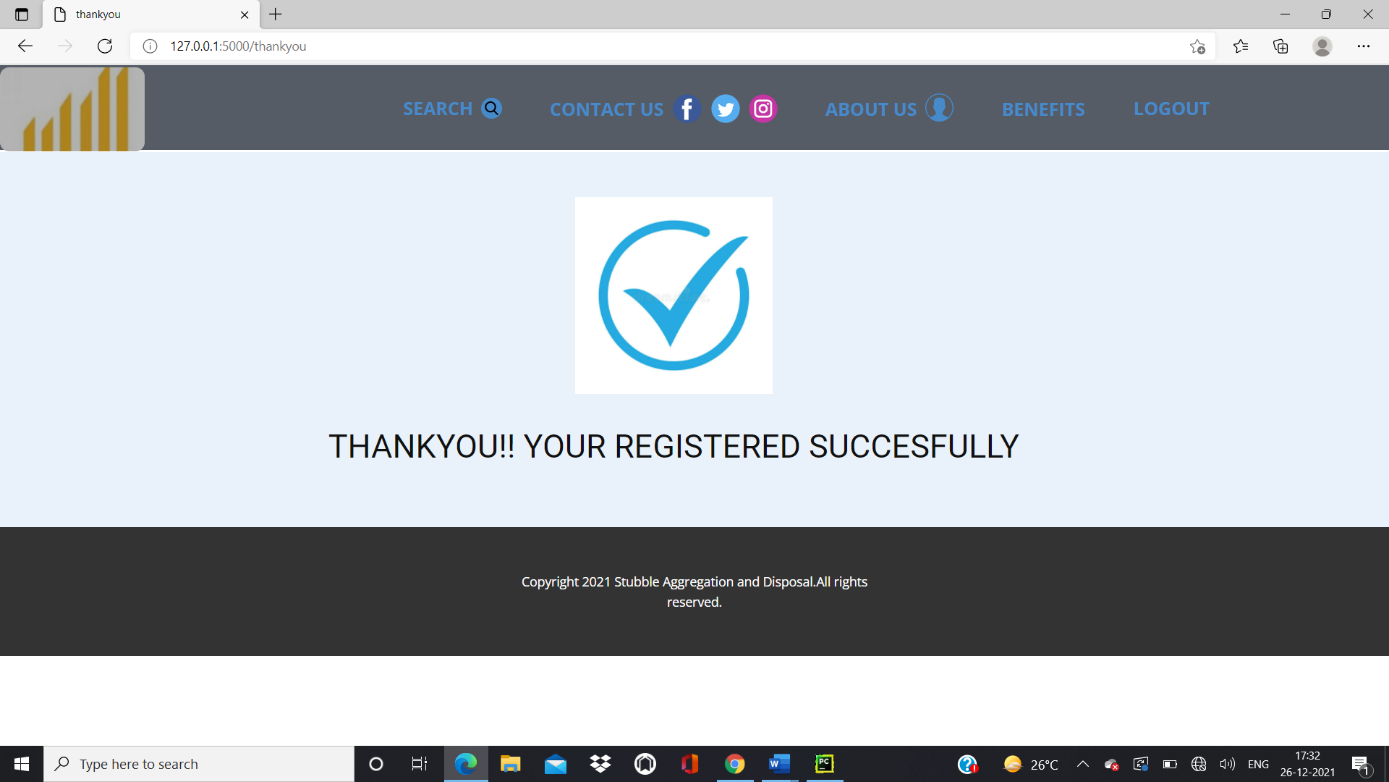


ABOUT US: -





THANKYOU:-



GITHUB LINKS AND FOLDER STRUCTURE: -

GitHub link:- <https://github.com/BejugamSathvika/Stubble-Management>

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

# 

# CHAPTER 6

## TESTING

**Unit Testing:** This is the lowest level of testing that is conducted to remove syntax & logic errors from a single unit. Individual components are used to ensure that they operate correctly. Each component is tested independently, without other system components.

