AGRISKILL: WEBAPP TO EXCHANGE AGRICULTURAL SKILLS IN RURAL COMMUNITY

Minor project-1 report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

By

HARIGOVIND P	(22UEAM0020)	**(VTU 21907)**
ADISH P	(22UEAM0004)	**(VTU 22055)**
NAVEEN KUMAR S	(22UEAM0042)	**(VTU 22700)**

Under the guidance of
Dr. R. LOTUS, M.Tech., PhD.,
ASSISTANT PROFESSOR

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

October, 2024

AGRISKILL: WEBAPP TO EXCHANGE AGRICULTURAL SKILLS IN RURAL COMMUNITY

Minor project-1 report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

By

HARIGOVIND P	(22UEAM0020)	**(VTU 21907)**
ADISH P	(22UEAM0004)	**(VTU 22055)**
NAVEEN KUMAR S	(22UEAM0042)	**(VTU 22700)**

Under the guidance of
Dr. R. LOTUS, M.Tech., PhD.,
ASSISTANT PROFESSOR

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

October, 2024

CERTIFICATE

It is certified that the work contained in the project report titled ”AGRISKILL:WEBAPP TO EXCHANGE AGRICULTURAL SKILLS IN RURAL COMMUNITY” by ”HARIGOVIND P (22UEAM0020), ADISH P (22UEAM0004), NAVEEN KUMAR S (22UEAM0042)” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor
Dr. R. Lotus
Assistant Professor
Computer Science and Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
October, 2024

Signature of Head of the Department
Dr. S.Alex David
Professor & Head
Artificial Intelligence & Machine Learning
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
October, 2024

Signature of the Dean
Dr. S P. Chokkalingam
Professor & Dean
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
October, 2024

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)
HARIGOVIND P
Date: / /

(Signature)
ADISH P
Date: / /

(Signature)
NAVEEN KUMAR S
Date: / /

APPROVAL SHEET

This project report entitled AGRISKILL:WEBAPP TO EXCHANGE AGRICULTURAL SKILLS IN RURAL COMMUNITY by HARIGOVIND P (22UEAM0020), ADISH P (22UEAM0004), NAVEEN KUMAR S (22UEAM0042) is approved for the degree of B.Tech in Artificial Intelligence and Machine Learning.

Examiners	**Supervisor**
	Dr.R.Lotus, M.Tech.,PhD.,

Date: / /
Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our **Honorable Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (Electrical), B.E. (Mechanical), M.S (Automobile), D.Sc., and Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, for her blessings.

We express our sincere thanks to our respected Chairperson and Managing Trustee **Mrs.RANGARAJAN MAHALAKSHMI KISHORE,B.E., Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, for her blessings.**

We are very much grateful to our beloved **Vice Chancellor Prof. Dr.RAJAT GUPTA,** for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. S P. CHOKKALINGAM, M.Tech., Ph.D., & Associate Dean, Dr. V. DHILIP KUMAR,M.E.,Ph.D.,** for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Professor & Head, Department of Artificial Intelligence and Machine Learning, Dr. S. Alex David, M.Tech., Ph.D.,** for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Dr.R.Lotus, M.Tech.,PhD.,** for her cordial support, valuable information and guidance, she helped us in completing this project through various stages.

A special thanks to our **Project Coordinator Dr.B.Prabhu Shankar,Associate Professor, Ph.D.,** for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

HARIGOVIND P	**(22UEAM0020)**
ADISH P	**(22UEAM0020)**
NAVEEN KUMAR S	**(22UEAM0042)**

ABSTRACT

Agriskill is a web app developed to address the manpower shortage in the agricultural sector by connecting landowners with skilled professionals. Traditional methods of finding agricultural workers are often inefficient and time-consuming, which can lead to reduced yields and economic losses. Using a sophisticated match-finding algorithm that combines TF-IDF with k-nearest neighbors, Agriskill effectively matches users based on their specific requirements. This approach leverages document retrieval properties to ensure relevant connections between landowners and skilled workers. By providing a user-friendly platform, Agriskill aims to create an efficient communication channel that simplifies the process of finding qualified help, ultimately reducing the barriers that lead many farmers to leave the industry. Through this initiative, Agriskill seeks to enhance agricultural productivity and support sustainable farming practices in the community.

Keyword:

Agricultural manpower, skill matching, landowners, skilled professionals, web application, match-finding algorithm, TF-IDF, k-nearest neighbors, cosine similarity, document retrieval, communication platform, agricultural productivity, sustainable farming, user-friendly interface, efficient connectivity, skill requirements, workforce optimization, rural development, technology in agriculture.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
JS	Java Script
SQL	Structured Query Language
TF-IDF	Term Frequency-Inverse Document Frequency
K-NN	K-Nearest Neighbour
NOSQL	Not Only SQL

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

Agriskill is the new modern platform that bridges the growing gap that agricultural workers are facing today. The primary purpose is to connect skilled workers with farmers who require assistance in cultivating their land. Agricultural practices are changing, and more and more efficient and scalable workforce solutions are being sought; Agriskill bridges the gap using advanced technology.

Agriskill provides the interface between the farmers that require skilled laborers and those laborers in search of farm-based work. Profiles and requirements are the input through which advanced algorithms, processing-the TF-IDF, coupled with k-nearest neighbors (k-NN), pass this information. This gives optimal matches of the workers and for whom they can toil. Agriskill shows a streamlined, accurate, and efficient solution with actionable insights and opportunities both for farmers and for the laborers.

Expert matching is a feature in this system that optimizes connections between users and experts using topic relevance and location so that users get quality advice and reduce response times, making it possible to solve the problems arising in agricultural practice in a timely manner. Traditional methods of finding an agricultural worker are usually very slow and rely solely on manual searches, which often prove inefficient, especially for regions with high labor demand. Agriskill offers a much more efficient and scalable interface for finding skilled labor by farmers and vice versa, thereby increasing productivity and cooperation between farmers.

1.2 Aim of the project

The primary goal of the AgriSkill web app is to establish a reliable and efficient platform that connects skilled agricultural workers with farmers in need of such services. Using modern technologies like TF-IDF and k-nearest neighbors (k-NN), AgriSkill aims to streamline the hiring process in the agricultural sector, which has increasingly become in demand due to the growing need for skilled labor. The project will improve accessibility by having a user-friendly interface where farmers and skilled workers can easily create profiles and outline their needs. Additionally, it aims to boost accuracy in matching by using advanced algorithms that make the connection relevant and data-driven. The platform fosters real-time communication and collaboration, helping users solve problems with greater urgency. Furthermore, the technology is designed to scale to accommodate higher data volumes and increased user interaction without loss of performance, making it a scalable and feasible solution in rural settings with limited infrastructure.

AgriSkill envisions the promotion of agriculture through increased yields, quality, and efficiency to contribute to food security and economic development in the industry. By empowering local communities and enhancing knowledge sharing, AgriSkill seeks to build resilience against the challenges faced in agriculture. The platform also aims to stimulate local economies by creating job opportunities, ultimately leading to sustainable development in rural areas. Through continuous engagement and feedback, AgriSkill plans to evolve and adapt, ensuring that it meets the changing needs of farmers and workers alike. Ultimately, the project aspires to create a thriving ecosystem where agricultural practices are optimized, benefiting not only individual users but also the agricultural sector as a whole.

1.3 Project Domain

AgriSkill is the online platform meant to unite farmers and agriculture workers with the aim of sharing knowledge and learning among themselves. The program empowers rural communities to make practical farming skills from crop management to livestock care as well as access to the expert advice that will introduce fresh ideas and innovative solutions. AgriSkill uses advanced technologies like TF-IDF and k-nearest neighbors (k-NN) to enhance the collaboration. It lets farmers who are connected to the network find solutions together, embracing better farming techniques in terms of productivity and efficiency. It also helps with the problem of labor scarcity in agriculture because the communities seek help from one another. The farmer learns new skills from his neighbors; therefore, he will depend less on external help. AgriSkill would develop strong local farming practices, encourage entrepreneurial opportunities, increase the economy in rural areas, create employment, and promote sustainable agriculture for the future. Therefore, the platform aims to build a stronger, more connected farming community where knowledge and opportunities can be easily shared to improve the general resilience of the agricultural sector. It encompasses a wide range of functionalities aimed at enhancing the agricultural ecosystem. It integrates a marketplace feature that allows farmers to post job listings, specifying their exact needs and requirements, which skilled workers can browse and apply for. Additionally, the platform incorporates a feedback and rating system, promoting accountability and trust within the community by enabling users to review and rate their experiences with each other. AgriSkill also aims to provide resources such as articles, tutorials, and expert advice, offering users valuable information to improve their farming practices and knowledge base. Furthermore, by utilizing data analytics, the platform can identify trends and common challenges faced by users, allowing for tailored support and resources to be developed, thereby continuously improving user experience and agricultural outcomes.

