**\_\_\_\_\_\_\_\_\_\_\_\_STUBBLIFY\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Submitted by**

**Laiba Faisal** (Fa-2021-BSCS-428)

**Laiba Imran** (Fa-2020-BSCS-114)

Session 2021-2025

**Supervised by**

**Mr Faiz Rasool**



**Department of Computer Science**

**Lahore Garrison University**

**Lahore**

**STUBBLIFY**

A project submitted to the

Department of Computer Science

In

Partial Fulfillment of the Requirements for the

Bachelor’s Degree in Computer Science

By

**Laiba Faisal**

**Laiba Imran**

**Internal Supervisor**

**Faiz Rasool \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Lecturer   
Department of Computer Science

**External Examiner** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Chairperson**

**Dr. Arfan Ali Nagra \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Associate Professor   
Department of Computer Science

**COPYRIGHTS**

This is to certify that the project titled “**STUBBLIFY”** is the genuine work carried out by **Laiba Faisal** **and Laiba Imran,** students of BSCS of Computer Science Department, Lahore Garrison University, Lahore during the academic year 2021-25, in partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Science and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Laiba Faisal

Laiba Imran

**DECLARATION**

This is to declare that the project entitled **STUBBLIFY** is an original work done by **Laiba Faisal, Laiba Imran** in partial fulfilment of the requirements for the degree “Bachelor of Science in Computer Science” at Computer Science Department, Lahore Garrison University, Lahore.

All the analysis, design and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or university.

**Group Members**

Laiba Faisal

Laiba Imran

**Supervisor**

**Mr Faiz Rasool**

Date: 05th May, 2025

**DEDICATION**

We would like to dedicate this project firstly to Allah Almighty who guided us with high seeking of knowledge and dedication to prosper our mission. Secondly, to our beloved parents who helped us by their support and guidance to always be dedicated and sincere to our goal. In the end, we would dedicate this project to our honorable teachers which increased our knowledge and also to our university, Lahore Garrison University and the Computer Science department.

**ACKNOWLEDGEMENTS**

First of all, we both would be highly grateful to Allah Almighty for providing us the patience and helping us stay focused and persevering on this path of difficulties and hard work. We have borne down our efforts in our Final Year Project. We cannot express enough thanks and gratitude to every person who has contributed to the road of success and completion of our project. We sincerely appreciate and acknowledge the part of all those individuals which played a general role in the completion of our project and would also like to extend our sincere thanks to all of them.

Secondly, we would like to express my deep sense of gratitude and harmony to our beloved Parents who were always there to uphold us morally, financially and showed their generous love and care throughout the entire period of learning. We are highly likely to thank all of the faculty members of Computer Science who have shared their utmost knowledge with us.

Lastly, I would like to express a deep sense of gratitude and honorable thanks to our Supervisor **Mr Faiz Rasool**, for encouraging us to proceed with this project and provide guidance alongside. He proved himself as a dedicated supervisor and will be a constant source of inspiration for us in our future as well.

**Table of Contents**

[**List of Tables** IX](#_Toc206475371)

[**List of Figures** X](#_Toc206475372)

[**Chapter 1** 1](#_Toc206475373)

[INTRODUCTION 1](#_Toc206475374)

[**Chapter 2** 4](#_Toc206475375)

[LITERATURE REVIEW 4](#_Toc206475376)

[**Chapter 3** 6](#_Toc206475377)

[PROBLEM DEFINITION 6](#_Toc206475378)

[**Chapter 4** 7](#_Toc206475379)

[SOFTWARE REQUIREMENT SPECIFICATION 7](#_Toc206475380)

[**Chapter 5** 17](#_Toc206475381)

[METHODOLOGY 17](#_Toc206475382)

[**5.1 Introduction** 17](#_Toc206475383)

[**5.2 Agile Model** 17](#_Toc206475384)

[**5.3 Data Collection** 19](#_Toc206475385)

[**5.4 Exploratory Data Analysis** 19](#_Toc206475386)

[**5.4.1 Feature Engineering** 20](#_Toc206475387)

[**5.4.2 Handling Missing values** 20](#_Toc206475388)

[**5.4.3 Handling Outliers** 20](#_Toc206475389)

[**5.4.4 Feature Correlation Heatmap** 21](#_Toc206475390)

[**5.5 Model Training for Feature Selection** 22](#_Toc206475391)

[**5.5.1 Random Forest Regressor and Feature Importance** 22](#_Toc206475392)

[**5.5.2 Extra Trees Regressor and Feature Importance** 23](#_Toc206475393)

[**5.5.3 Feature Selection** 24](#_Toc206475394)

[**5.6 Model Training** 24](#_Toc206475395)

[**5.7 Frontend Development** 29](#_Toc206475396)

[**5.7.1 Logo and Branding** 30](#_Toc206475397)

[**5.7.2 Model Integration** 30](#_Toc206475398)

[**5.7.3 API Integration** 30](#_Toc206475399)

[**5.7.4 Dashboard Design** 30](#_Toc206475400)

[**Chapter 6** 31](#_Toc206475401)

[DETAILED DESIGN AND ARCHITECTURE 31](#_Toc206475402)

[**6.1 System Architecture** 31](#_Toc206475403)

[**6.2 Detailed System Design** 33](#_Toc206475404)

[**Chapter 7** 47](#_Toc206475405)

[IMPLEMENTATION AND TESTING 47](#_Toc206475406)

[**Chapter 8** 54](#_Toc206475407)

[RESULTS AND DISCUSSION 54](#_Toc206475408)

[**Chapter 9** 57](#_Toc206475409)

[CONCLUSION AND FUTURE WORK 57](#_Toc206475410)

[REFERENCES 58](#_Toc206475411)

## **List of Tables**

[**Table 1: Comparison of Models 25**](#_Toc204034459)

[**Table 2 Models Results 27**](#_Toc204034460)

[**Table 3: Comparison of Feature Importance Plots 29**](#_Toc204034461)

[**Table 4: Testing Requirements 49**](#_Toc204034462)

[**Table 5: Registration Test Case 49**](#_Toc204034463)

[**Table 6: Login Test Case 50**](#_Toc204034464)

[**Table 7: Dashboard Test Case 50**](#_Toc204034465)

[**Table 8: Data Fetching Test Case 51**](#_Toc204034466)

[**Table 9: Pre-trained Model Test Case 51**](#_Toc204034467)

[**Table 10: Predictions Test Case 52**](#_Toc204034468)

[**Table 11: Logout Test Case 52**](#_Toc204034469)

[**Table 12: Test Cases Results 53**](#_Toc204034470)

## **List of Figures**

[**Figure 1: Agile Model 18**](#_Toc205360153)

[**Figure 2: Feature Correlation Heatmap 21**](#_Toc205360154)

[**Figure 3: Feature Importance from Random Forest Regressor 22**](#_Toc205360155)

[**Figure 4: Feature Importance from Extra Tree Regressor 23**](#_Toc205360156)

[**Figure 5: LightGBM Training Curve 25**](#_Toc205360157)

[**Figure 6: XGBoost Training Curve 26**](#_Toc205360158)

[**Figure 7: Random Forest Training Curve 26**](#_Toc205360159)

[**Figure 8: LightGBM Feature Importance Plot 28**](#_Toc205360160)

[**Figure 9: XGBoost Feature Importance Plot 28**](#_Toc205360161)

[**Figure 10: Random Forest Feature Importance Plot 29**](#_Toc205360162)

[**Figure 11: Stubblify Logo 30**](#_Toc205360163)

[**Figure 12: Use Case Diagram 37**](#_Toc205360164)

[**Figure 13: ER Diagram 38**](#_Toc205360165)

[**Figure 14: Architectural Diagram 39**](#_Toc205360166)

[**Figure 15: Activity Diagram 40**](#_Toc205360167)

[**Figure 16: Sequence Diagram 41**](#_Toc205360168)

[**Figure 17: Component Diagram 42**](#_Toc205360169)

[**Figure 18: State Machine Diagram 43**](#_Toc205360170)

[**Figure 19: Class Diagram 44**](#_Toc205360171)

[**Figure 20: Data Flow Diagram 45**](#_Toc205360172)

[**Figure 21: Database Diagram 46**](#_Toc205360173)

[**Figure 22: Signup Page 54**](#_Toc205360174)

[**Figure 23: Login Page 54**](#_Toc205360175)

[**Figure 24: Dashboard 55**](#_Toc205360176)

[**Figure 25: Predictions 55**](#_Toc205360177)

[**Figure 26: About Us 56**](#_Toc205360178)

[**Figure 27: Contact Us 56**](#_Toc205360179)

**List of Abbreviations**

|  |  |
| --- | --- |
| PSCA | Punjab Safe Cities Authority |
| DFD | Data Flow Diagram |
| ER | Entity Relationship |
| SRS | Software Requirement Specification |
| HTTP | Hypertext Transfer Protocol |
| URL | Uniform source Locator |
| API | Application Programing Interface |
| LightGBM | Light Gradient Boosting Machine |
| XGBoost | Extreme Gadient Boosting |

**ABSTRACT**

Stubble burning remains a matter of grave concern being the primary causative agent of the recurring smog crisis in Punjab, Pakistan. The resulting smog and deteriorating air quality adversely impact public health and disrupt the daily life activities. STUBBLIFY is an AI-based system that deploys a pre-trained machine learning model, Light Gradient Boosting Machine (LightGBM) regressor, to forecast stubble burning incidents. The model has been trained on the stubble burning dataset for New Delhi, India, that is sourced from Kaggle. The model achieved an R2 score of 0.96, indicating its high predictive performance. A user-authenticated dashboard is provided by the Django-built frontend, which allows users to view the data fetched in real-time through third party APIs alongside the predicted number of fire counts generated by the Light GBM Regressor. To mitigate the negative effects of burning crop residue, STUBBLIFY serves an early warning tool that will assist the Punjab Safe Cities Authority (PSCA) to take proactive measures and implement the necessary policies in due time. Moreover, the effectiveness of these policies can be monitored over time by regularly analyzing the system’s predictions.

# **Chapter 1**

## INTRODUCTION

**1.1 Background**

Stubble burning is the practice of burning the leftover crop material after harvesting. This is common in agricultural regions such as Punjab. Although this approach is efficient and economical for farmers, there are serious risks to the environment and public health. Air quality is severely deteriorated when stubble is burned because it releases dangerous pollutants into the atmosphere, such as carbon dioxide, carbon monoxide, and particulate matter. By raising greenhouse gas emissions, this pollution aggravates climate change and adds to global warming. Additionally, the smoke from these fires has the ability to spread over great distances, affecting not only the immediate area but also areas well beyond the fields where the burning takes place, thereby affecting larger populations and areas.

Burning stubble has a negative impact on public health, especially respiratory disorders, in addition to harming the environment. Asthma, bronchitis, and other chronic respiratory diseases are more likely to develop in people who live in areas where stubble burning occurs frequently. Children, the elderly, and people with pre-existing medical conditions are among the vulnerable groups that may experience both immediate and long-term health issues as a result of the harmful fumes emitted during the burning process. The need for effective real-time prediction and management solutions grows as the practice expands. Through the identification of high-risk areas, prompt interventions, and the promotion of sustainable, alternative farming methods that are less detrimental to the environment and human health, these solutions can help lessen the negative effects of stubble burning.

**1.2 Objective**

The project, **Stubblify**, seeks to address this serious environmental problem of stubble burning by creating a reliable system for real-time prediction of stubble burning occurrences. This system analyzes weather and air quality data using a machine learning model to estimate the immediate risk of stubble burning incidents. The system enables authorities and environmental organizations to keep track of the possibilities of field fires and take prompt measures to prevent or control stubble burning.

The project uses the **LightGBM Regressor** model for its machine learning component. It has been trained on the data for New Delhi, India, available on Kaggle. The dataset contains weather-related variables that affect the probability of stubble burning, such as temperature, humidity, and wind speed, and air pollutants data, such as carbon monoxide and PM2.5 (particulate matter ≤ 2.5 micrometers). Based on these variables, the trained model guages the risk of stubble burning incidents in real-time, providing a more thorough and precise prediction system.

A dashboard built with Django has been created to provide a responsive and easy-to-navigate user interface to view the predictions. Authorized users (i.e., concerned members of the PSCA) can securely log in to access the system. Stubblify addresses the unmet need for a trustworthy, data-driven method to forecast stubble burning, which will benefit the environment and public health. By deploying a machine learning model, Stubblify aims to produce precise predictions in real-time that can direct immediate actions in agricultural regions like Punjab.

**1.3 Report**

The purpose of this report is to provide a thorough overview of the project. This chapter covers its background, development process, and potential applications, as outlined below:

* Introduces the concept of stubble burning.
* Describes the methodologies used in project development, including the machine learning models tried and evaluated.
* Explains the Django-based frontend, featuring a dashboard accessible after user authentication.
* Discusses the overall effectiveness of the predictive model.