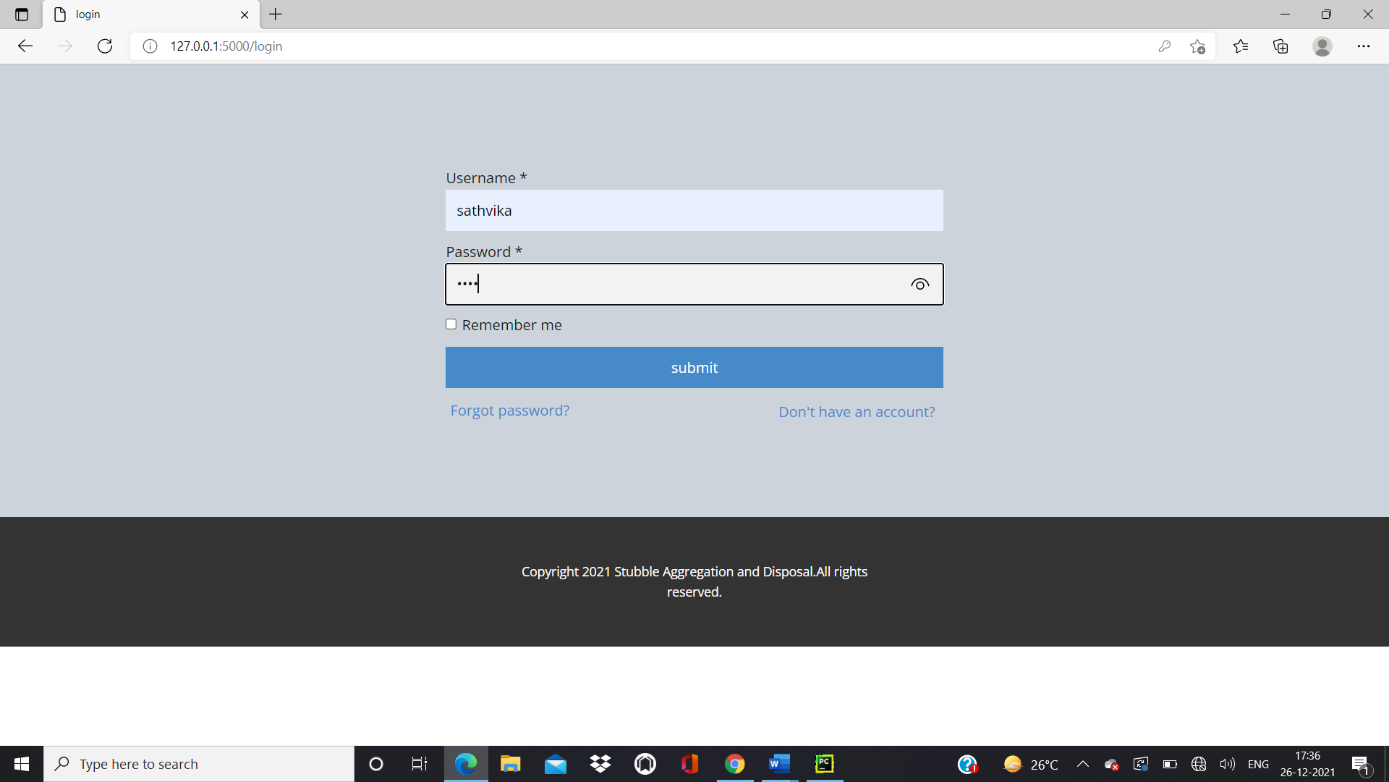
**Module Testing:** A module is a collection of dependent components such as an object class, an abstract data type or some looser collection of procedures and functions. A module encapsulates related components, so can be tested without other system modules.