1.4 Scope of the Project

The AgriSkill project deals with the vast scope towards transforming the agricultural landscape. Use of technology helps the integration of farmers and skilled agriculturists for fruitful outputs. The project works centrally on collaboration, developing greater knowledge sharing, and eventually improving agricultural practices among residents in rural communities. Creating a vibrant online community so that users can come on board to discuss various questions, share experiences, or learn from each other falls under the goals of the AgriSkill initiative. Crop management, pest control, and other sustainable practices are areas of skills exchange. Indeed, this implies a continuous learning and adaptation culture. Using the strength of advanced algorithms like TF-IDF and k-NN, the platform offers expert matching services to all users based on specific needs or local challenges

AgriSkill helps overcome labor shortages by providing a marketplace for skilled labor. Farmers can easily access local workers and thereby enhance the strength of local economies. Through user interactions, the project seeks to gain data-driven insights into agricultural trends, thus making the platform features more advanced in the future. AgriSkill seeks to make rural areas sustainable and economically viable by encouraging best practices, job opportunities, and self-reliance. The scalable platform is going to grow and shift with demands to allow for a more resilient, sustainable, and prosperous agricultural community.

Chapter 2

LITERATURE REVIEW

2.1 Literature Review

Samya Pathirage and Athula Ginige (2020) [1] proposed an online platform that enables farmers to access and apply agricultural knowledge within their regions while promoting permaculture practices. The platform encourages collaboration between farmers and experts by using technology for knowledge exchange. This system aims to increase agricultural productivity and sustainability through the exchange of farming techniques and innovations tailored to the specific environmental conditions of each region.

Siddhartha Paul Tiwari (2021) [2] highlighted the role of information and communication technology (ICT) in bridging the gap between research and farming practices. By providing timely access to relevant information, ICT tools help farmers make better decisions, ultimately improving productivity and sustainability in agriculture. Tiwari emphasized the importance of integrating these tools to allow the real-time sharing of agricultural expertise and findings, ensuring that research insights reach farmers in practical and applicable ways.

Sidi Sanyang, Sibiri Jean-Baptiste Taonda, Julienne Kuiseu, N'Tji Coulibaly, and Laban Konat (2021) [3] discussed the shift in agricultural research across Africa, noting how innovation is critical for fostering collaboration between farmers, researchers, and policymakers. Their work outlined the need for platforms that bring diverse stakeholders together to address agricultural challenges and develop sustainable farming practices. By fostering such collaborations, these platforms can help farmers implement innovative methods tailored to local environmental and socioeconomic contexts.

Giulio Ermanno Pibiri and Rossano Venturin (2019) [4] reviewed methods for optimizing data retrieval systems, such as compressing inverted indexes. These techniques enhance storage and query performance, which is crucial for large-scale platforms like AgriSkill. Efficient data retrieval allows farmers to quickly access relevant knowledge, improving the system's usability and relevance for users.

Victor Lempitsky (2020) [5] introduced the inverted multi-index method, which improves search performance in large-scale environments. This method reduces search time and increases accuracy, making it particularly useful in applications where precise knowledge retrieval is essential. In the context of the AgriSkill platform, this approach would help match farmers with relevant agricultural advice more efficiently, ensuring that users receive tailored recommendations.

Cai-zhi Liu, Yan-xiu Sheng, and Yong-Quan Yang (2021) [6] demonstrated improvements in classification accuracy using the TF-IDF algorithm. Their work shows how this algorithm can identify key features in datasets, which can be applied to agricultural platforms to better match farmers' needs with the correct expert or solution. Their findings support the use of advanced algorithms for more accurate knowledge exchange.

Prafulla Bafna et al. (2022) [7] implemented TF-IDF for document clustering, exploring its effectiveness in identifying topic-based clusters in textual data. The study highlights TF-IDF's advantage in emphasizing distinct words within a corpus, facilitating meaningful clusters.

Dadgar et al. (2022) [8] introduced a hybrid TF-IDF and Support Vector Machine (SVM) approach for news classification. This model leverages TF-IDF's strength in capturing word frequency with SVM's capabilities in pattern recognition, proving beneficial in classifying news articles.

Liang et al. (2020) [9] focused on text feature extraction by combining TF-IDF with semantic associations, which improves clustering accuracy by factoring in contextual meanings of terms beyond raw frequency counts.

Alfirna Rizqi Lahitani et al. (2015) [10] discussed the application of cosine similarity for measuring similarity in online essay assessments. This study showcases cosine similarity's role in determining document relatedness by examining vector angles and distances, beneficial in educational evaluation contexts.

Liming Zheng et al. (2021) [11] explored cosine similarity for line protection in large-scale wind farms, an unconventional application where the method was adapted to compare time series data for fault detection, demonstrating k-nn similarity's versatility.

Lailil Muflikhah and Baharum Baharudin (2018) [12] applied k-nn in document clustering, proving effective in grouping documents with high semantic similarity.

Hakim et al. (2021) [13] developed an automated document classification system for Bahasa Indonesia news articles based on TF-IDF. This system addresses the challenges in non-English text processing, highlighting TF-IDF's adaptability across languages.

Dreuw et al. (2021) [14] implemented TF-IDF and k-nn for scientific document classification, focusing on abstracts, where their combination enabled accurate thematic grouping by emphasizing key terms within short text.

Yunanda et al. (2017) [15] proposed a recommendation system using TF-IDF and cosine k-nn on Microsoft News data, proving successful in recommending articles based on user preferences by comparing term-weighted vectors.

Snigdha et al. (2021) [17] developed a movie recommendation system using TF-IDF vectorization enhancing the personalization of movie suggestions by matching user profiles with movie content.

Salman et al. (2022) [18] introduced a co-occurrence-based cosine similarity feature extraction method, contributing to the refinement of similarity scoring in text mining tasks by focusing on frequently co-occurring term pairs.

Xu et al. (2017) [19] and Liang and Qian (2018) [20] focused on job recommendation systems. Xu et al. improved job matching through a collaborative filtering approach integrated with TF-IDF, while Liang and Qian developed a personalized system utilizing collaborative filtering and TF-IDF, thus tailoring job recommendations based on user profiles.

2.2 Gap Identification

Although there are so many papers that indicate the importance of agricultural technology and integration, still a wide gap exists about how such systems can be made applicable in real farm settings and their adaptation according to specific problems and situations farmers encounter. Most of the studies focus on data analysis techniques and methodologies that largely miss practical application to the fields of farming. The outcome is that the technological innovations towards better productivity in agriculture might not attain their full realization, considering the fact that the contexts for the farmer are of no relevance or application. Second, although several algorithms have recently been developed that look promising in improving data retrieval and classification, including TF-IDF and k-nearest neighbors (k-NN), there is almost negligible scientific research about how they could improve knowledge exchange and interdependence between the farmer and the expert.

More importantly, though there is much discourse today about connectivity in the agricultural sector as an imperative, research to engage farmers in the development and use of such platforms is still lacking. Mismatches between technological offerings and the needs of the farming community lead to low adoption and ineffective solutions. This gap can be bridged if the future research puts importance on the understanding of need and preference among the people residing in the rural communities. These kinds of participatory studies have been carried out not only on obtaining feedback from the farmers but also involving them at all the stages during the design and implementation of the technology initiatives.