The chapter-wise breakdown of the report is as follows:

**Chapter 1:**

Background and overview of the project

**Chapter 2:**

Literature review related to the project

**Chapter 3:**

Definition and elaboration of the problem statement

**Chapter 4:**

Software Requirements Specification (includes the functional and non-functional Requirements)

**Chapter 5:**

Details of the methods, tools and technologies used

**Chapter 6:**

System architecture and design

**Chapter 7:**

Testing of the system and maintenance plan

**Chapter 8:**

This defines the evaluation of results.

**Chapter 9:**

The future direction is described.

# **Chapter 2**

## LITERATURE REVIEW

Stubble burning is a method used traditionally by farmers in underdeveloped countries to clear fields rapidly and cost-effectively after the harvesting season. This allows for swift planting for the next winter crop [1]. Gurdev Singh published a study that highlights the environmental and health hazards of stubble burning, such as the release of greenhouse gases and particulate matter, posing serious health risks to both humans and animals. This makes it necessary to urgently intervene and direct the farmers to adopt sustainable and environment-friendly practices for crop residue management [2]. A study found that stubble burning changes the properties of soil, disrupting soil ecosystems, and nutrients cycles. It leads to a significant drop in soil fertility [3].

Navya Koura’s study on stubble burning in Moga, Punjab links the issue to crop subsidies and free utilities that encourage rice-wheat cycles, resulting in excess stubble. With limited alternatives, farmers resort to burning, causing air pollution, soil damage, and health risks. The study recommends crop diversification, subsidy reforms, awareness campaigns, and improved stubble management tools for sustainable agriculture [4].

Another study used the MODIS active fire count satellite data to analyze stubble burning incidents in Punjab and Haryana, India during the year 2021. The emissions of carbon monoxide and PM2.5 were monitored to gauge the impact on air quality of Delhi. The results showed that 30-35% of pollution in Delhi during October–November 2021 was caused by burning of crop residue [5].

This study uses MODIS Active Fire Data (MCD14DL-NRT) to identify stubble burning hotspots across India, focusing on trends over the past two decades. It finds a sharp spike in fire activity during October–November, aligning with the paddy harvest season. The research highlights Punjab as the most affected state, with stubble burning during winter contributing to severe air pollution and health risks in the Indo-Gangetic Plain [6].

A study introduces a system designed to calculate a burning index score for crops, helping tackle problems like pollution and soil nutrient loss caused by stubble burning. It identifies likely burning zones by analyzing the harvest season, current crop area maps, and historical fire hotspot data. To improve accuracy, the system uses a mix of satellite imagery, drone-based sensing, and crowdsourced inputs to flag high-risk areas for timely intervention [7].

A study explores how machine learning and blockchain can work together to detect and prevent stubble burning. Satellite images processed through trained machine learning models help pinpoint pollution sources in real time. If pollution levels exceed a certain threshold, the location is securely stored on the blockchain, making it accessible to authorities through a decentralized app for timely action [8]. In a study, Sentinel-2 data and fuzzy ML techniques were used to detect burnt paddy fields, finding that kernel-based MPCM classifiers were more accurate and robust than distance-based ones. It offers an effective tool for monitoring stubble burning [9].

Apart from this, Sentinel-2A/B satellite imagery and a modified fuzzy classification method were deployed to detect and analyze stubble burning in Patiala, Punjab. It found a sharp increase in burning activity between October 30 and November 19, 2018. The results offer valuable insight into estimating pollution levels linked to agricultural burning. [10]. Furthermore, stubble burning in Punjab causes sharp rises in PM2.5 and PM10 levels, especially in hot spots like Bathinda. It shows the link between air pollution and stubble burning therefore, emphasizing the role of weather in pollutant buildup [11].

This study enhances field-level detection of paddy stubble burning in India by combining PCM-S and ISM classification on PlanetScope and Sentinel-2 data. It accurately identified 27.07 sq. km of burnt fields near Patiala, demonstrating high accuracy and clear class separation. The findings emphasize the persistent nature of stubble burning despite mitigation efforts, urging immediate action [12].

In another study, a machine learning-based approach to detect crop residue burning in smallholder sugarcane fields using high-resolution HLS satellite data is presented. It compares support vector machine (SVM) and neural networks, with artificial neural network (ANN) showing higher accuracy The study highlights how integrating remote sensing with advanced classifiers can significantly improve the mapping of burnt areas [13].

# **Chapter 3**

## PROBLEM DEFINITION

Stubble burning is a common problem in the agricultural areas of underdeveloped countries where it is customary to burn crop residues that remain after harvest. It is a rapid and economical way to clear the fields for the next crop. Although machinery is available that chops off the stubble, the machinery is expensive and therefore cannot be afforded by the poor farmers. However, the burning of residues has detrimental effects on the environment and public health. Particulate matter, carbon monoxide, and volatile organic compounds are among the dangerous pollutants released by the practice, which greatly increase air pollution and respiratory ailments. Due to a lack of efficient real-time prediction and prompt intervention systems, stubble burning persists despite its dangerous effects.

The majority of the reactive techniques used today to control and lessen stubble burning only deal with the problem after it has already happened. Increased pollution levels and health risks result from the lack of an effective system to assess the current stubble burning will risks. Furthermore, authorities, farmers, and environmental organizations find it challenging to take actions immediately when real-time, early warning systems are lacking.

An important environmental concern is the lack of a proactive strategy to monitor and manage real-time stubble burning incidents. By using machine learning for real-time fire risk estimation based on weather and air quality data, Stubblify addresses this issue. The project intends to provide a solution that can assess the current likelihood of stubble burning incidents by utilizing advanced data analysis. This will enable immediate and rapid intervention by the PSCA and lessen the environmental harm.

# **Chapter 4**

## SOFTWARE REQUIREMENT SPECIFICATION

**4.1 Introduction**

**4.1.1 Purpose**

This project is built to predict the likelihood of stubble burning incidents using real time data for weather and air pollutants. The functional, non-functional, and technical requirements of the model are described here, so that stakeholders and developers can understand this easily. The scope is focused on the prediction system component that uses environmental monitoring framework. This document addresses the machine learning subsystem that processes temperature, humidity, wind direction, and particulate matter concentrations (PM2.5).

**4.1.2 Document Conventions**

For the formatting of this document, the following conventions have been used:

* Writing style: Times New Roman
* Font size: 12 for text, 14 for sub-headings and 16 for headings
* Line spacing: 1.5

**4.1.3 Intended Audience and Reading Suggestions**

This section describes the intended audience for this software requirements specification chapter. It discusses the stakeholders that are involved in the development, deployment and usage of this predictive system. Each audience can use specific sections of the document in which they are interested. The list of intended audience is as below:

**4.1.3.1 Developers**

The people that are involved in developing the machine learning model and the front-end can resort to this chapter as a reference for system requirements, data specifications and integration details. They can focus on functional requirements and system features to understand the system’s functionality and architecture.

**4.1.3.2 Data Engineers and Researchers**

Data engineers and researchers can understand the requirements for data collection, preprocessing and model development. They can see the data requirements described in this document to understand the data set specifications.

**4.1.3.3 Testers**

Testers can refer to this chapter to design test cases and validate the model accuracy and functionality against defined requirements. They can see the functional requirements and system features to identify key components for testing.

**4.1.3.4 End Users**

End users include PSCA and its policy makers, and environmental agencies. This chapter will help them understand the system better to make the best use of it.

**4.1.3.5 Documentation writers**

Document writers can use this chapter as a foundation for creating user manuals and help guides.

**4.1.4 Product Scope**

This project is designed to predict the current likelihood of stubble burning in agricultural regions through real-time nowcasting. It aims to help mitigate the harmful impact of burning crop residue on public health and the environment. By analyzing data such as weather conditions and air pollution in real-time, the number of fire incidents due to stubble burning is estimated in real-time. **Stubblify** gives accurate predictions that will help policymakers, environmental agencies, and farmers to take prompt measures and intervene timely so as to minimize the associated hazards. The system is being developed to provide support to efforts to minimize air pollution and prevent soil degradation, while encouraging efficient use of resources and adoption of sustainable agricultural practices. Stubblify gives valuable real-time insights to stakeholders that will help them to make informed decisions that protect the environment, improve public health, and contribute to global initiatives against climate change.

**4.2. Overall Description**

**4.2.1 Product Perspective**

The Stubble Burning Prediction model**, Stubblify,** is a standalone, data-driven solution aimed at tackling the rising problem of stubble burning and its impact on the environment. As an independent product, it employs a machine learning model to analyze various data points, including weather conditions, and air quality data to nowcast the chances of stubble burning in real time in particular areas. Stubblify is developed to be compatible with current agricultural monitoring systems, providing stakeholders with a real-time risk assessment tool that enhances decision-making and helps to reduce environmental damage.

Although Stubblify is a new and self-sufficient product, it can play a vital role in broader environmental management efforts. By delivering real-time insights, it supports larger systems focused on lowering air pollution, managing crop residue, and promoting sustainable farming practices. Its interface facilitates data sharing with external weather and air quality APIs, ensuring it works well with related systems.

**4.2.2 Product Functions**

The product functions of the system are described below. Here is an overview of its main features:

* Data Collection and Integration
* Data Preprocessing
* Data fetching in real-time (using external APIs)
* Prediction Module
* Visualization of prediction results
* Risk Status

All these functions are interlinked through a smooth workflow that routes data from external APIs, through preprocessing, prediction, and visualization modules, providing actionable insights for effective interventions and decision-making.

**4.2.3 User Classes and Characteristics**

The primary users of Stubblify are the authorized personnel and decision-making bodies within PSCA and environmental agencies. The users are expected to know how to use basic web apps to help them navigate through the dashboard. They are expected to have background knowledge of agricultural practices and environmental pollution. They must be capable of interpreting the system output to design and implement strategic policies timely to prevent field fires.

**4.2.4 Operating Environment**

The system has been built to function seamlessly in both cloud-based and on-premises environments. It works well with standard hardware like desktops, laptops, and servers. The system is compatible with major operating systems, including Windows, Linux, and macOS. To obtain real-time data from external APIs, it needs a stable internet connection.

**4.2.5 Design and Implementation Constraints**

The stubble burning predictive model is being developed using Python and Django, but there are a few constraints to keep in mind. The system should be able to run on standard hardware while also being optimized for low-resource devices. At the same time, we want to use high-performance tools like GPUs for handling large datasets efficiently. The backend is developed using python and frontend dashboard is developed using Django. It also needs support for external APIs that provide weather, and air quality data, using protocols. We also need to follow the best practices for Django to make sure our system is scalable and maintainable.

**4.2.6 Assumptions and Dependencies**

The development and operation of thesystem depends on several key assumptions and dependencies. One major assumption is that we'll have access to reliable and sufficient data like weather patterns, crop types, and satellite images sourced from APIs and government databases. This data is crucial for effective training and running of our predictive model. Moreover, the project depends on stable internet connectivity for real-time data retrieval and system operation.

Furthermore, the model depends on the consistent performance of external libraries and frameworks, such as those used for machine learning (for instance, Scikit-learn or TensorFlow). These tools must remain supported and updated throughout the development and deployment phases. Any changes regarding these assumptions—like data availability or support for external libraries—could significantly affect the system's functionality, accuracy, or delivery timeline.

**4.3 External Interface Requirements**

**4.3.1 User Interfaces**

This section outlines the user interface layout, design and functions of each user interface of the Django-based web application. It will include the following interfaces:

* Dashboard – that will display the predicted fire count, risk status and the values for the parameters fetched from APIs in real-time and visualization related to the stubble burning across the region
* Input form – allows the user to enter city name for which prediction is required
* Prediction Results - the screen displays the results of the prediction based on the data input by the user
* Risk status – this shows whether a city is at high, medium or low risk of having stubble burning
* Standard buttons - that help to navigate

**4.3.2 Hardware Interfaces**

1. **Supported Device types:**

* Servers: the prediction algorithms run on servers that might be cloud based
* Client devices: the app can be accessed through desktops laptops and mobile phones (web browsers)
* Storage devices: either network attached storage for cloud storage platforms like AWS will be deployed to store the substantial amounts of data

1. **Communication protocols:**

* Client server communication: use HTTPS for secure data transmission
* Data acquisition: FTPs and APIs will be used to fetch data from the government databases
* Storage access: RESTful API's will be used to fetch data from cloud platforms

1. **Server requirements:**

A server with a minimum of 32GB RAM and two terabyte SSD with multi core processor that supports NVIDIA GPU will be required.

**4.3.3 Software Interfaces**

1. **Operating System**

Windows or Ubuntu

1. **APIs**

* Weather Data API
* Air Quality API

1. **Tools and Libraries**

* Django framework
* Machine Learning Libraries

**4.3.4 Communications Interfaces**

For safe online communication, the system will mostly use the HTTP/HTTPS protocols. Real-time predictions, reports, and insights will be accessible to stakeholders through the user interface.

Data fetched through APIs will be in the JSON or CSV format for compatibility. Backups and fail-safe protocols will be integrated.

**4.4** **System Features**

This section describes the key features of the project along with their functional requirements.