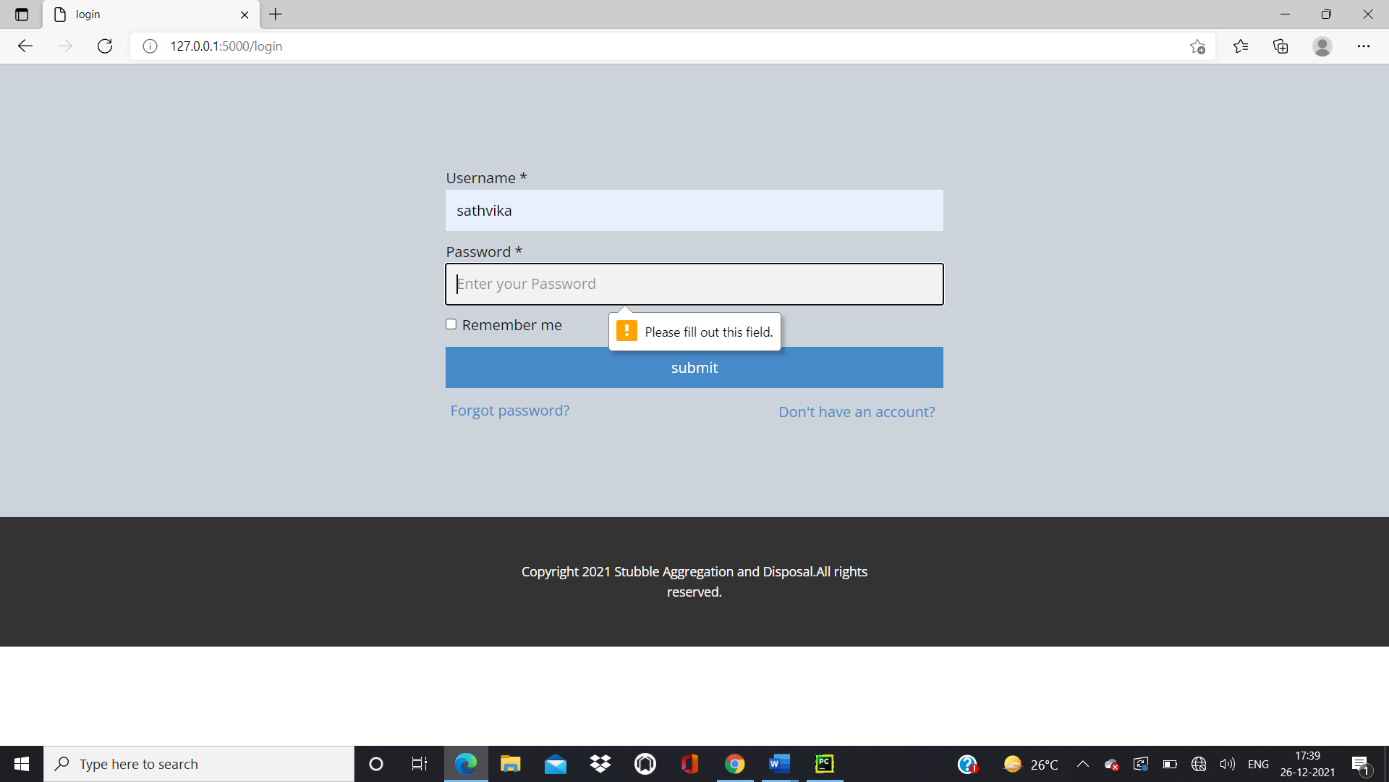
**System Testing:** The sub-systems are integrated to make up the system. The system as a complete entity is tested over here. This process is concerned with finding errors that result from unanticipated interactions between sub-systems. It is also concerned with validating that the system meets its functional and non-functional requirements and testing the emerged system properties.