Farmer-led approaches ensure that the efforts made by farmers regarding technology and knowledge-sharing initiatives are effective and efficient and directly aligned to the realities of agricultural practice. Closing this gap is important both for increasing the capacity of technological solutions to support sustainable agriculture and for productivity and resilience in rural areas. A focus on participatory approaches may help transform the agricultural technology landscape to better meet the needs of farmers. It will usher in an environment of co-creation of innovative solutions with farmers, ensuring more sustainable practices, increased yields, and greater economic stability in rural communities. In the long run, these research gaps will lead to significant contributions toward overall agricultural sector development, empowered farmers, and food security for generations to come.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

Currently, sharing knowledge with the farmers about agriculture is mainly based on the traditional ways. It has been done in form of agricultural extension services, workshops, and print media. These have been useful in reaching the services out to the farmers, but again there are many limitations. One of the limitations is that these services are limited in reach. This usually happens during the dissemination of such information in the rural areas where access to resources might be limited. Best practices and recent agricultural technologies may not reach farmers in good time; hence their adaptation and productivity is slowed down. Furthermore, relying on face-to-face workshops makes low turnout inevitable from scheduling conflicts, hard transport conditions, or ignorance about offered events. This old-fashioned approach does not also benefit from the gigantic worth of digital technology such as information exchange and cooperative learning may be carried out instantly through these sources.

The current systems lack interactive parts where farmers can get in touch with the experts and the peers. It can be stated that the knowledge is mostly channelled in one direction; therefore, this creates a gap through which farmers will go ahead to apply practices, not knowing how they will translate into effect. Valuable insights gathered in highly heterogeneous agricultural communities cannot be harvested and shared widely without a central framework. Hence, farmers are denied the opportunity of learning from each other; therefore, there will be inconsistency in the whole agricultural sector, coupled with less overall productivity. In a nut shell, these traditional methods do have merits, yet remain insufficient to handle the dynamic needs within the modern agricultural landscape.

3.2 Problem statement

Significant challenges in new systems of agricultural knowledge-sharing prevent farmers from getting both relevant and timely information. Through methods like workshops, much of this support becomes a poor source for farmers who would readily embrace new agricultural practices and the related technologies. Generally, these methods are one-way; therefore the opportunities farmers have in order to interact with experts or ask questions are little. More importantly, these systems outreach is limited, especially for rural based farmers, where many of them lack the resources or latest updates on information. Such a scenario leads to continuing ignorance for agriculture, which infers direct low productivity and discontinuity in agricultural sustainability.

The proposed system aims to offer a dynamic online platform that will provide real-time interaction among farmers, experts, and their peers in order to address these challenges. This is how it facilitates access through modern technology to lots of information and practical resources by farmers catering to their needs in particular. The platform forums, QA sections, and video tutorials will all be very interactive features that will enable the user to engage with some of the most knowledgeable sources so that collaborative learning and knowledge exchange take place. Additionally, the platform will recommend personalized inputs based on the individual's preference and conditions in various regions, thus giving the most relevant and actionable insights to the farmer. It is through this gap that the proposed system improves the decision-making ability of farmers to create a more sustainable agricultural environment by bridging the gap between research and practical applications.

3.3 System Specification

The AgriSkill web application would provide a comprehensive online platform to share knowledge, collaborate, and enhance agricultural practices among farmers, agricultural workers, and experts. The system thus will include user registration and authentication which allows users to create accounts as either a farmer, expert, or agricultural worker, offering personalized experiences to the users. It will promote knowledge sharing, facilitating users from uploading any content, such as articles, videos, tutorials, and best practice guides, segregated under categories like crop management and sustainable practices. Interactive features for the users include forums and discussion boards in which they can pose questions and engage with one another. There will also be a real-time chat capability for urgent matters. Meanwhile, they will enable farmers to book virtual appointments with experts who will provide them with personalized advice supported by an online resource library containing research papers and farming tools. Nonfunctional Requirements The focus is on providing an easy-to-use and navigate interface for the users; high performance pertaining to the multiple users; and the highest security over the user data. It will be designed to grow in scale, accessible for people with disabilities, and have offline capability to counter connectivity problems that rural areas possess. A frontend technology like HTML, CSS, and Python with frameworks like Express or Django on the backend. User data and interactions will be stored safely in a relational database, such as PostgreSQL, or a NoSQL variant like MongoDB. The application will be hosted on a cloud platform like AWS or Google Cloud to ensure reliability and scalability. AgriSkill generally aims to enhance the knowledge sharing and collaboration of rural farming communities toward better-informed decision-making by farmers that contributes to better farming practices.

3.3.1 Hardware Specification

Server requirements:

- Processor: Multi-core processor (Intel Xeon Gold 5118 or AMD Ryzen 7 5800X)
- RAM: Minimum 16 GB DDR4 (expandable to 32 GB or more for scalability)
- Storage: Minimum 200 GB SSD for fast data access (with options for additional cloud storage as needed)
- Network: High-speed internet connection (minimum 1 Gbps bandwidth)
- Operating System: Linux-based OS (Ubuntu 20.04 LTS or CentOS 8)
- Backup: Regular backup solutions (cloud-based and local backups)

Client requirements:
Desktop/Laptop:

- Processor: Intel Core i5 (10th generation or newer) or AMD Ryzen 5 (4000 series or newer)
- RAM: Minimum 8 GB DDR4

- Storage: Minimum 256 GB SSD
- Operating System: Windows 10/11, macOS 10.15 or later, or Linux
- Browser: Latest versions of Chrome, Firefox, or Safari

Mobile Devices:
Smartphone/Tablet:

- Processor: Quad-core processor (Qualcomm Snapdragon 678 or equivalent)
- RAM: Minimum 4 GB
- Storage: Minimum 64 GB internal storage
- Operating System: iOS 14 or later, Android 10 or later
- Screen Size: Minimum 5.5 inches for optimal user experience

These specifications ensure that both the server and client devices are capable of providing a smooth, efficient, and responsive user experience for the AgriSkill web application.

3.3.2 Software Specification

Frontend Technologies:

- HTML5: For creating structured and semantic web pages.
- CSS3: For styling and layout; recommended frameworks:
  - Bootstrap 5: For responsive design.
- JavaScript: For interactive elements and enhanced user experience.

Backend Technologies:

- Programming Language:
  - Python (v3.8 or newer): For server-side application development.
- Framework:
  - Django (v3.2 or newer): A high-level Python web framework that supports rapid development and clean design.
- Database:

    - MySQL (v8 or newer): An alternative relational database option.

- Matching Algorithm:
    - TF-IDF: Implementation for calculating the importance of terms in documents.
    - KNN: Uses a specified number of nearest neighbors to classify or predict data points based on the similarity (e.g., distance) to other labeled points in the dataset.

- Development Tools:
    - Integrated Development Environment (IDE):
        * Visual Studio Code (latest version): For code editing, with support for extensions and debugging.
        * PyCharm (v2021.3 or newer): A powerful IDE specifically for Python development.
    - Version Control:
        * Git (latest version): For source code management and collaboration.
        * GitHub or GitLab: For hosting repositories and version control.

3.3.3 Standards and Policies

Google Colab

Google Colab is an online environment that provides an easy-to-use platform for coding, particularly in Python. It supports a wide range of machine learning and data science libraries. As a cloud-based platform, Colab allows developers to collaborate in real-time, share code, and perform complex computations without the need for extensive local resources. Colab also supports GPU and TPU acceleration, making it suitable for more intensive tasks like model training.