**4.4.1 Fetch data from APIs**

**4.4.1.1 Description and Priority**

STUBBLIFY will make use of the APIs required to fetch real-time data from the meteorological website and air quality website.

**4.4.1.2 Stimulus/Response Sequences**

**Stimulus:**

* The user requests for the latest data from the interface.

**Response:**

* The system sends a request to the external APIs to fetch data.
* The external APIs respond with the data.
* The system preprocesses the data and validates its correctness.
* The system integrates the data into the prediction model.
* The system makes predictions.
* The system displays the prediction results.
* In case of failure of API request, the system notifies the user.

**4.4.1.3 Functional Requirements**

**REQ-1: API Integration**

* The system should integrate APIs to fetch meteorological and air quality data.

**REQ-2: Data Request**

* The system should send HTTPS requests to the external APIs with required parameters (e.g. region).

**REQ-3: Data Processing**

* The system should validate the API responses to ensure accuracy and correctness of the data.
* The system should handle missing values or corrupted data (e.g. trying to retrieve data again).

**REQ-4: Real-time fetching**

* The system should fetch real-time data when the user requests update

**REQ-5: Error Handling**

* The system should log errors in case of failed API requests.
* The system should notify the users if data fetching fails.
* The system should provide a retry mechanism in case of failure in data fetching.

**4.4.2 Data Preprocessing**

**4.4.2.1 Description and Priority**

The system will preprocess the data obtained through APIs.

**Priority:** High

**4.4.2.2 Stimulus/Response Sequences**

**Stimulus:**

* The system receives real-time data through the external APIs.

**Response:**

* The system detects missing values.
* The system identifies outliers.
* The system standardizes numerical data.
* The system stores the processed data in the database.

**Stimulus:**

* The system detects missing values.

**Response:**

* The system fills in the missing values using a strategy (mean and mode imputation from historical data).

**Stimulus:**

* The system identifies outliers.

**Response:**

* The system informs the user about the outliers.
* The system will replace the outliers with a threshold value from the historical data.

**4.4.2.3 Functional Requirements**

**REQ-1: Detecting and handling missing values**

* The system should identify any missing values.
* The system should implement a method to fill in the missing values.
* Mean imputation for numerical data and mode imputation for encoded categorical data should be performed.

**REQ-2: Outliers Removal**

* Once detected, the anomalies can be replaced with extreme values from the relevant historical data.

**REQ-3: Data Standardization**

* The system should normalize the data to match the scale of features used in the model training procedure.

**REQ-4: Real-time Feature Selection**

* The system should be capable of identifying and extracting only the useful features from the dataset.

**REQ-5: Data Transformation**

* The system should convert the data to the required format in case of any discrepancies related to the data format.

**4.4.2.4 Functional Requirements**

**REQ-1: Integration of real-time data**

* This is ensured by the feature 4.4.1 Fetch data from APIs.

**REQ-2: Preprocessing of the data**

* This is ensured by the feature 4.4.2 Data Preprocessing.
* The pre-processed data should be given to the model.

**REQ-3: Model Integration**

* The system should incorporate the model (trained on existing data).

**REQ-4: Model Output**

* The system should make accurate predictions.

**REQ-5: Output Visualization**

* The system should display the model output through maps and heatmaps.

**REQ-6: Error Handling**

* The system should log the error details.
* The system should notify the user about the error.

**4.4.5.3 Functional Requirements**

**REQ-1: Visualization Formats**

* This system should support the display of real-time data fetched through the APIs.

**REQ-2: Data Loading and Display**

* The system should accurately retrieve the output of the model.
* The system should render the model output on the interface accurately.

**REQ-3: Color Coding**

* The system should show the risk status of a city using colors (red for high alert, yellow for medium risk and green for safe).

**REQ-4: Interactivity**

* The system should allow the users to interact with the visualizations.

**REQ-5: Responsive Design**

* The visualizations should be responsive and easy to interpret.
* The visualizations should be resized automatically to fit the screen size of the user’s device.

**REQ-6: Interface Update**

* The system should update the visualization when the user requests an update and the model output for the updated data is displayed.

**4.4.3.4 Functional Requirements**

**REQ-1: Model Reuse**

* The system should be able to make repeated use of the model.

**REQ-2: Execution Time**

* The system should be capable of making predictions in pre-defined time.

**REQ-3: Output Formatting**

* The system should format the model output to create useful visualizations for the user.

**REQ-4: Error Handling**

* The system should handle errors during predictions.
* The system should notify the user about the error.
* The system should list the possible trouble shooting methods.

**4.5** **Other Nonfunctional Requirements**

**4.5.1 Performance Requirements**

The system should fetch data though APIs within 5 seconds. Prediction latency should be minimal, for example, 10 seconds for batch input of 1000 instances. Database operations should be completed in no more than 2 seconds. This ensures efficiency of the system. The data backups should not take much time.

**4.5.2 Safety Requirements**

The system should have a fail-safe mechanism to pause operations in case of failure. The accuracy of the system should be high, let’s say greater than 85 percent.

**4.5.3 Security Requirements**

The database transmissions between APIs and the system should be encrypted using the HTTPS protocol. The system must log each operation performed by the user and the system.

**4.5.4 Software Quality Attributes**

The user interface must be interpretable for non-technical users. The system must handle discrepancies in data without crashing. The system must not have any compatibility issues with the APIs and external software.

**4.5.5 Business Rules**

The model should be updated and retrained bi-annually. User access controls can be introduced.

# **Chapter 5**

## METHODOLOGY

### **5.1 Introduction**

This chapter discusses the steps to attain the objective of the project. The methodology is discussed in detail and all the steps involved in making the final product are elaborated thoroughly. The reasoning involved in each decision step is also justified and discussed.  
The software development life cycle approach used, the dataset source, the preprocessing techniques, the machine learning models applied, and their results are thoroughly discussed.

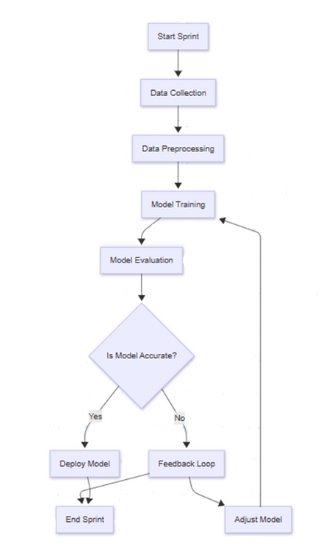
In addition, key challenges encountered during both the model training and deployment phases are highlighted. The overall pipeline from data acquisition to real-time prediction is mapped to provide a comprehensive understanding of the system. Comprehensive figures and tables have been added, where appropriate, to provide visualizations for better interpretation.

### **5.2 Agile Model**

The agile model was used for Stubblify because of the nature of the project that needs continuous experimentation and refinement of models used. Agile model’s iterative and flexible nature allowed continuous improvements ensuring that the best machine learning model was deployed for accurate predictions.

The machine learning models (Light GBM, XGBoost, and Random Forest) needed frequent fine-tuning and hyperparameter optimization frequently to achieve maximum accuracy. The agile model helped to conduct regular evaluation of these models and improve their accuracies. The feedback loops provided insights to redesign the models for better performance.

Moreover, the agile approach facilitated seamless collaboration between the development and research components of the project. It also enabled the integration of new data sources and features without disrupting the existing workflow, ensuring adaptability to evolving project requirements



**Figure 1: Agile Model**

### **5.3 Data Collection**

Data collection is an important phase of any project. In this stage, we thoroughly searched the internet for all available resources and the available datasets. The historical data related to stubble burning in Pakistan was not available to us. Therefore, the closest match was the dataset for stubble burning in New Delhi, India, sourced from Kaggle [14].

This dataset has data for five different stations within New Delhi for the months from September to December for the years 2012 to 2021. It has 14 input features and one output column (i.e. the fire counts). The input features are:

* Wind direction (WD)
* Wind speed (WS)
* Relative humidity (RH)
* Air temperature (AT)
* Gross State Domestic Product (GSDP)
* Gross State Value Added (GSVA)
* GSDP per Capita (GSDP\_CAP)
* Human Development Index (HDI)
* Particulate matter 10 (PM 10)
* Particulate matter 2.5 (PM 2.5)
* Carbon monoxide (CO)
* Sulfur dioxide (SO2)
* Nitrogen dioxide (NO2)
* Date

The data for these five stations of New Delhi was merged into a single csv file. The missing values were looked for.

### **5.4 Exploratory Data Analysis**

The exploratory data analysis (EDA) was conducted to gain insights into the patterns and relationships within the dataset. The main purpose was to identify the most influential features for stubble burning prediction and remove the noisy features from the dataset. It comprised of the steps discussed in this section.

## **5.4.1 Feature Engineering**

The “Date” column was processed to extract multiple temporal features that could improve model performance. Specifically, “DayOfYear” was used to capture seasonal trends in stubble burning, while “Weekday” helped identify any recurring weekly patterns, such as increased activity on weekends or weekdays. The “Month” feature was useful as it is the most important feature that helps to detect the peak burning periods aligned with harvesting seasons. These features were added to the existing dataset to ensure that the temporal features were learnt by the models.

Later, during feature selection, some of these had been removed on the basis of feature selection method (discussed later in this chapter).

## **5.4.2 Handling Missing values**

Mean imputation was used to fill the missing values in the numeric columns. The missing values were replaced with the mean of the entire column. Certain rows had missing or corrupted values in the date column and therefore had been removed.

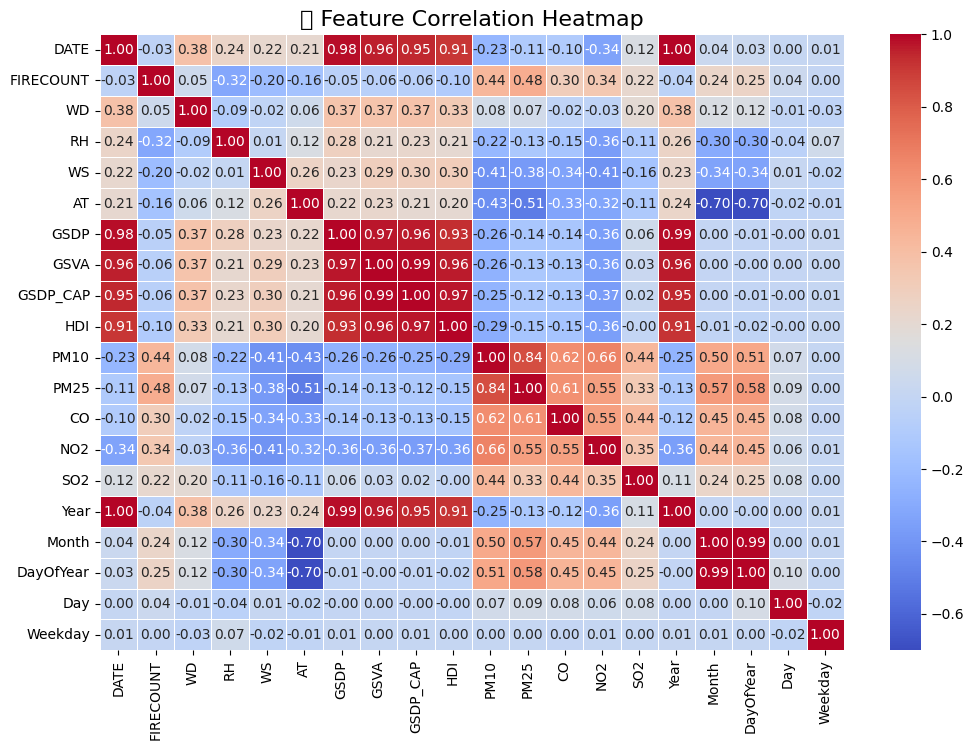
## **5.4.3 Handling Outliers**

Outliers are data points that deviate significantly from the general distribution of a dataset and can have a disproportionate impact on machine learning models, especially when the dataset is relatively small. We used the interquartile range (IQR) method to identify and manage such outliers.

A python function was used for the purpose. It computed the first quartile (Q1) and third quartile (Q3) of each numerical feature to obtain the IQR (defined as Q3 - Q1), that represnts the middle 50% of the data. It then capped the extreme values between the calculated upper (Q1 - 1.5 × IQR) and lower (Q3 + 1.5 × IQR) bounds.

This technique was chosen to clean the data without altering its original structure. Since the dataset was relatively small, completely removing outliers could have resulted in the loss of important information. By capping the extreme values instead of deleting them, the impact of unusually high or low data points was reduced without compromising the overall integrity of the dataset. This approach helped ensure that the model could learn from as much data as possible while still being protected from the distortion that outliers can cause during training.

## **5.4.4 Feature Correlation Heatmap**

The feature correlation heatmap was plotted to visualize the correlation among different features.