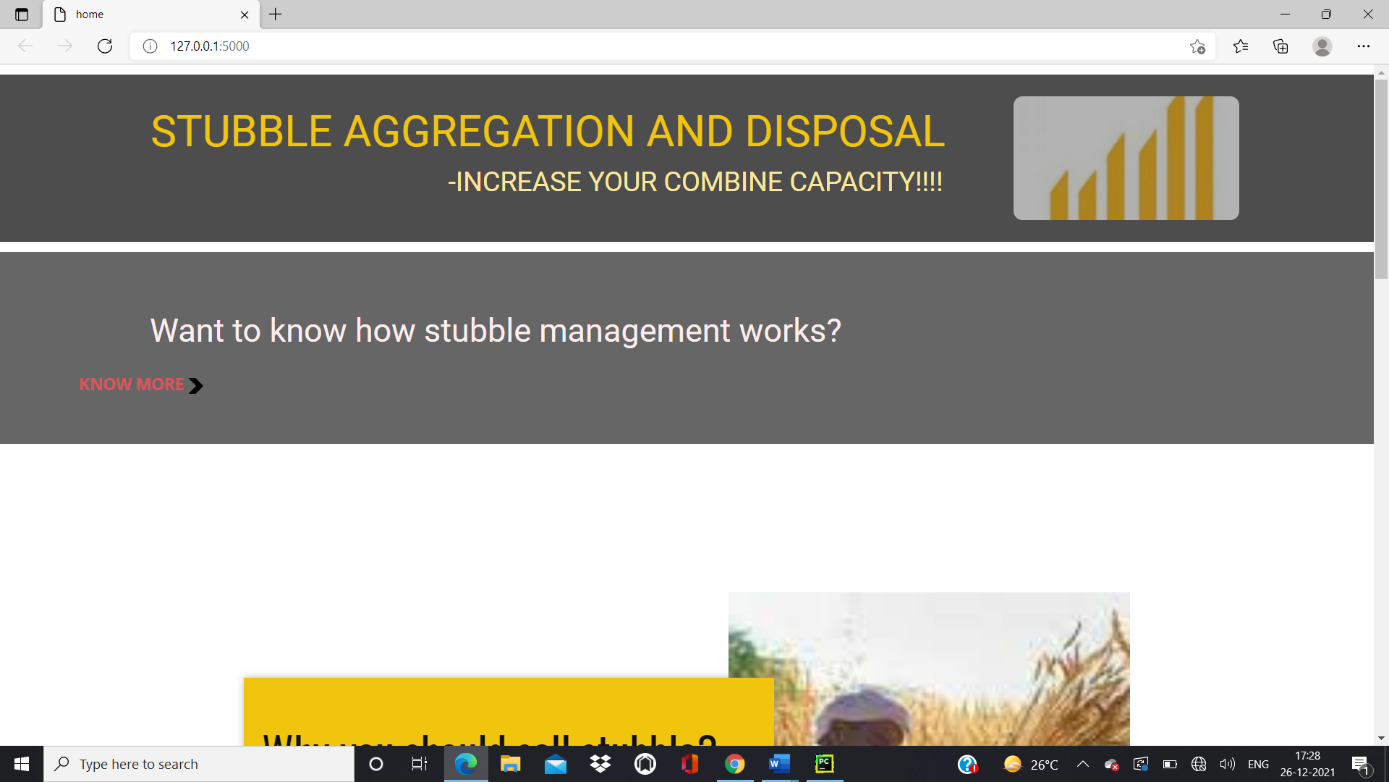
**Accepting Testing:** This is the final stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the system customer rather than simulated test data. Acceptance testing may reveal errors and omissions in the system requirements defined because the real data exercise the system in different ways from test data. It may also reveal requirements problems where the system ‘s facilities do not really meet the user’s needs, or the system performance is unacceptable.

**LOGIN TESTING:** -

If you enter wrong password, it will not direct you to next page. If you enter correct password, then it will direct you to next page. You need to fill every field and then click on submit button, otherwise it will ask you to fill every field.







REGISTER TESTING: -

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

DATABASE TABLES: -

Graphical user interface, text

Description automatically generated

LOGIN TABLE: -

Graphical user interface, text, application, email

Description automatically generated

INDUSTRIES TABLE: -

Graphical user interface, application

Description automatically generated

# 

STUBBLE REGISTRATION TABLE: -

Graphical user interface, table

Description automatically generated with medium confidence

# CHAPTER 7

## CONCLUSION:

We have successfully developed a secure, user-friendly web application. This system can search industries and sell stubble.

It uses a client/server architecture based on the HTTP protocol. It is developed in Microsoft’s

Visual Studio. The farmer or person will sell his stubble based on his wish. The business logic tier communicates with the database tier requesting the results of the query sent by it. The results obtained by the database are displayed on the data grid, by refreshing the grid rather than refreshing the entire web page.

This system will reduce the time to search for a nearest industry for farmer to sell his stubble.

## 

## FUTURE SCOPE:

This project has built as per the basic requirements of a user. But as time goes on there will be change in the requirements of the user like many farmers don’t know English or how to search. To make even more user-friendly website we can add voice recognition.

By this project attempt to burn stubble decreases farmers try to stick to fact the selling of stubble benefits them by making their cropping cycle more efficient and less costly. The benefits of this project are reducing harvest cost, reducing fuel consumption, eco-friendly environment.

Now the project can collect the stubble. But in future we want to even collect waste from industries, households like waste management system, and collecting of food from parties, hotels or even homes when they want to distribute remaining food like food management system. We want to add a module called payments for immediate transaction between farmers and industries and they can even track their stubble or waste and can see how it is changing as final product.

We usually shop from a website which offers us less price, discount, and coupons. Like that we even want to give coupons to users who shop in our website so, that they are more eager to shop.

# 

# CHAPTER 8

## REFERENCES

* <https://www.youtube.com/results?search_query=navin+reddy+django>
* <https://www.youtube.com/results?search_query=aman+dhattarwal+web+development>
* <https://www.sih.gov.in/sih2020PS>
* <https://www.w3schools.com/>
* <https://getbootstrap.com/>
* <https://www.bbc.com/news/world-asia-india-54930380>