Standard Used: ISO/IEC 27017

Pycharm

PyCharm is an Integrated Development Environment (IDE) tailored for Python development. It provides powerful tools for writing, debugging, and testing

Python code. With support for Django, PyCharm is an excellent tool for developing the AgriSkill webapp's backend. It also integrates with Git and other version control systems, facilitating collaboration and ensuring that code is consistently backed up and secure.
Standard Used: ISO/IEC 27001

Chapter 4

METHODOLOGY

4.1 Proposed System

4.2 General Architecture

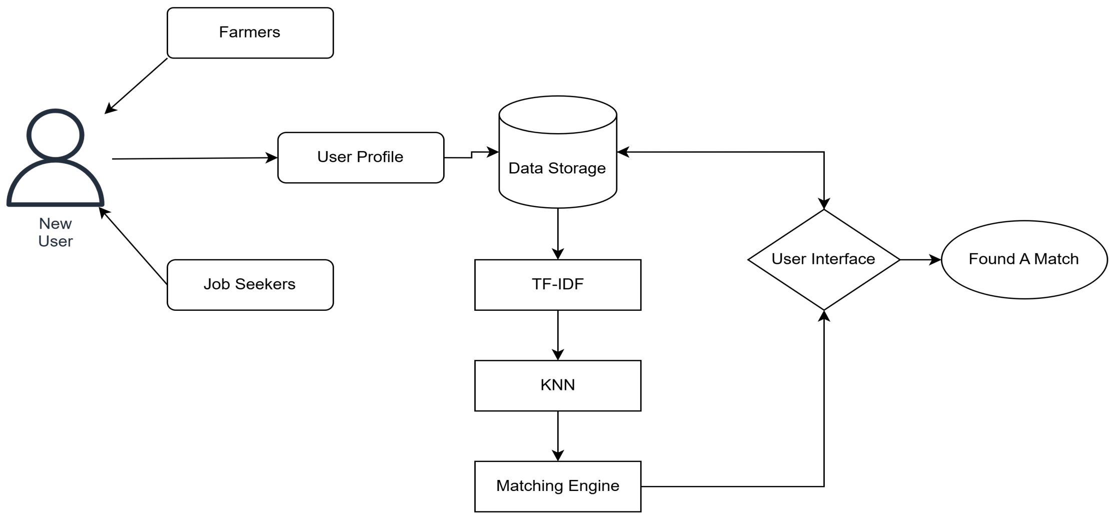


Figure 4.1: **Agriskill webapp**

This is the architectural diagram of the AgriSkill web application that is a comprehensive platform for the exchange of agricultural knowledge and skills among rural people. The application will present a user interface where the applicant can navigate and find suitable agricultural skills while allowing the farmer to present their expertise.

The system has an advanced matching engine using TF-IDF and k-NN algorithms that ensures pairing the seeker with appropriate farmers according to the seeker's requirements of skills and farmers' offerings. The TF-IDF technique improves the relevance of matches by analyzing the importance of specific terms within the context of the users' profiles, while the k-NN algorithm further refines the process by considering proximity in a multi-dimensional space to identify the closest matches.

4.3 Design Phase

4.3.1 Data Flow Diagram

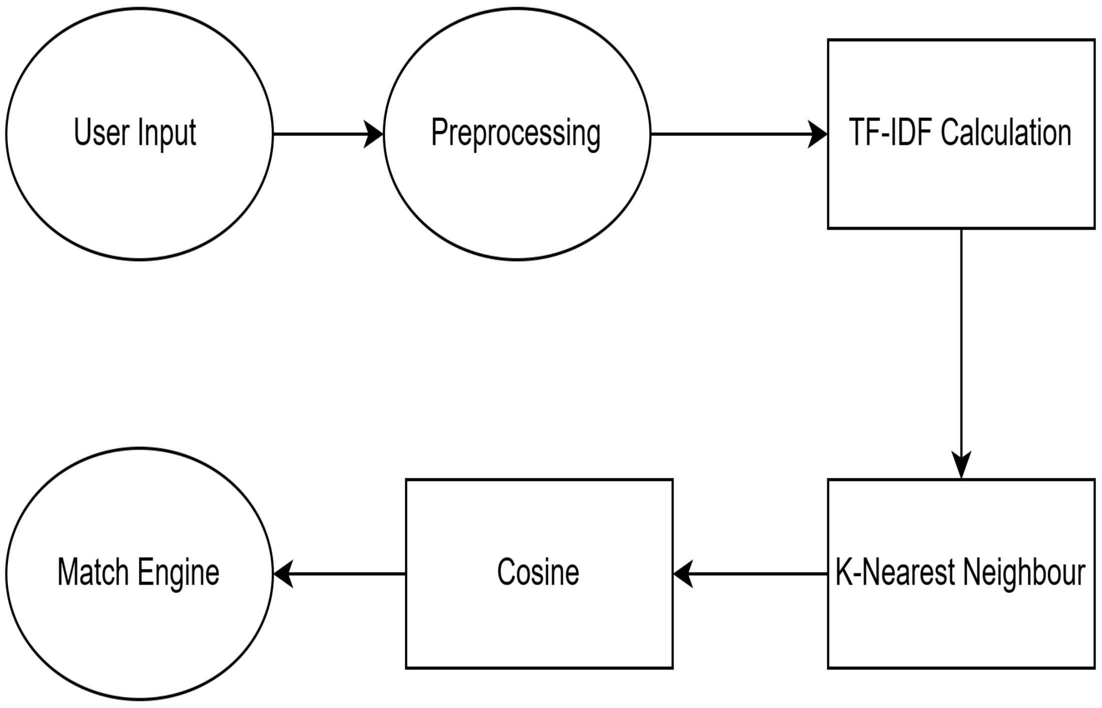


Figure 4.2: **Data flow diagram for Agriskill**

The figure 4.2 depicts a simplified text-based search engine system. It starts by taking user input, preprocessing it, and then calculating TF-IDF to assess word importance. The system then uses cosine similarity to compare the user input with documents in the collection and employs K-Nearest Neighbors to rank the most similar documents. This approach, commonly used in information retrieval, effectively retrieves relevant documents based on user queries.

4.3.2 Use Case Diagram

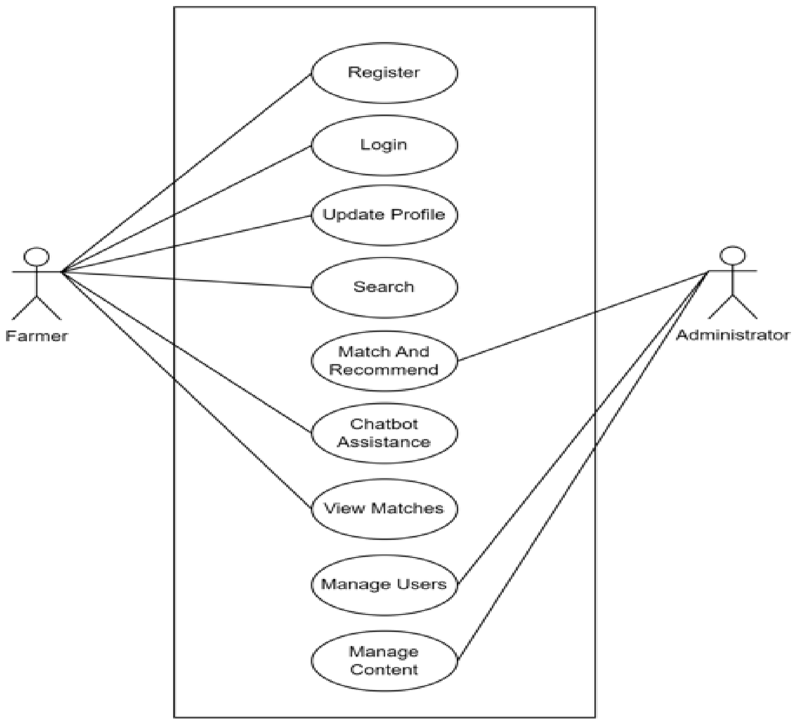


Figure 4.3: **Use case Diagram for Agriskill**

The figure 4.3 depicts the use case diagram for the AgriSkill web application outlines the various functionalities available to both farmers and administrators. Farmers can create new accounts, log in to their existing profiles, update their personal information, search for job seekers or other farmers based on their specific requirements and view matches recommended by the system. Additionally, administrators have the authority to manage user accounts, including creating, editing, and deleting them. They also have the responsibility of managing the content on the platform, which involves adding, editing, and deleting information. This comprehensive use case diagram provides a clear understanding of the key features and roles within the AgriSkill web application, ensuring a smooth and efficient user experience for both farmers and administrators alike.