**Figure 2: Feature Correlation Heatmap**

The heatmap shows Pearson correlation coefficients between variables. It shows that pollutants like PM2.5, CO and PM10 have a moderate to strong positive correlation with fire count. PM2.5 appears to be the strongest indicator for stubble burning activity. Moreover, the weather parameters have a negative correlation with the fire count suggesting an indirect or nonlinear relationship. The socioeconomic indicators such as GSDP, GSVA, GSDP\_CAP, and HDI are highly correlated with each other and date-related features, indicating potential multicollinearity in the dataset.

### **5.5 Model Training for Feature Selection**

## **5.5.1 Random Forest Regressor and Feature Importance**

To determine the feature importance and remove any noisy features in the data, a random forest regressor was trained using 5-fold cross-validation. The model was evaluated using the R² metric, and the average R² score obtained was 0.985, which demonstrates the model’s predictive effectiveness. After this, the model was trained on the entire dataset and feature importance values were extracted and plotted.

A graph with blue and white lines

AI-generated content may be incorrect.

**Figure 3: Feature Importance from Random Forest Regressor**

The feature importance plot shows that "DayOfYear" (0.608) and "Month" (0.267) are the pivotal features with significantly higher importance than the other features. These temporal features likely capture seasonal patterns critical to the stubble burning activity. “RH” (0.023), "AT"(0.023), "WD" (0.014), and "WS"(0.012) contribute moderately. Features that show minimal influence on the fire count include "GSVA" (0.011), "GSDP\_CAP" (0.009), "Day" (0.007)," and "Weekday"(0.006). The least important features are "PM10" (0.005), "HDI" (0.003), "Year" (0.003), "PM2.5" (0.003), "GSDP" (0.002), "SO2" (0.002), "NO2" (0.001), and "CO" (0.001). This indicates that the economic factors and certain pollutants have limited predictive power in the model.

## **5.5.2 Extra Trees Regressor and Feature Importance**

To cross-verify the feature importance of random forest and its validation, an extra trees regressor was trained using 5-fold cross-validation to measure its performance (R² score). The average R² score was 0.987. The model was then trained on the entire dataset to obtain the feature importance values and to plot them.

A graph with blue lines

AI-generated content may be incorrect.

**Figure 4: Feature Importance from Extra Tree Regressor**

The Extra Trees model reveals that "Month" (0.541) and "DayOfYear" (0.313) are the two most important features revealing that temporal patterns greatly influence the stubble burning activity. The moderately important features include "Day" (0.021), "AT" (0.017), "PM2.5" (0.016), "RH" (0.014), and "WS" (0.011). The socioeconomic features such as "GSDP\_CAP" (0.011), "Weekday" (0.010), "WD" (0.009), "GSVA" (0.008), "GSDP" (0.008), "HDI" (0.008), and "Year" (0.008) have relatively low importance. The least significant features include "PM10" (0.004), "CO" (0.001), "SO2" (0.001), and "NO2" (0.0005).

## **5.5.3 Feature Selection**

Based on the correlation heatmap and the feature importance values from the extra trees regressor (which had a slightly higher average R² score during 5-fold cross-validation than random forest regressor, i.e. 0.987007), we devised criteria for feature selection. The criteria used for feature selection was:

* High feature importance in extra trees regressor
* Moderate to strong correlation with firecount
* Low multicollinearity among factors

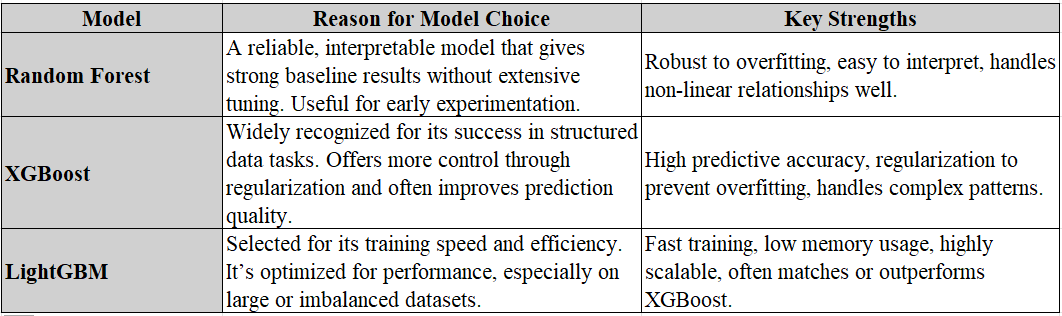
The features selected are listed below:

* DayOfYear
* Month
* AT (Air Temperature)
* RH (Humidity)
* WS (Wind Speed)
* WD (Wind Direction)
* PM25

### **5.6 Model Training**

This is a regression problem as we need to predict the number of fire counts. The data has features that have mixed scales, and there is moderate multicollinearity along with complex interactions (e.g., between weather and pollutants), so we made use of non-linear regression models, i.e., tree models. The tree models are known for handling non-linear data and multicollinearity.

Table 1 below briefly describes the models chosen to evaluate on the dataset. The models were trained using the final set of selected features from the preprocessed dataset. For hyperparameter tuning, a two-stage approach was adopted. First, a Randomized Search was performed across a broad hyperparameter space — including parameters such as learning\_rate, num\_leaves, and max\_depth. Based on the top-performing combinations from this step, a refined Grid Search was conducted over a narrower range to fine-tune the models. The best-performing configuration from either method was selected for training each respective model.



**Table 1: Comparison of Models**

Model training was performed using an 80/20 train-test split as per the standard practice. This ensures that the models had sufficient data to learn patterns while retaining enough unseen data to reliably evaluate performance. This balance helps minimize overfitting and provides a realistic estimate of how the model generalizes to new inputs.

Model performance was tracked during each iteration. The training and validation RMSE (Root Mean Squared Error) were recorded across boosting rounds to evaluate convergence and generalization performance. The following figures (Figure 5, 6, and 7) show the training curves obtained for LightGBM, XGBoost, and Random Forest respectively.

A graph with blue and orange lines

AI-generated content may be incorrect.

**Figure 5: LightGBM Training Curve**

A graph of training curves

AI-generated content may be incorrect.

**Figure 6: XGBoost Training Curve**

A graph with orange and blue lines

AI-generated content may be incorrect.

**Figure 7: Random Forest Training Curve**

The evaluation metrics used were:

* **Root Mean Squared Error (RMSE)**: Measures prediction error magnitude
* **R² Score:** Measures proportion of variance explained.

A table with numbers and text

AI-generated content may be incorrect.The following table shows the results obtained for each model:

**Table 2 Models Results**

The final model for deployment was selected based on performance metrics and interpretability. The Random Forest Regressor exhibited the highest RMSE (92.44), which was significantly worse than those of the other models (XGBoost: 51.40, LightGBM: 59.92) and was therefore rejected.

Although XGBoost achieved the lowest RMSE (51.4) and the highest R² score (0.99), it was not chosen due to its over-reliance on a single feature i.e. the "month". This is evident by the feature importance plots of the XGBoost and LightGBM regressors. The feature importance plot for XGBoost shows that it relies primarily on a single feature (i.e. month), to make predictions. This over-dependence suggests that the model may be overfitting or ignoring more meaningful drivers of stubble burning. On the contrary, LightGBM, relied on all the features in a balanced manner ensuring bter feature usage and generalization. Furthermore, XGBoost tended to overestimate fire counts when tested on real-world data. Thus, the LightGBM regressor proved itself to be the most reliable choice for deployment out of the three regressors.

Moreover, LightGBM also offered faster training times and lower computational complexity, making it suitable for real-time applications. Its robustness across multiple test cases further reinforced its suitability for deployment in the context of stubble burning prediction.

The feature importance plots for the three models are shown in Figures 8, 9, and 10 below. A graph with different colored bars

AI-generated content may be incorrect.

**Figure 8: LightGBM Feature Importance Plot**

A graph with text on it

AI-generated content may be incorrect.

**Figure 9: XGBoost Feature Importance Plot**

A graph with different colored bars

AI-generated content may be incorrect.

**Figure 10: Random Forest Feature Importance Plot**

As per these plots, the comparison table (Table 3) was created that further justifies the choice of LightGBM regressor as the final model.



**Table 3: Comparison of Feature Importance Plots**

### **5.7 Frontend Development**

The frontend of Stubblify was designed with the goal of providing users with a clean, intuitive, and informative interface to interact with the trained machine learning model.

## **5.7.1 Logo and Branding**

To build a cohesive and memorable identity, a unique logo was designed for Stublify, drawing inspiration from the central theme of stubble burning. The logo features an abstract fusion of leaves and a digital flame, symbolizing both nature and the environmental impact of agricultural fires.

A meaningful tagline, *“Spot to Stop”*, was incorporated into the dashboard to reflect the platform’s core mission: *detecting stubble burning incidents in real-time to enable timely intervention.* This aligns closely with the objectives of the Punjab Safe Cities Authority (PSCA), which is actively working to curtail smog and promote environmental sustainability.

**A logo with corn and fire

AI-generated content may be incorrect.**

**Figure 11: Stubblify Logo**

## **5.7.2 Model Integration**

Once the LightGBM model was selected as the final model for deployment, it was saved and stored as a pickl(.pkl) file. This allowed the trained model to be reloaded within the Django environment without the need for retraining.

## **5.7.3 API Integration**

To support real-time prediction, the app pulls current environmental data through two external APIs. The API keys were obtained for weather data (*OpenWeatherMap API)* and for air quality data *(API Ninjas)*.

## **5.7.4 Dashboard Design**

The dashboard layout was developed using **Django 4.2,** along with **HTML, CSS,** and **Bootstrap** for styling and responsiveness. The design emphasizes simplicity and clarity, following a visual theme consistent with the project’s logo and branding.

Users are required to log in to access the main prediction dashboard, where they can enter the name of a city to receive a stubble fire count prediction. The predicted number of fire counts is displayed alongside the corresponding risk level and real-time environmental data fetched via integrated APIs.

# **Chapter 6**

## DETAILED DESIGN AND ARCHITECTURE

### **6.1 System Architecture**

The **STUBBLIFY** system uses weather data to forecast stubble burning incidents. The system uses a Django dashboard to facilitate user interaction and incorporates machine learning models for prediction. This section gives a high-level overview of the system's functional subsystems, their roles, and how they work together to deliver the required services.

#### **6.1.1 Architecture Design Approach**

To enable scalability and maintainability, the system was designed in a modular fashion. Subsystems make up each of the system's main functions, and each subsystem is in charge of particular duties. The method relies on a client-server architecture for communication between the machine learning model and the backend (API-based data fetching) and a Model-View-Controller (MVC) architecture for the frontend (Django dashboard). The backend services were managed using a microservices-based architecture. To manage data retrieval, processing, prediction, and visualization, the main components are divided into discrete services. New features can be added with ease thanks to the modular design, which also makes maintenance easier.

#### **6.1.2 Architecture Design**

The STUBBLIFY system is made up of the following main subsystems at a high level.

**6.1.2.1 Data Fetching:**

This subsystem is in charge of using external APIs to retrieve meteorological data like temperature, humidity, wind speed, fire counts.

**6.1.2.2 Data Preprocessing**

This subsystem is responsible for cleaning, normalizing, and transforming data to use in machine learning models after it has been fetched. To make sure the data is prepared for precise predictions, it manages outliers, missing values, and feature extraction. It uses NumPy for data manipulation and Pandas.

**6.1.2.3 Machine Learning**

Predictions are the responsibility of this subsystem. The machine learning model LightGBM has been trained. Using preprocessed weather data, these models forecast the probability of stubble burning. It uses Pandas, LightGBM, and Scikit-learn.

**6.1.2.4 Prediction**

The machine learning models forecast the probability and severity of stubble burning. This subsystem provides the trained models with real-time data and generates the prediction. It uses Django and Python.

**6.1.2.5 Frontend Visualization**

To visualize the predictions produced by the backend models, the frontend subsystem is constructed using Django. Users can interact with the system, enter the desired region, and view predictions. To guarantee safe access, it additionally offers a user authentication system. It uses HTML, CSS, Django, and Python.

**6.1.2.6 Authentication**

This subsystem is in charge of overseeing user authentication. The visualization tools and prediction dashboard are only accessible by authenticated users. It guarantees individualized and safe system access. It uses Django.

#### **6.1.3 Subsystem Architecture**

The STUBBLIFY project's Subsystem Architecture is made up of a number of interrelated parts that are intended to manage data processing, user interactions, and real-time predictions. To predict fire counts, users can register, log in, and enter the name of their city using the User Interface. These inputs are processed by the backend, which also uses a pre-trained LightGBM model to predict fire counts and integrates with external APIs to retrieve real-time weather and air quality data. The user is then given a smooth and responsive experience as the system presents the results with a color-coded risk status.

### **6.2 Detailed System Design**

This section provides a detailed architecture breakdown of the system. The objective is to shed light on how the system's functionality is broken down into different parts and how these parts work together to produce the intended results. The function, duties, limitations, interactions, and integration of every software component into the overall system will be discussed.

#### **6.2.1 Components**

The STUBBLIFY system is made up of several components, each of which has a specific function. These elements fall into the following categories:

* Data fetching module
* Data preprocessing module
* Machine learning module
* Frontend module
* Authentication module

These elements guarantee the system's scalability, modularity, and ease of maintenance.