4.3.3 Class Diagram

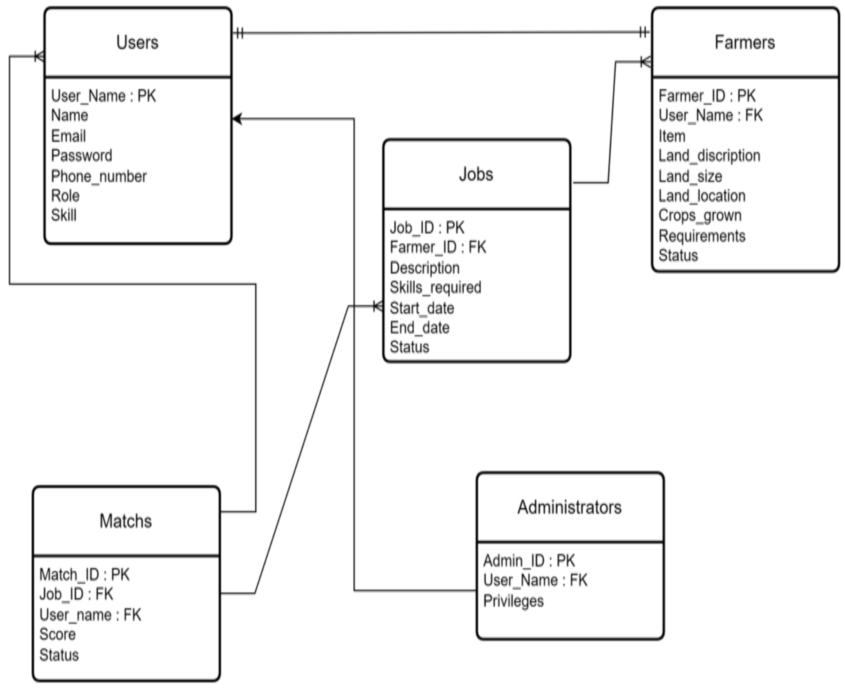


Figure 4.4: **Class diagram for Agriskill**

The figure 4.4 describes the class diagram for the AgriSkill web application outlines the data model, defining the entities, attributes, and relationships between them. Users, including farmers and administrators, are represented by the Users entity. Farmers have additional attributes related to their agricultural expertise, while Jobs represent job postings created by farmers. The Matches entity captures the matches between farmers and job seekers, and Administrators represent those with administrative privileges. The diagram also illustrates the relationships between these entities, such as one-to-many relationships between Users and Farmers, Farmers and Jobs, and Administrators and Users, as well as a many-to-many relationship between Users and Jobs. This class diagram provides a clear representation of the data structure and relationships within the AgriSkill web application, ensuring that the data is organized and accessible in a structured manner.

4.3.4 Sequence Diagram

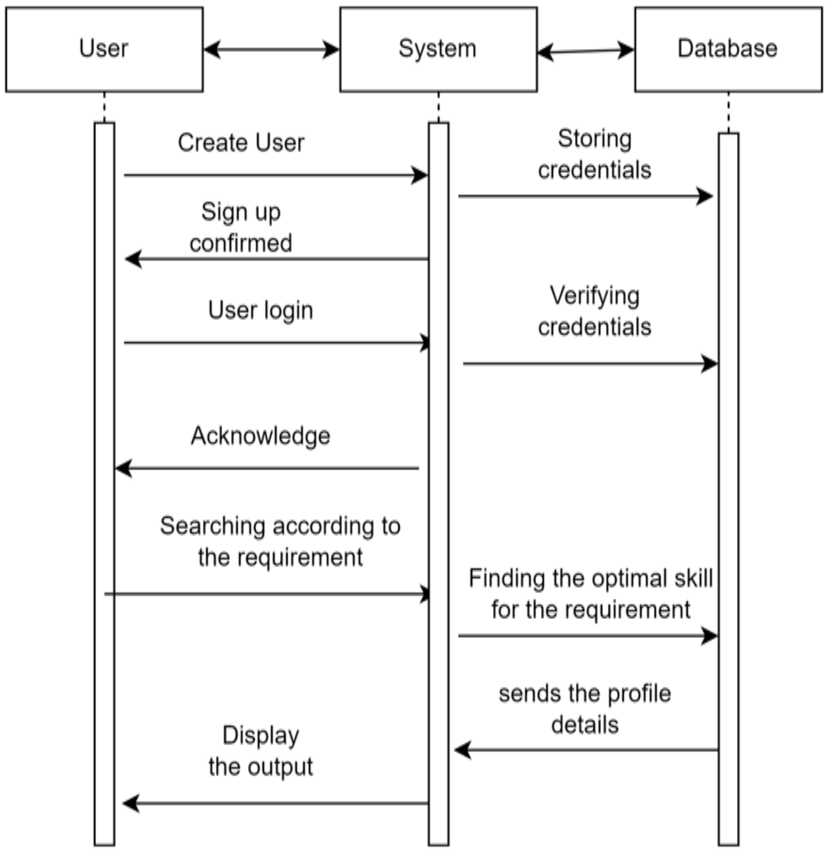


Figure 4.5: **Sequence diagram for Agriskill**

The figure 4.5 describes the sequence diagram that illustrates the interactions between the user, system, and database in a user registration and login process. The user initiates the process by creating a new account and signing up. The system then stores the user's credentials in the database. When the user logs in, the system verifies their credentials against the stored information in the database. If the credentials are valid, the system acknowledges the successful login. The user then proceeds to search for a specific requirement, and the system finds the optimal skill that matches the requirement. Finally, the system sends the profile details of the matched skill to the user

4.3.5 Collaboration diagram

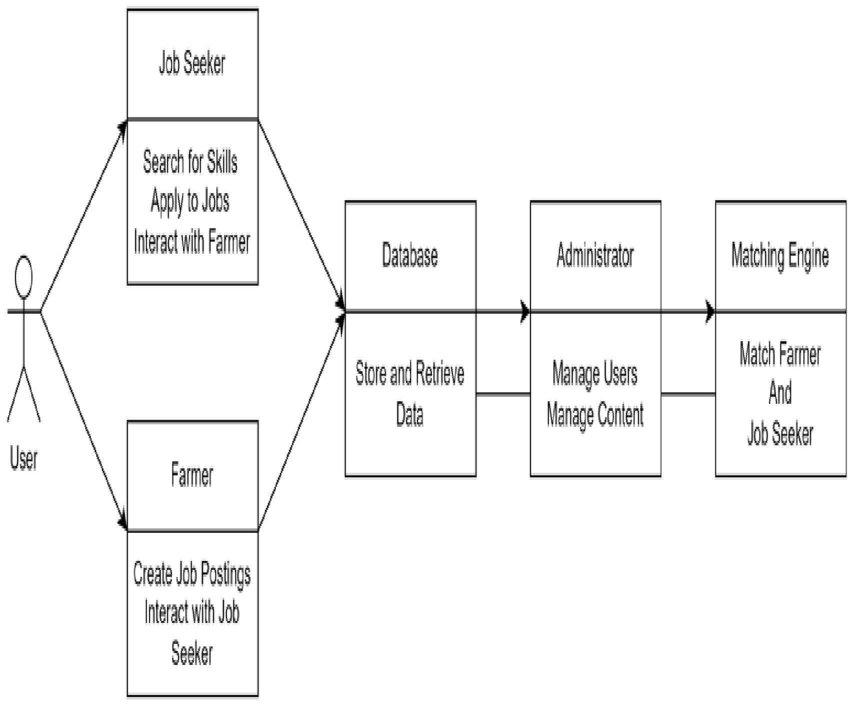


Figure 4.6: **Collaboration diagram for Agriskill**

The figure 4.6 it shows the provided collaboration diagram outlines the key roles and interactions within the AgriSkill web application. While it doesn't explicitly depict the connections between entities, the interactions implied in the diagram represent these relationships. For example, the interaction between the Farmer and Job Seeker suggests a connection through the platform. The Matching Engine connects Farmers and Job Seekers based on their profiles. The Database serves as the central connection point, storing and retrieving data for all entities. While the diagram doesn't use specific connectors or lines to visually represent these connections, the implied relationships between the entities are clear.

4.3.6 Activity Diagram

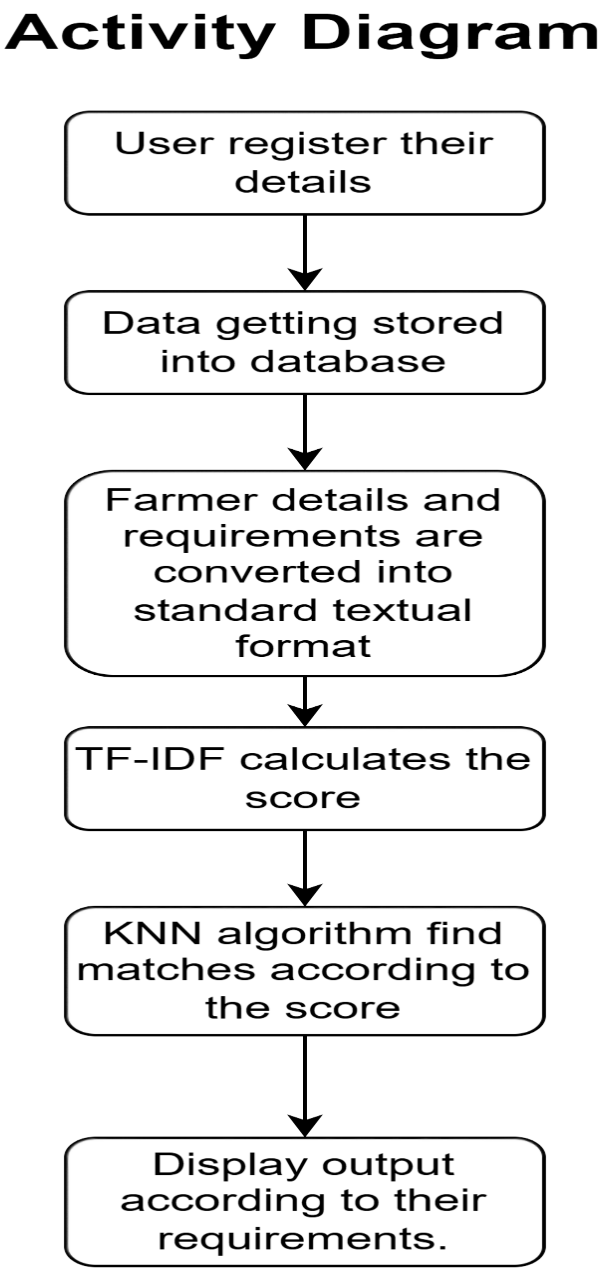

Activity Diagram
User register their details
Data getting stored into database
Farmer details and requirements are converted into standard textual format
TF-IDF calculates the score
KNN algorithm find matches according to the score
Display output according to their requirements.

Figure 4.7: **Activity diagram for Agriskill**

The figure 4.7 describes a system that matches the farmers with their requirements. User registration and storage of the data take place. The requirements are transformed into standard format. The TF-IDF score is calculated and KNN algorithm finds out the matches on the basis of the score. In the last step, it displays the relevant matches to the user.

4.4 Algorithm & Pseudo Code

4.4.1 Algorithm

Step 1: Start the program.

Step 2: Import required libraries (e.g., pandas, numpy, sklearn).

Step 3: Collect and organize information on users (farmers) and their queries.

Step 4: Gather expert profiles and their areas of expertise.

Step 5: Preprocess the text data (e.g., tokenize, remove stop words, and apply TF-IDF vectorization) to create a feature matrix for both user queries and expert expertise.

Step 6: Implement K-Nearest Neighbors (KNN) to find experts. Define a function that accepts a user query and determines the nearest experts based on their expertise features.

Step 7: Set the number of neighbors (k) for KNN and find the top-matching experts based on the nearest neighbor algorithm.

Step 8: Rank the matched experts from highest to lowest similarity score based on their distance to the query in the feature space.

Step 9: Return the list of recommended experts to the user.

Step 10: Evaluate the recommendations and test the system's performance (e.g., using metrics like accuracy, precision, and recall).

Step 11: Stop.

4.4.2 Pseudo Code

```
START Program

IMPORT required libraries (pandas, numpy, sklearn)

FUNCTION main()

    users_data = collect_users_data()


    experts_data = collect_experts_data()


    user_queries = preprocess_text(users_data.queries)
    expert_expertise = preprocess_text(experts_data.expertise)


    tfidf_vectorizer = initialize_TFIDF_vectorizer()
    user_features = tfidf_vectorizer.fit_transform(user_queries)
    expert_features = tfidf_vectorizer.transform(expert_expertise)


    knn_model = initialize_KNN_model(n_neighbors=k)


    knn_model.fit(expert_features)


    FOR each query IN user_features
        top_experts_indices = knn_model.kneighbors(query, return_distance=False)


        ranked_experts = rank_experts_by_distance(top_experts_indices)


        recommended_experts = get_experts_by_indices(ranked_experts)
        display_recommendations(recommended_experts)


    evaluate_recommendations()

END main
```

4.4.3 Data Set / Generation of Data

In the AgriSkill WebApp, the dataset plays a crucial role in facilitating the exchange of agricultural knowledge and expertise. The data is generated from multiple sources to ensure comprehensive coverage of agricultural practices, expert profiles, and user interactions.

The dataset consists of two primary components: user profiles and expert profiles. User profiles are created based on farmers' inputs, where they provide information about their farming practices, specific queries, and areas where they seek guidance. This data is collected through user registration forms and feedback submissions, capturing details such as farming types (crops, livestock), geographical location, and challenges faced.

On the other hand, expert profiles are constructed from a curated list of agricultural experts, advisors, and mentors. These profiles include their areas of expertise, qualifications, and contact information. The dataset also incorporates a knowledge base that includes articles, tutorials, and best practices related to various agricultural topics, which is continuously updated to reflect the latest advancements in the field.

Additionally, to enhance the recommendation engine, user interactions are logged, allowing the system to analyze patterns in user queries and expert responses. This dynamic dataset enables the AgriSkill WebApp to provide personalized recommendations and foster meaningful connections between farmers and agricultural experts, ultimately promoting knowledge sharing and improving agricultural productivity in rural communities.

4.5 Module Description

4.5.1 Module 1

ID	Name	District	City	Help Needed	Skill	Email	Availability
1	John Doe	Idukki	Thodupuzha	Land Preparation (Banana)	Banana Cultivation	john.doe@example.com	Full-time
2	Sarah Thomas	Kollam	Karunagappally	Planting (Coconut)	Coconut Farming	sarah.thomas@example.com	Part-time
3	Ravi Kumar	Thrissur	Chalakudy	Weeding (Paddy)	Paddy Weeding	ravi.kumar@example.com	Full-time
4	Meera Mohan	Alappuzha	Kuttanad	Irrigation (Rice)	Rice Irrigation	meera.mohan@example.com	Part-time
5	Ajay Menon	Ernakulam	Kochi	Harvesting (Rubber)	Rubber Harvesting	ajay.menon@example.com	Full-time
6	Priya Nair	Palakkad	Mannarkkad	Seed Sowing (Vegetables)	Vegetable Sowing	priya.nair@example.com	Part-time
7	Ramesh Pillai	Malappuram	Manjeri	Tree Planting (Rubber)	Rubber Planting	ramesh.pillai@example.com	Full-time
8	Anita Kurup	Kannur	Thalassery	Land Preparation (Paddy)	Paddy Cultivation	anita.kurup@example.com	Part-time
9	Suresh Babu	Kozhikode	Vadakara	Ploughing (Banana)	Banana Farming	suresh.babu@example.com	Full-time
10	Latha Krishnan	Pathanamthitta	Ranni	Fertilizer Application (Coconut)	Coconut Fertilizing	latha.krishnan@example.com	Part-time
11	Binu Varghese	Kottayam	Changanassery	Pest Control (Paddy)	Paddy Pest Control	binu.varghese@example.com	Full-time
12	Divya Raj	Kasaragod	Nileshwaram	Watering (Vegetables)	Vegetable Watering	divya.raj@example.com	Part-time
13	Arun Das	Wayanad	Kalpetta	Mulching (Banana)	Banana Mulching	arun.das@example.com	Full-time
14	Kavya Shaji	Thiruvananthapuram	Neyyattinkara	Harvesting (Paddy)	Paddy Harvesting	kavya.shaji@example.com	Part-time
15	Nikhil P	Alappuzha	Haripad	Soil Preparation (Rubber)	Rubber Soil Prep	nikhil.p@example.com	Full-time

Table 4.1: Collection of Data

Collecting the datasets which contains of the worker id ,name ,age, gender, location, skills, experience, wage, preferred, languages, contact information, notes

4.5.2 Module 2

```
import os
import pandas as pd
data = pd.read_csv("skilshare.csv")
print(data.head())
print(data.isnull().sum())
print(data.describe())
data.fillna(0, inplace=True)
print(data.head())
```

Check whether the data contains null values or not and describing the data. Preprocessing is the process of cleaning and processing the data. Along with that it performs vectorizarion of the text present for match-finding.