#### **6.2.2 Definition**

Every element has a particular function in the general operation of the system as described here:

**Data fetching:** Linking to outside APIs, this subsystem gathers real-time weather data required for forecasts. The aim is to offer current data to enable accurate stubble burning event projections.

**Data preprocessing:** Raw data from outside APIs is turned into a format fit for the machine learning models by a subsystem called data preprocessing. To guarantee the models run effectively, it standardizes the data, manages missing values, and cleans it.

**Model training:** Leveraging preprocessing subsystem data, this component trains machine learning models RandomForest, XGBoost and LightGBM to give real time prediction for areas at risk for stubble burning. The model project real time data fetching and then show predictions.

**Frontend development:** The frontend subsystem offers a user interface (UI) whereby users may view forecasts and track areas at risk of stubble burning. Making the system understandable and accessible requires this interface.

**User Authentication:** This class guarantees system access only for authorized users, so verifying the authentication subsystem. It keeps session states for the users and manages the login and registration procedures.

#### **6.2.3 Responsibilities**

Every element is intended with particular duties to guarantee flawless running:

* Data fetching module: This is responsible for fetching the data by using rest APIs
* Data preprocessing: This is responsible for checking that the data is cleaned and pre processed
* Machine learning module: This is responsible for training data set by using different models
* Frontend module: this is responsible for showing dashboard
* Authentication module: This is responsible for confirming that user is authorized

#### **6.2.4 Constraints**

Each component has some constraints that are described as follows:

* Data fetching should not be slow
* If the data is not preprocessed properly it will lead to incorrect results
* Overfitting and underfitting of model will affect the results
* Unauthorized persons will not be able to use it

#### **6.2.5 Composition**

Every component consists of several smaller elements cooperating to meet the objectives of the system:

* Data Fetching: API Client, Response Validator, Data Formatter.
* Missing Value Handler, Outlier Detection, Feature Extractor is the subsystem of data preprocessing.
* Training Module, Model Evaluation, Prediction Generator is the subsystem of machine learning model.
* User Interface for **Frontend Visualization**
* Login handler, registration handler, session manager makes up the **authentication subsystem.**

#### **6.2.6 Uses/Interactions**

Data fetching interacts with the weather and air quality APIs to retrieve real time data. Then preprocessing subsystem receives data and remove any outliers and preprocess the data. Machine learning model uses the preprocessed data for training and then the data is displayed on the frontend that is viewed.

#### **6.2.7 Resources**

A number of essential resources are needed to operate the Stubblify system effectively. Django web frame is used, and APIs are incorporated to retrieve real-time weather data and air-quality data. Scalability is guaranteed by the system design, which also includes strong error handling to reduce problems like deadlocks and erratic data retrieval. Moreover, it requires a stable internet connection to run completely because it uses external APIs in real-time to fetch the required dat.

#### **6.2.8 Processing**

The processing part of the STUBBLIFY project is in charge of managing the whole data flow, from data retrieval to prediction. The system predict stubble burning events by analyzing weather data. APIs are used to retrieve the data initially. Preprocessing involves removing outliers and handling missing values with imputation techniques. After that, training and prediction are done using machine learning model, LightGBM regressor. Based on weather patterns, the model predicts fire counts by analyzing the processed data. The system has error-handling mechanisms in place to detect problems like data fetching errors or model prediction failures, and it guarantees concurrency control to handle multiple requests during processing. In order to ensure that no out-of-date or unnecessary data is used in predictions, the system also cleans up temporary data.

#### **6.2.9 Interface/Exports**

The interface component of the project manages both integration with external services and the backend-frontend communication. Data APIs, model predictions, and visualization outputs are the main services this component offers. The data APIs act as a link to retrieve real-time weather and air quality data, which is subsequently processed and utilized by the pre-trained lightgbm regressor to make predictions. Fire counts are predicted and sent to the frontend for visualization. The data format and structure are defined by a set of parameters for every service.

Additionally, the system handles exceptions, such as unsuccessful API calls or incorrect data inputs. A Django dashboard is used by the frontend interface to show the machine learning model's output. The backend exports predictions in an approachable format, i.e. fire count display, while the frontend interacts with the backend to obtain processed data. The system guarantees seamless communication between various components while preserving performance and dependability by giving the user clear, meaningful outputs.

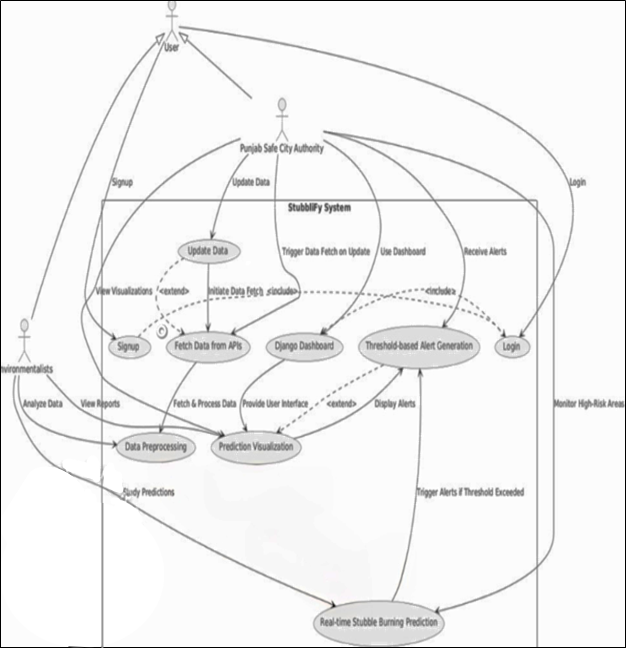
Users are required to log in to access the main prediction dashboard, where they can enter the name of a city to receive a stubble fire count prediction. The predicted number of fire counts is displayed alongside the corresponding risk level and real-time environmental data fetched via integrated APIs. Other features include the ability for users to sign up for a new account or log out securely. Static informational pages such as "About Us" and "Contact Us" have also been created to enhance user engagement and transparency.

#### **6.2.10 Detailed Subsystem Design**

This part briefly describes the complete functionality of our system by recognizing multiple users and their nature of usage with Stubblify. Here, each module is defined with diagrams for a brief conceptual model of our project.

The system diagrams illustrate the system components, their working and how they interact with one another. They represent the core functionalities of the system, the sub-components of the system modules and their integration with one another. Furthermore, they also describe the use cases and the working of the system to ensure that the use cases can be implemented.

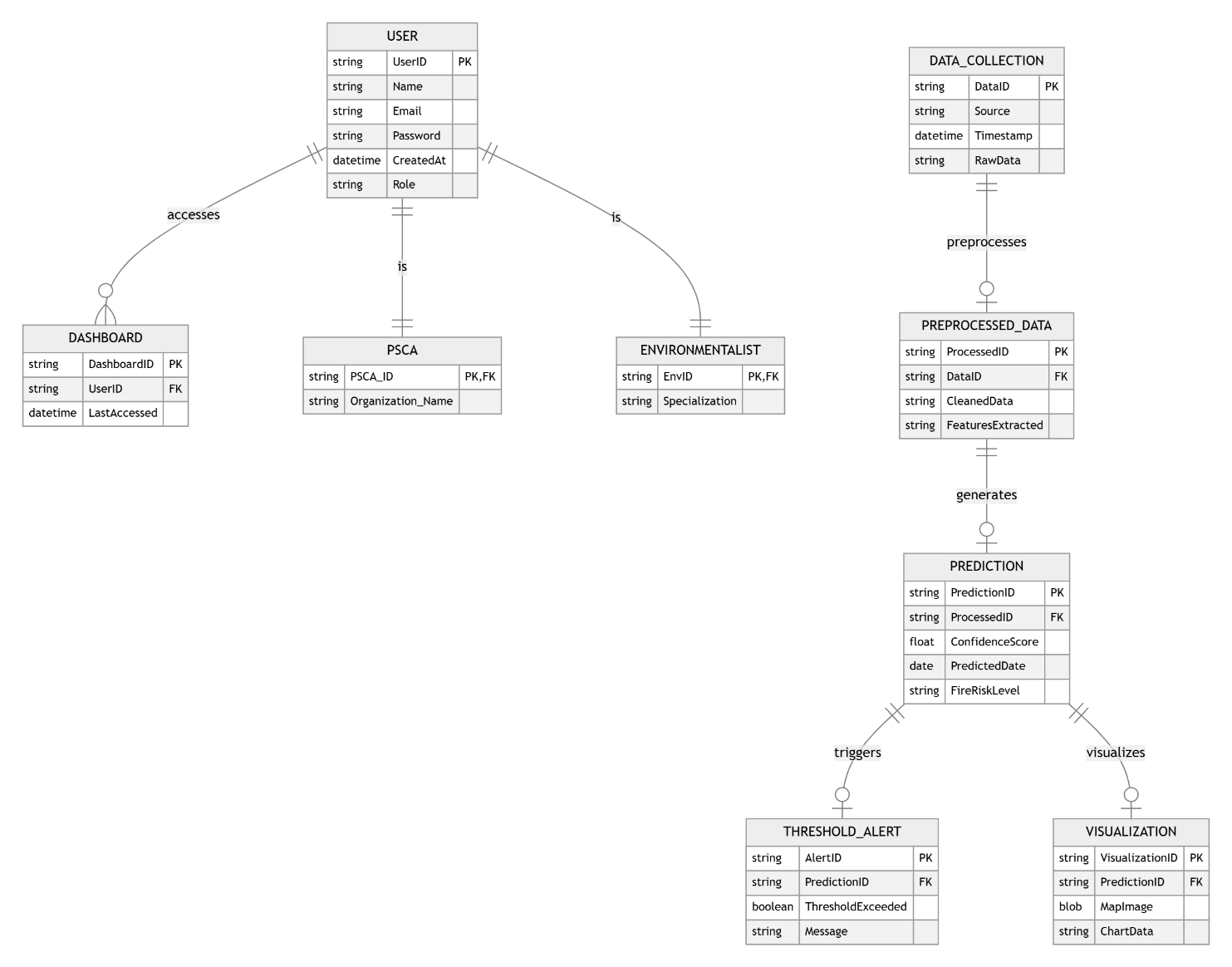
* + - 1. **Use Case Diagram**



**Figure 12: Use Case Diagram**

The use case flow is depicted in this diagram, where users and environmentalists engage with the system to register, log in, update data, and view visualizations. The system retrieves information from APIs, analyzes it, and produces predictions about stubble burning in real time.

* + - 1. **ER Diagram**

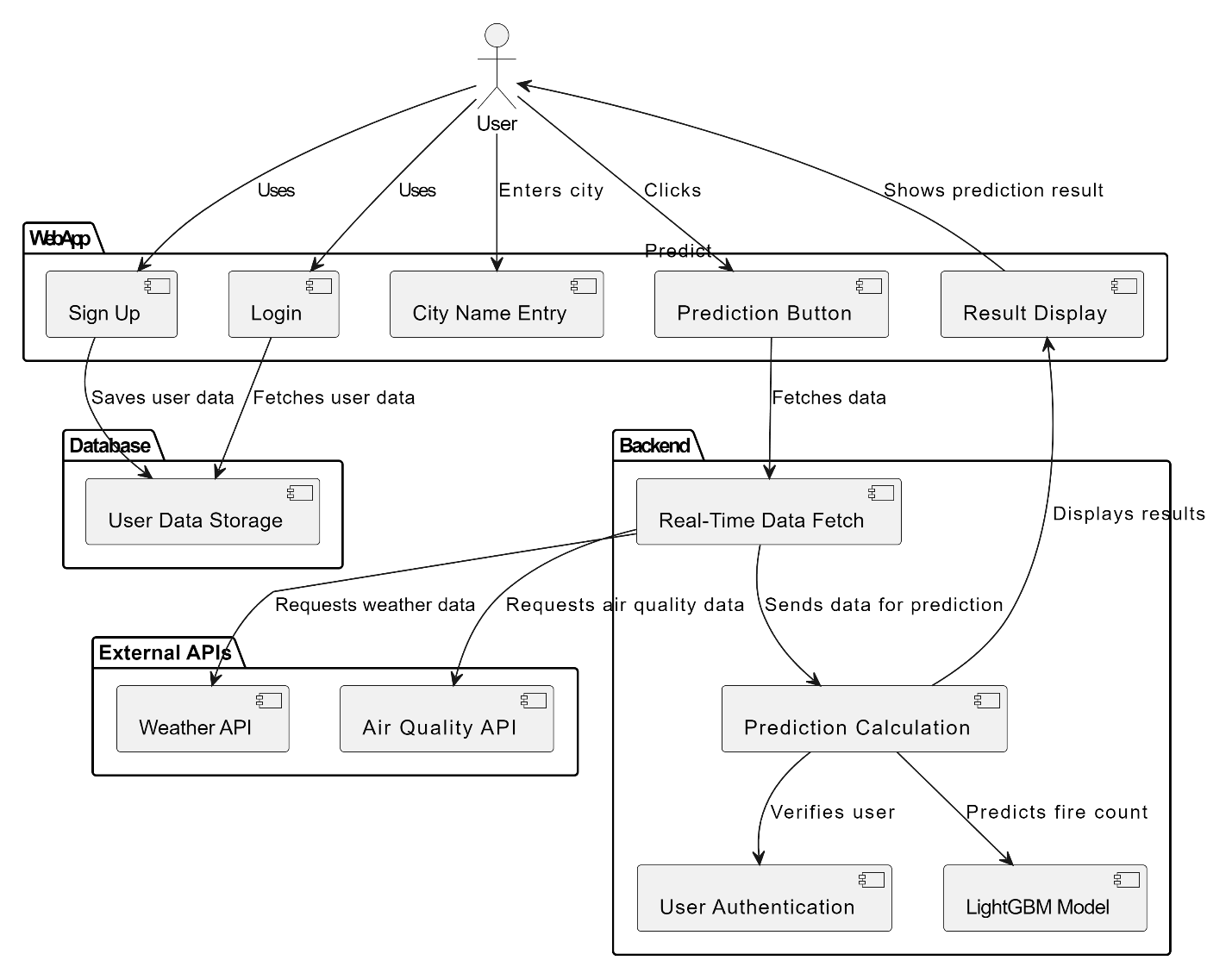


**Figure 13: ER Diagram**

This is the entity relationship diagram of stubblify. The relationships between users, environmentalists, data collection, preprocessing, prediction, and visualization are depicted in this diagram, which is a representation of the stubblify database schema. It demonstrates how the system preprocesses raw data, makes fire risk predictions, and uses threshold values to initiate alerts or visualizations to help predict stubble burning.

This ER diagram models key entities and their relationships in the Stubblify framework, supporting data collection, preprocessing, prediction, alerting, and visualization

* + - 1. **Architectural Diagram**



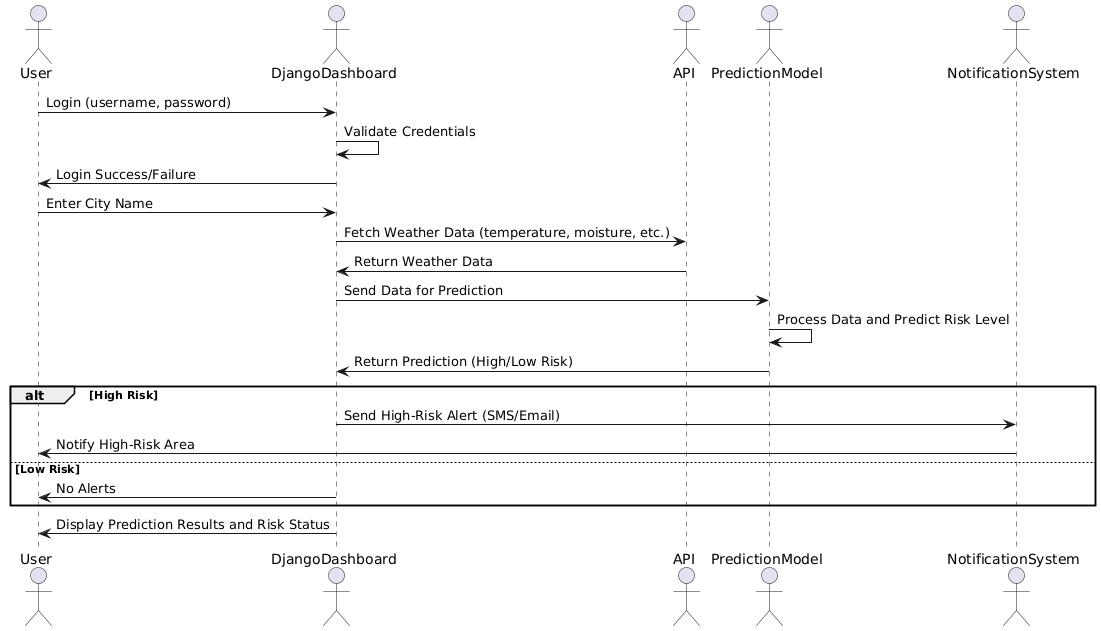
**Figure 14: Architectural Diagram**

This is an architectural diagram that shows how different components interact with each other. The different components of Stubblify work together to make it functional.

* + - 1. **Activity Diagram**

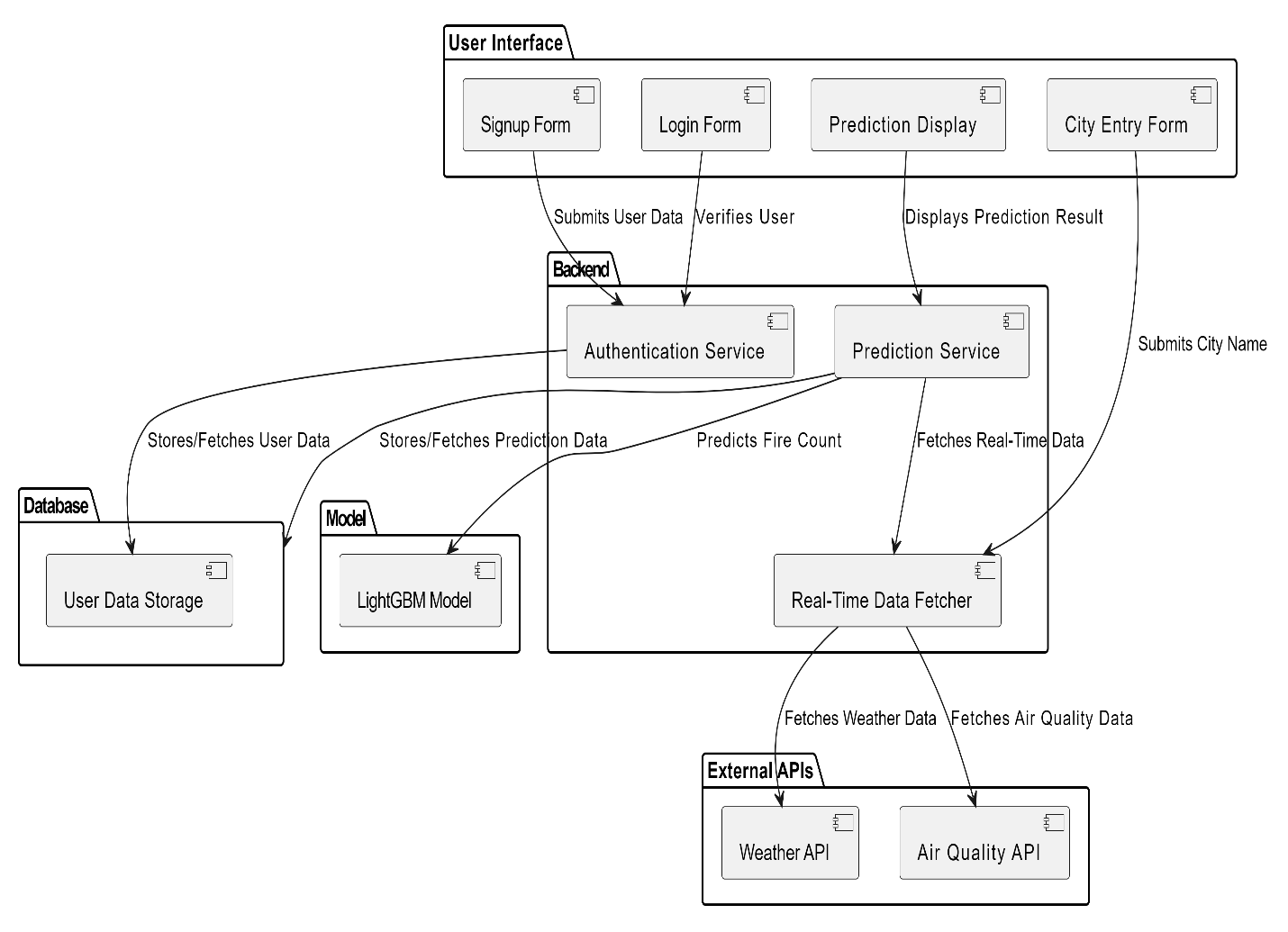
**Figure 15: Activity Diagram**

This is an activity diagram of the project that shows the user journey from logging into the system to seeing the results of prediction.

* + - 1. **Sequence Diagram**

**Figure 16: Sequence Diagram**

This sequence diagram describes the flow of actions in the project where user first logs in and after authentication, he can view the predictions.

* + - 1. **Component Diagram**

**Figure 17: Component Diagram**

The system component diagram is shown here. It demonstrates how the APIs handles requests and provides predictions based on user inputs, communicates with the Django Dashboard. User inputs, model predictions. Data preprocessing, model training, and prediction are further components of the machine learning model that handle, train, and show stubble burning incidents based on the weather and air quality data fetched by the APIs.

* + - 1. A screenshot of a computer screen

         AI-generated content may be incorrect.**State Machine Diagram**

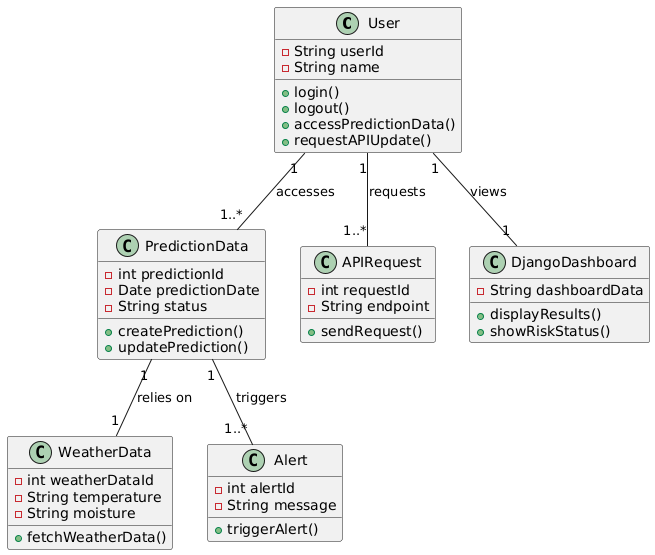
**Figure 18: State Machine Diagram**

This diagram illustrates the step-by-step workflow of our project.  
We begin by obtaining the data from Kaggle, followed by a thorough exploratory data analysi and data cleaning process to ensure consistency. Once the data is prepared, we perform feature selection to identify the most relevant variables that influence the prediction.

Next, three machine learning models are trained and evaluated using appropriate performance metrics (RMSE & R2) to determine which one performs best. After careful comparison, the model with the highest accuracy and reliability is selected.

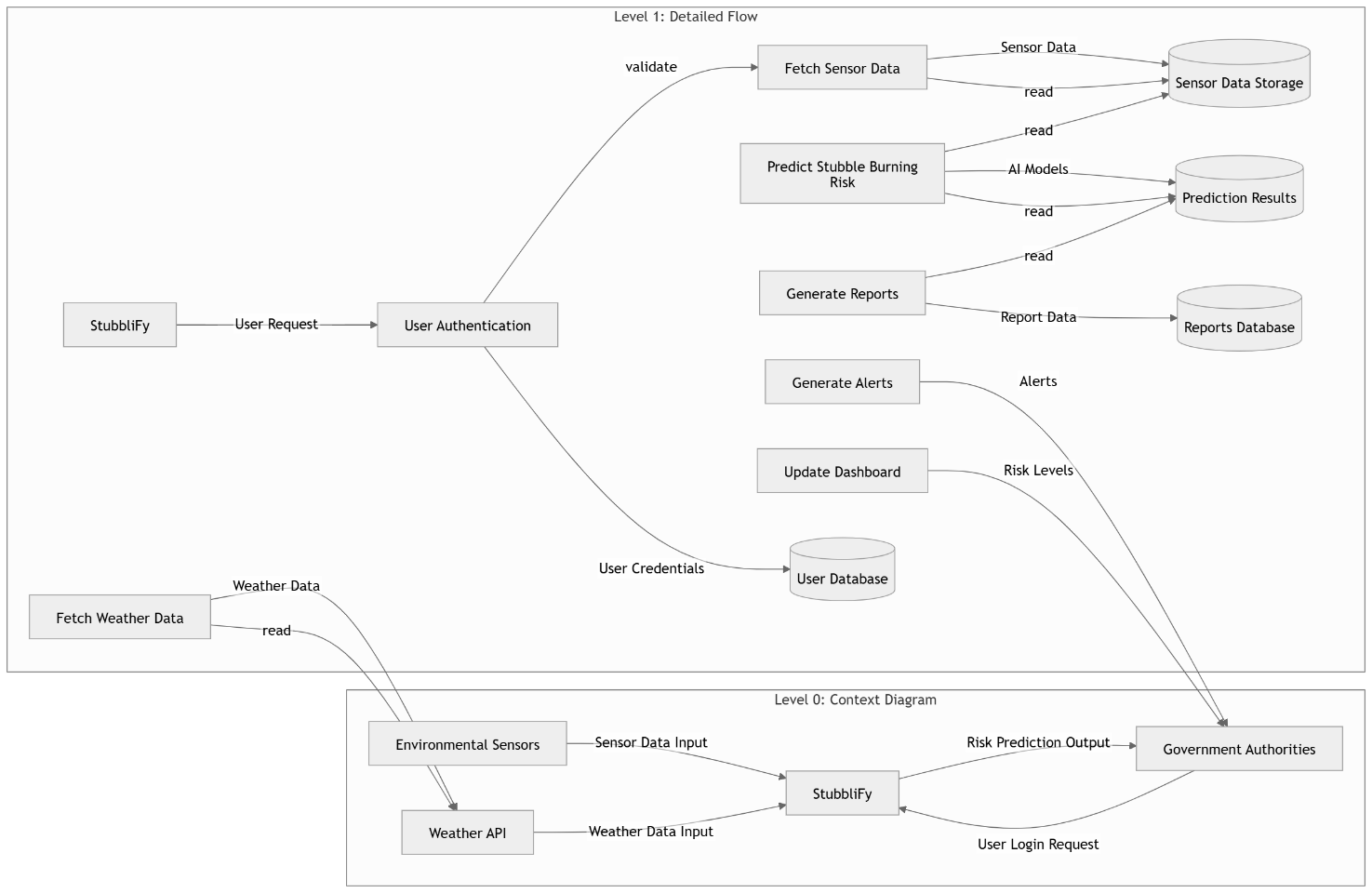
In the final phase, this optimized model is deployed and seamlessly integrated with a Django-based web dashboard, allowing users to interact with the system and receive real-time predictions in a

Each transition between states represents a logical flow of the project pipeline, ensuring that all components are systematically connected. This structured approach helped maintain clarity, modularity, and consistency throughout the development lifecycle.