4.5.3 Module 3

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors
df = pd.read_csv('SkillShare.csv')
df['combined_features'] = df['Skill'] + " " + df['Help Needed']
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf_vectorizer.fit_transform(df['combined_features'])
k = 5
knn = NearestNeighbors(n_neighbors=k, metric='cosine')
knn.fit(tfidf_matrix)
def find_matches(query, num_matches=5):
    query_tfidf = tfidf_vectorizer.transform([query])
    distances, indices = knn.kneighbors(query_tfidf, n_neighbors=num_matches)
    matches = df.iloc[indices[0]].copy()
    matches['Similarity'] = 1 - distances[0]
    return matches
query = "Land Preparation Banana"
matches = find_matches(query, num_matches=5)
print("\nMatched skilled professionals:")
print(matches[['ID', 'Name', 'Skill', 'Help Needed', 'Similarity']])
```

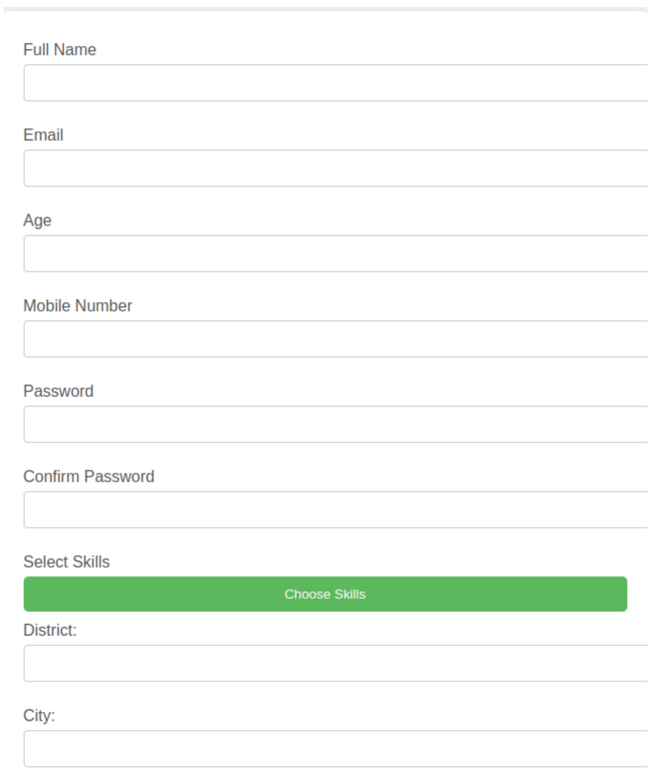
It is a section of the program were TF-IDF and k-NN is implemented.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design


Full Name

Email

Age

Mobile Number

Password

Confirm Password

Select Skills

Choose Skills

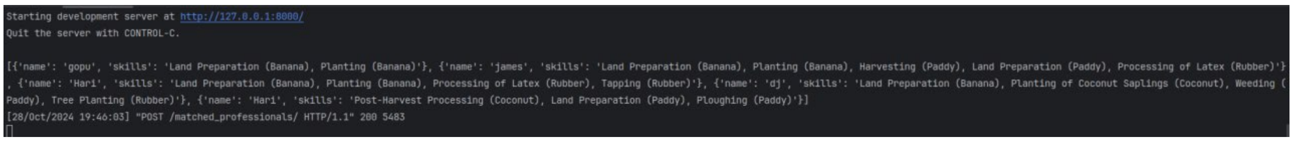
District:

City:

Figure 5.1: **Input design**

The figure 5.1 depicts the input design of the Agriskill webapp.It takes in the user details which is then stored in database and used to match the users according to it. The matching is done through filtering out the users based on skill and district.

5.1.2 Output Design



```
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

[{'name': 'gopu', 'skills': 'Land Preparation (Banana), Planting (Banana)'}, {'name': 'james', 'skills': 'Land Preparation (Banana), Planting (Banana), Harvesting (Paddy), Land Preparation (Paddy), Processing of Latex (Rubber)'}
, {'name': 'Hari', 'skills': 'Land Preparation (Banana), Planting (Banana), Processing of Latex (Rubber), Tapping (Rubber)'}, {'name': 'dj', 'skills': 'Land Preparation (Banana), Planting of Coconut Saplings (Coconut), Weeding (
Paddy), Tree Planting (Rubber)'}, {'name': 'Hari', 'skills': 'Post-Harvest Processing (Coconut), Land Preparation (Paddy), Ploughing (Paddy)'}]
[28/Oct/2024 19:46:03] "POST /matched_professionals/ HTTP/1.1" 200 5483
```

Figure 5.2: **Output design**

The figure 5.2 depicts the output design of the Agriskill webapp.It shows the sample output got while executing it using Term-Frequency Inverse Document Frequency and K-Nearest Neighbour.

5.2 Testing

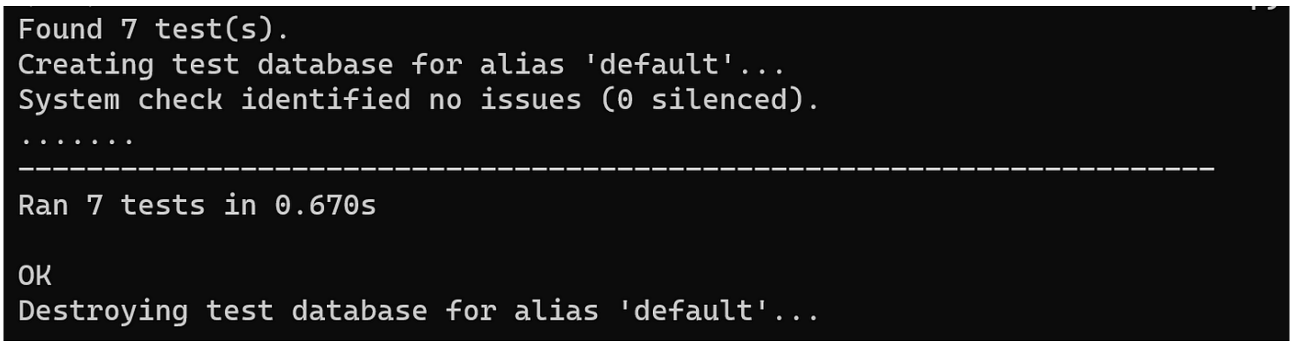
5.3 Types of Testing

5.3.1 Unit testing

Input

```python
    import unittest
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors

class TestLandownerMatching(unittest.TestCase):
    def test_parse_skills(self):
        skills = "farming, irrigation, harvesting"
        parsed_skills = [skill.strip() for skill in skills.split(',') if skill.strip()]
        expected = ["farming", "irrigation", "harvesting"]
        self.assertEqual(parsed_skills, expected)

    def test_tfidf_vectorization(self):
        skills = ["farming, irrigation", "irrigation, planting"]
        vectorizer = TfidfVectorizer()
        tfidf_matrix = vectorizer.fit_transform(skills)
        self.assertEqual(tfidf_matrix.shape[0], 2)  # 2 documents
        self.assertGreater(tfidf_matrix.shape[1], 0)  # Number of features based on unique words

    def test_knn_matching(self):
        skills = ["farming, irrigation", "irrigation, planting"]
        vectorizer = TfidfVectorizer()
        tfidf_matrix = vectorizer.fit_transform(skills)

        knn = NearestNeighbors(n_neighbors=1, metric='cosine')
        knn.fit(tfidf_matrix)

        distances, indices = knn.kneighbors(tfidf_matrix[0])
        self.assertEqual(len(indices[0]), 1)  # Expect 1 closest neighbor due to n_neighbors=1

if __name__ == "__main__":
    unittest.main()
```

Test result



```
Found 7 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.......
----------------------------------------------------------------------
Ran 7 tests in 0.670s