* + - 1. **Class Diagram**

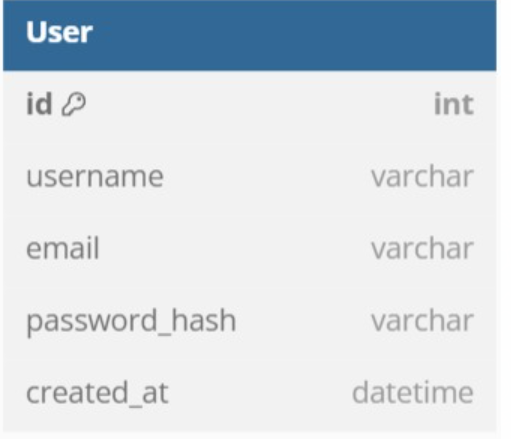
**Figure 19: Class Diagram**

This diagram displays system structure and the interactions between its various parts. The User class oversees prediction data, API request, and login. The Prediction data uses data to retrieve the required readings and stores prediction details. Furthermore, the API request class manages data requests and interfaces with external systems.

* + - 1. **Data Flow Diagram**

**Figure 20: Data Flow Diagram**

The system workflow is depicted in this diagram, starting with request processing and user authentication. It demonstrates how API data is retrieved, and processed in order to use models to nowcast the number of expected stubble burning incidents. Depending on the outcome of the prediction, the system then creates reports and modifies the dashboard. Government authorities receive the final predictions and risk assessments so they can act promptly by taking action in real-time.

* + - 1. **Database Diagram**

**Figure 21: Database Diagram**

Stubblify stores the login credentials of the users (specifically PSCA employees and agricultural management authorities) to ensure that only authorized personnel can access, monitor, and analyze stubble burning predictions.

# **Chapter 7**

## IMPLEMENTATION AND TESTING

**7.1 Implementation**

For the implementation of project we used combination of machine learning models and web frameworks to accurately predict stubble burning risks. Django is used to make a dashboard that is integrated with REST APIs to fetch real time weather data. For the model training we used LightGBM that is a powerful machine learning model and is light weight. The implementation of the system consists of following parts:

* Project Initialization
* User Authentication
* Dashboard integration
* Data fetching through REST APIs
* Model training
* Prediction
* Showing results on dashboard

**7.1.1 Tools**

For the implementation and development of project we used these tools

* VS code
* Google Colab
* Python
* Django framework

**7.2 Testing**

To ensure the smooth working of the project, we have used several testing methodologies to check that the project is working properly.

**7.2.1 Unit Testing**

After the development and integration of the modules, the system is tested. System is tested multiple times for proper functionality in matters of any data input, data processing or the output from the system for proper execution of the circumstances designed for testing. We used the integrated testing framework in Django to carry out unit testing. This required testing separate parts, including machine learning models, data preprocessing features, and data fetching APIs. Making sure every part worked properly when used alone.

All model-related tests included edge cases such as missing weather parameters or extremely high wind speeds. Frontend components, like prediction forms and error messages, were also tested independently for expected behavior. This ensured that any broken logic or invalid outputs could be caught early before integration.

**7.2.2 Integration Testing**

After the unit testing, integration testing is performed to check that all the components of the project work together as expected. In this testing flow to data from APIs is checked that the model accurately uses the data and predict accurate results that are then displayed on the dashboard.

The end-to-end pipeline, from API data fetching to model output and dashboard rendering, was tested repeatedly. Real-time prediction flow was validated using multiple test cities to check consistent behavior.Any mismatches in input-output mappings or API delays were corrected.

**7.2.3 Performance Testing**

We conducted performance testing to make sure the application can manage numerous requests without experiencing noticeable lags, since the system processes data in real-time and communicates with APIs. To assess the system's performance when many users are logged in at once, load testing was done on the Django dashboard.

Simulated concurrent user sessions were used to observe system stability under peak loads. We particularly monitored prediction latency, dashboard refresh speed, and API response time.

**7.2.4 Testing Requirements**

The table below (Table 4) elaborates the functional requirements of the system along with the test cases designed to ensure that the test requirements were satisfied.

|  |  |  |
| --- | --- | --- |
| **SR No.** | **Functional Requirements** | **Description** |
| FR-01 | Registration | User will create an account |
| FR-02 | Login | This will provide authentication to the user and users can view predictions after logging in |
| FR-03 | Dashboard | It will display results of the predictions and is viewed after authentication |
| FR-04 | Real Time data Fetching | The datawill be fetched from the APIs to perform predictions |
| FR-05 | Pre-Trained Model | Themodel is trained by using LIGHTGBM regressor to provide predications |
| FR-06 | Predictions | Prediction results are displayed on the dashboard |
| FR-07 | Logout | Users can logout anytime |

***Table 4: Testing Requirements***

**7.3 Test Cases**

**7.3.1 Registration**

|  |  |
| --- | --- |
| **Test case ID:** | Registration form\_User\_01 |
| **Version:** | 1.0 |
| **Reference** | Registration |
| **Purpose** | User wants to register |
| **Prerequisite** | User should have a stable internet connection |
| **Execution Description** | Users have to click on the Signup button, then the User will fill the registration form with required details. |
| **Expected Result:** | User Registered Successfully |
| **Actual Result:** | User Registered Successfully |
| **Pass/Fail:** | Pass |

**Table 5: Registration Test Case**

**7.3.2 Login**

|  |  |
| --- | --- |
| **Test case ID:** | Login\_User\_1 |
| **Version:** | 1.0 |
| **Reference** | Login |
| **Purpose** | User must be logged in to view predictions |
| **Prerequisite** | User have to open login page |
| **Execution Description** | User have to enter his/her valid email and password to successfully log in to the application |
| **Expected Result:** | User Logged in Successfully |
| **Actual Result:** | User Logged in Successfully |
| **Pass/Fail:** | Pass |

**Table 6: Login Test Case**

**7.3.3 Dashboard**

|  |  |
| --- | --- |
| **Test case ID:** | Dash\_1 |
| **Version:** | 1.0 |
| **Reference** | Dashboard |
| **Purpose** | To verify that the user that was authenticated can view dashboard |
| **Prerequisite** | User must be registered and logged in and dashboard must be deployed |
| **Execution Description** | After logging in, user will be directed to dashboard where he can view the results of prediction |
| **Expected Result:** | Dashboard loads successfully after logging in and it displays the results of the predictions |
| **Actual Result:** | Dashboard loads successfully and results are displayed |
| **Pass/Fail:** | Pass |

**Table 7: Dashboard Test Case**

**7.3.4 Real Time Data Fetching**

|  |  |
| --- | --- |
| **Test case ID:** | REALTIME\_1 |
| **Version:** | 1.0 |
| **Reference** | Real Time data fetching |
| **Purpose** | To verify that model can fetch real time data by using APIs and display prediction results |
| **Prerequisite** | User must be logged in and dashboard must be accessible |
| **Execution Description** | Login and access dashboard and observe real time data fetching and results must be displayed |
| **Expected Result:** | System fetches the data successfully |
| **Actual Result:** | System fetches the data successfully |
| **Pass/Fail:** | Pass |

**Table 8: Data Fetching Test Case**

**7.3.5 Pre-Trained Model**

|  |  |
| --- | --- |
| **Test case ID:** | MODEL\_1 |
| **Version:** | 1.0 |
| **Reference** | Pre-trained model |
| **Purpose** | To verify that the user selected pre-trained model and it will perform accurate predictions on the real time fetched data |
| **Prerequisite** | Pre-trained model must be saved and is loaded accurately during the run time |
| **Execution Description** | Model must be selected before the predictions and it will give accurate result |
| **Expected Result:** | Predictions run successfully using model that is pre-trained and output displays on the dashboard |
| **Actual Result:** | Predictions run successfully using model that is pre-trained and output is displayed on the dashboard |
| **Pass/Fail:** | Pass |

**Table 9: Pre-trained Model Test Case**

**7.3.6 Predictions**

|  |  |
| --- | --- |
| **Test case ID:** | PREDICT\_1 |
| **Version:** | 1.0 |
| **Reference** | Predictions |
| **Purpose** | To verify that the model is giving accurate predictions |
| **Prerequisite** | User is logged in and dataset is uploaded or fetched and a machine learning model is pre-selected |
| **Execution Description** | User logged in and click on the show prediction/output on the dashboard |
| **Expected Result:** | System runs prediction without errors and displays outputs on the dashboard |
| **Actual Result:** | System runs prediction without errors and displays outputs on the dashboard |
| **Pass/Fail:** | Pass |

**Table 10: Predictions Test Case**

**7.3.7 Logout**

|  |  |
| --- | --- |
| **Test case ID:** | LOGOUT\_1 |
| **Version:** | 1.0 |
| **Reference** | Logout |
| **Purpose** | User wants to logout |
| **Prerequisite** | User must be logged in before |
| **Execution Description** | User have to navigate to the logout button on the dashboard |
| **Expected Result:** | User successfully logged |
| **Actual Result:** | Logout successfully |
| **Pass/Fail:** | Pass |

**Table 11: Logout Test Case**

**7.3.8 Test Cases Results**

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | **Expected Result** | **Actual Result** |
| TC-01 | Pass | Passed |
| TC-02 | Pass | Passed |
| TC-03 | Pass | Passed |
| TC-04 | Pass | Passed |
| TC-05 | Pass | Passed |
| TC-06 | Pass | Passed |
| TC-07 | Pass | Passed |

**Table 12: Test Cases Results**

**7.4 Validation**

The Stubblify system has been tested, verified and implemented successfully and thus ensured that all the requirements listed in the software requirements specification are totally fulfilled. In case of invalid input equivalent error messages are displayed**.**

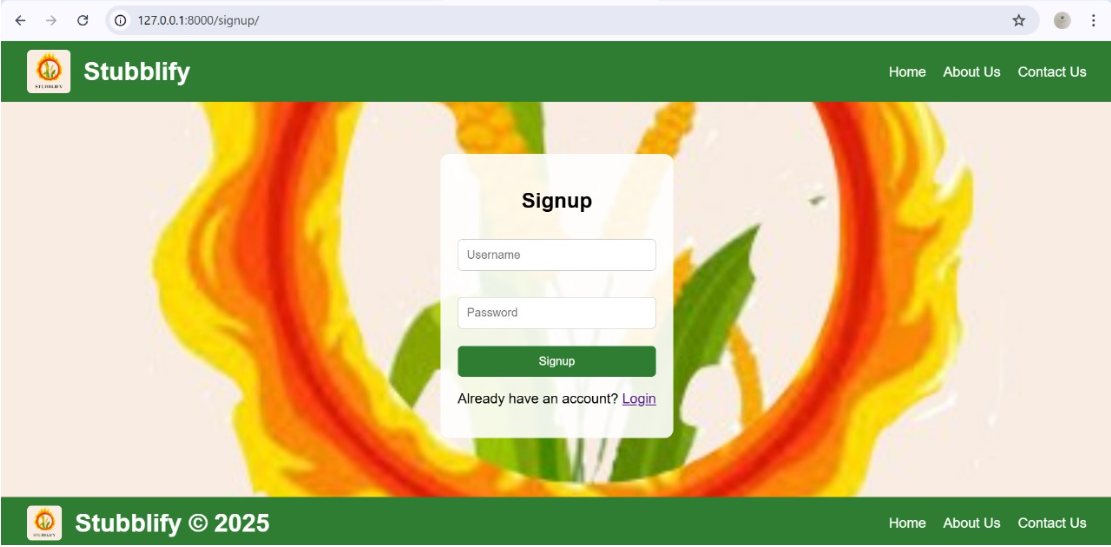
# **Chapter 8**

## RESULTS AND DISCUSSION

**8.1 Results of Test Evaluation**

Table 2 (Models Results) in chapter 5 of this document shows the results obtained for the three models evaluated on this dataset.

**8.1.1 Signup Page**



**Figure 22: Signup Page**

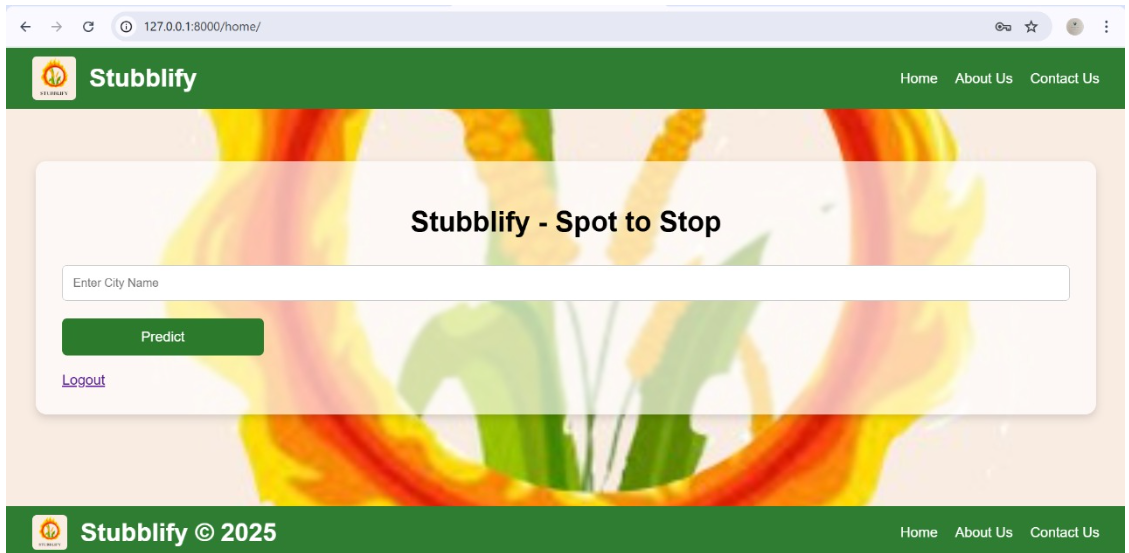
**8.1.2 Login Page**

A screenshot of a computer screen

AI-generated content may be incorrect.

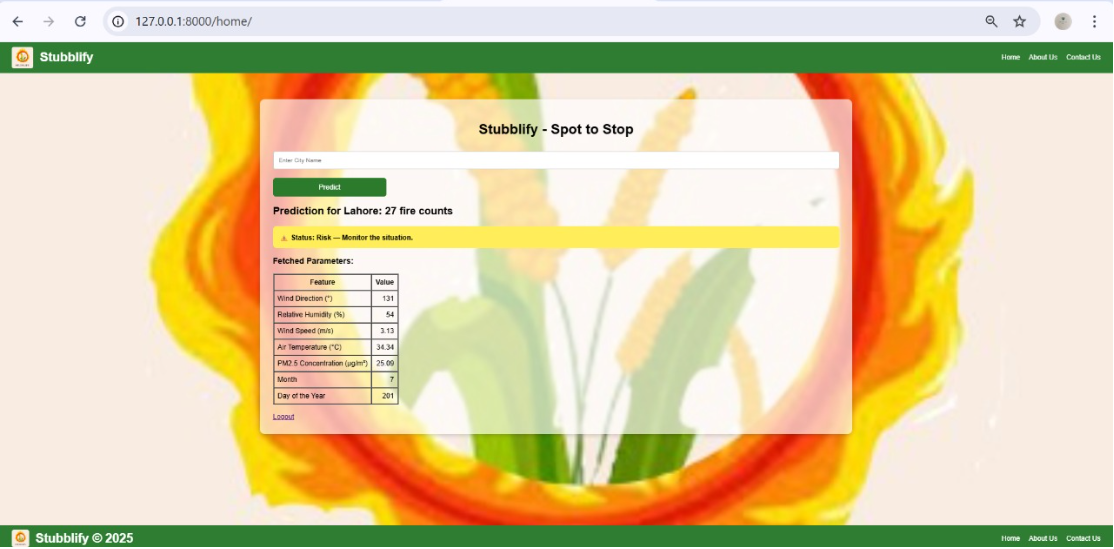
**Figure 23: Login Page**

**8.1.3 Dashboard**



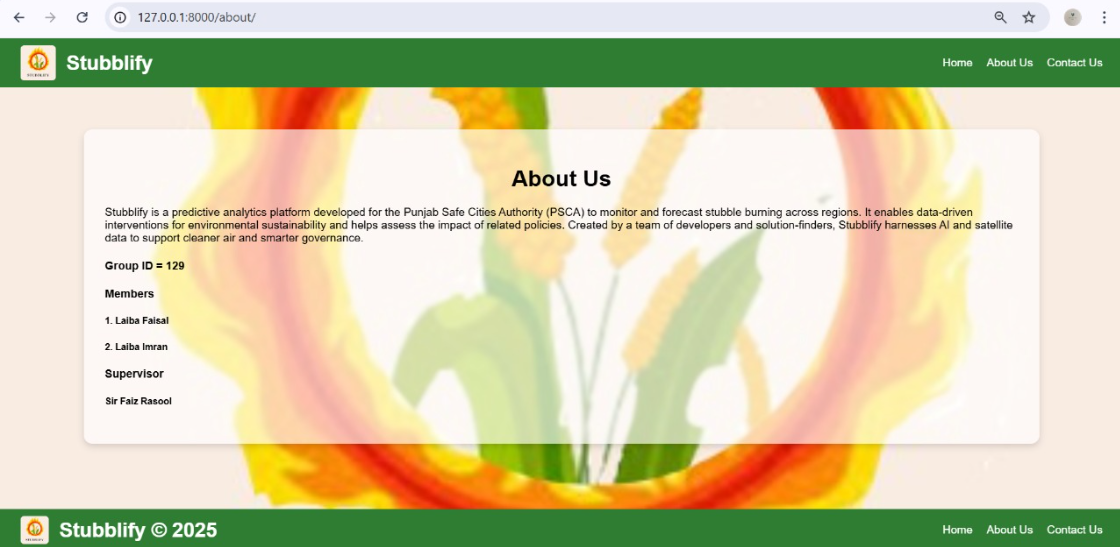
**Figure 24: Dashboard**

**8.1.4 Real Time Predictions**



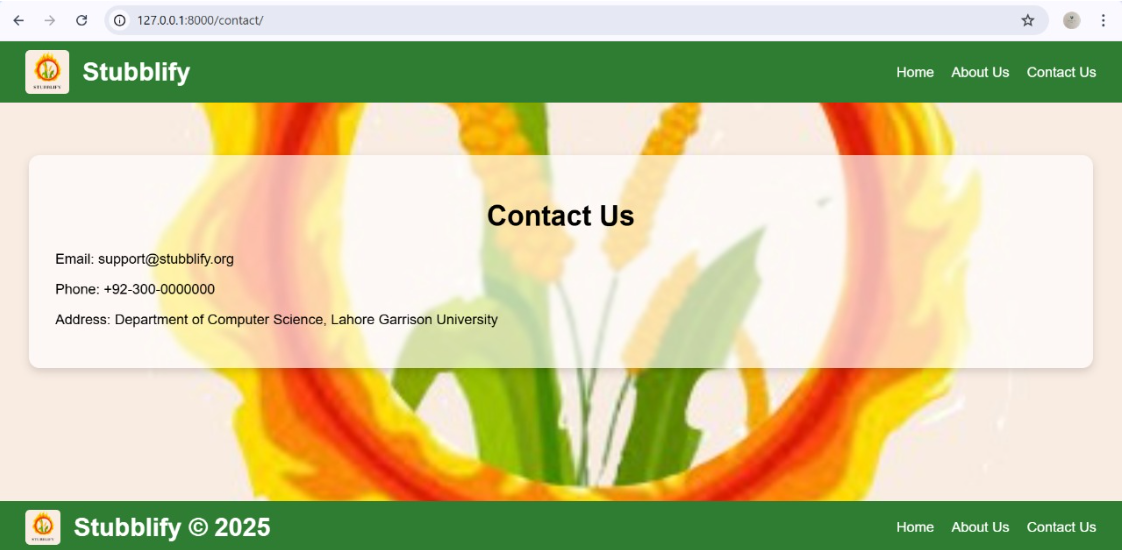
**Figure 25: Predictions**

**8.1.6 About Us**



**Figure 26: About Us**

**8.1.7 Contact Us**



**Figure 27: Contact Us**

# **Chapter 9**

## CONCLUSION AND FUTURE WORK

**9.1 Conclusion**

This report contains complete documentation of our project from its scope and implementation to its testing. With a very basic knowledge of training a model and connecting it with frontend that we had before this project, we tried our best to learn and develop an efficient system to solve a real-world problem. This project has not only helped us in learning and exploring advanced frameworks and libraries but also improved our collaborative and teamworking skills. Through this project we learnt current day market required frameworks and skills such as Django, model training and working with APIs.

**9.2 Future Work**

Many solutions have been tried to tackle the issue of stubble burning, but a major hurdle has always been the lack of localized data and tools. As we worked on this project, we began to see more clearly what could make these systems useful for people on the ground. Looking ahead, using forecasting techniques that are better suited to Pakistan’s agricultural landscape could make a real difference by helping authorities respond earlier and more effectively. During this journey, we also explored ways to make the data more relevant to our local context, setting the stage for stronger, more meaningful insights. By bringing in agricultural details like crop types and harvest cycles, there’s a real opportunity to improve the model’s accuracy even further. Although we’ve developed a web-based system for now, we see a lot of potential in turning it into a mobile app in the future, making it easier to use.

# REFERENCES

[1] S. C. a. G. S. M. I. Abdurrahman, "Stubble burning: Effects on health & environment, regulations and management practices," *Environmental Advances, vol. 2, p. 100011,* 2020.

[2] G. Singh, "The environmental impact of stubble burning," *International Journal of Science and Research Archive,* vol. vol. 12, no. no. 2, pp. 114-116, 2024.

[3] S. K. Das, "The effect of stubble burning and residue management on soil properties: A review," *International Journal of Plant & Soil Science,* vol. 36, no. 24, pp. 1-10, 2024.

[4] N. Koura, "Stubble Burning: Grassroot Analysis of Problems and Solutions in Moga District, Punjab.," *International Journal of Social Science and Economic Research,* 2024.

[5] P. G. e. al., "Satellite-retrieved stubble-burning activities in North India and their impact on air quality in Delhi," *Science of The Total Environment,* vol. 794, 2021.

[6] A. G. e. al, "Stubble Burning Detection Using Multi-Sensor and Multi-Temporal Satellite Data," in *IEEEXplore*, 2023.

[7] S. K. D. K. S. S. S. V. P. S. a. S. P. Mohite, "System and method for computing burning index score pertaining to crops," 2020.

[8] A. R. H. A. M. F. &. S. M. Y. Faridi, " Stubble Burning Detection and Prevention using Machine Learning and Blockchain.," in *.International Conference on Computing for Sustainable Global Development*, 2023.

[9] A. K. a. P. U. K. Chhapariya, "A fuzzy machine learning approach for identification of paddy stubble burnt fields," *Spatial Information Research,* vol. 29, no. 3, pp. 319-329, 2021.

[10] A. &. K. A. Singh, "Identification of Paddy Stubble Burnt Activities Using Temporal Class-Based Sensor-Independent Indices Database: Modified Possibilistic Fuzzy Classification Approach.," *Journal of The Indian Society of Remote Sensing,* vol. 48, no. 3, pp. 423-430, 2020.

[11] M. B. M. &. P. V. Asif, "Spatio-temporal variation of ambient particulate matter (PM10 and PM2.5) in Punjab: role of stubble burning and meteorological factors.," *Springer,* 2025.

[12] A. K. A. Palavesam Sarathamani, "Synergistic contextual information and individual sample as mean training approach: paddy stubble burning mapping. Environ Monit Assess," *Springer Nature,* 2025.

[13] V. B. Koutilya PNVR, "Mapping sugarcane residue burnt areas in smallholder farming systems using machine learning approaches," *Smart Agricultural Technology,* vol. 6, 2023.

[14] S. Banerjee, "kaggle," [Online]. Available: https://www.kaggle.com/datasets/shekharbanerjee/stubble-burning. [Accessed 23 3 2025].