OK
Destroying test database for alias 'default'...
```

Figure 5.3: **Output unit testing**

The figure 5.3 depicts the output after executing the unit testing.It tests it with a series of alias and run tests according to those.

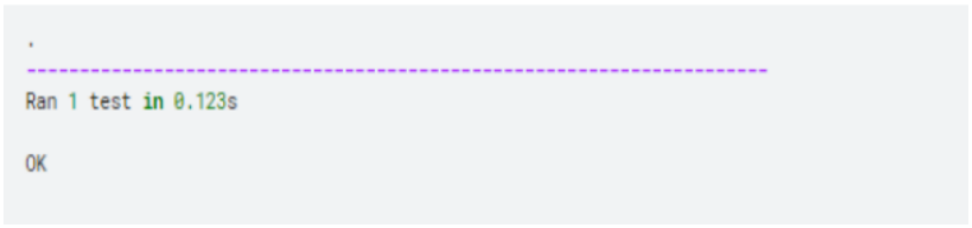
Test result

5.3.2 Integration testing

Input

```
import unittest
from unittest.mock import MagicMock

class TestIntegration(unittest.TestCase):
    def test_landowner_matching(self):
        request = MagicMock()
        request.method = 'POST'
        request.POST = {'total_area': '100', 'skills': 'irrigation, harvesting'}

        # Mock database objects and method calls
        SkilledProfessional.objects.all = MagicMock(return_value=[
            MagicMock(skills="irrigation, planting"),
            MagicMock(skills="harvesting, plowing"),
        ])

        # Call the function and check the output
        response = landowner_home(request)
        self.assertEqual(response.status_code, 302)  # Redirect to 'matched_professionals'
        self.assertIn('matched_professionals', request.session)  # Check if data is stored in
            session

    def test_professional_matching(self):
        request = MagicMock()
        request.method = 'POST'
        request.session = {'user_email': 'test@example.com'}

        # Mock skilled professional and landowner database objects
        SkilledProfessional.objects.get = MagicMock(return_value=MagicMock(skills="irrigation,
            planting"))
        Landowner.objects.filter = MagicMock(return_value=[
            MagicMock(full_name="Landowner A", help_needed="irrigation, harvesting"),
            MagicMock(full_name="Landowner B", help_needed="planting"),
        ])

        # Call the function and check the output
        response = skilled_professional_home(request)
        self.assertEqual(response.status_code, 302)  # Redirect to 'matched_landowners'
        self.assertIn('matched_landowners', request.session)  # Check if data is stored in
            session

if __name__ == "__main__":
    unittest.main()
```

Test result



```
.
----------------------------------------------------------------------
Ran 1 test in 0.123s

OK
```

Figure 5.4: **Output integration testing**

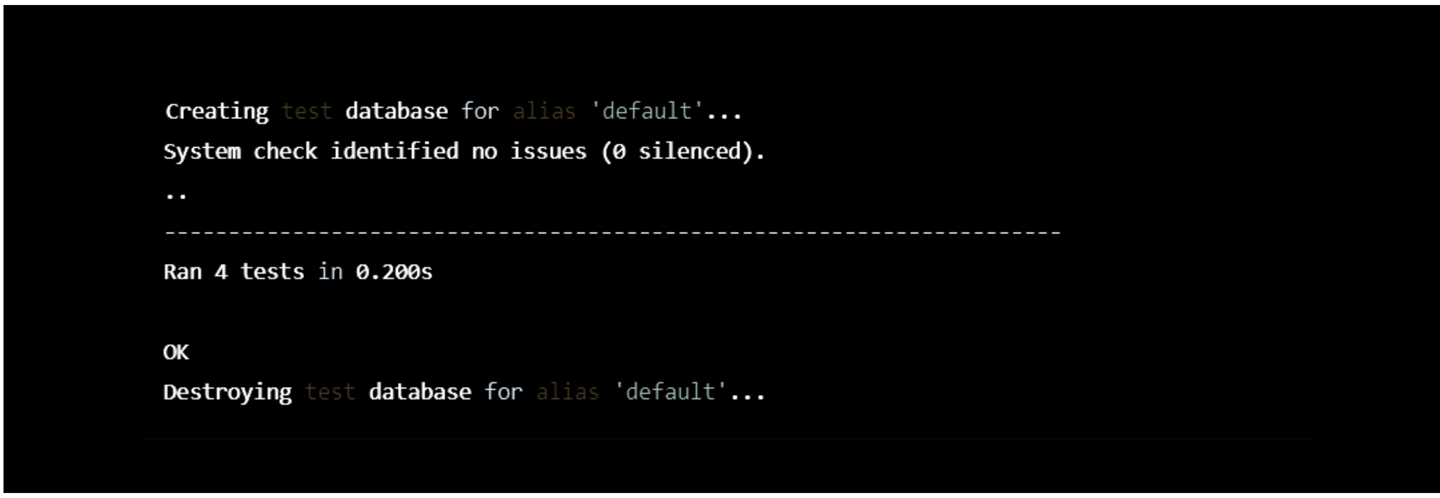
This figure 5.4 depicts test result from integration testing done in the webapp.

5.3.3 System testing

Input

```
from django.test import TestCase, Client
from django.urls import reverse
from myapp.models import SkilledProfessional, Landowner

class LandownerMatchingSystemTest(TestCase):
    def setUp(self):
        self.client = Client()
        SkilledProfessional.objects.create(full_name="Pro 1", skills="irrigation, harvesting")
        SkilledProfessional.objects.create(full_name="Pro 2", skills="planting, plowing")
    def test_landowner_matching_flow(self):
        # Submit form data as a landowner
        response = self.client.post(reverse('landowner_home'), {
            'total_area': '100',
            'skills': 'irrigation, planting'
        })class SkilledProfessionalMatchingSystemTest(TestCase):
    def setUp(self):
        self.client = Client()
        # Creating test skilled professional
        self.professional = SkilledProfessional.objects.create(
            full_name="Pro 1", email="test@example.com", skills="irrigation, harvesting"
        )
        Landowner.objects.create(full_name="Landowner A", help_needed="irrigation")
        Landowner.objects.create(full_name="Landowner B", help_needed="planting")
    def test_professional_matching_flow(self):
        # Log in as a skilled professional
        session = self.client.session
        session['user_email'] = 'test@example.com'
        session.save()
        response = self.client.post(reverse('skilled_professional_home'))
        self.assertRedirects(response, reverse('matched_landowners'))
        session = self.client.session
        self.assertIn('matched_landowners', session)
        matched_landowners = session['matched_landowners']
        expected_landowners = [{'name': "Landowner A", 'help_needed': "irrigation", 'district':
             "", 'city': ""}]
        self.assertEqual(matched_landowners, expected_landowners)
        self.assertRedirects(response, reverse('matched_professionals'))
        session = self.client.session
        self.assertIn('matched_professionals', session)
        matched_professionals = session['matched_professionals']
        expected_professionals = [{'name': "Pro 1", 'skills': "irrigation, harvesting"}]
        self.assertEqual(matched_professionals, expected_professionals)
class SkilledProfessionalMatchingSystemTest(TestCase):
    def setUp(self):
        self.client = Client()
        self.professional = SkilledProfessional.objects.create(
            full_name="Pro 1", email="test@example.com", skills="irrigation, harvesting"
        )
        Landowner.objects.create(full_name="Landowner A", help_needed="irrigation")
        Landowner.objects.create(full_name="Landowner B", help_needed="planting")
    def test_professional_matching_flow(self):
        # Log in as a skilled professional
        session = self.client.session
        session['user_email'] = 'test@example.com'
        session.save()
        response = self.client.post(reverse('skilled_professional_home'))
        self.assertRedirects(response, reverse('matched_landowners'))
        session = self.client.session
        self.assertIn('matched_landowners', session)
        matched_landowners = session['matched_landowners']
        expected_landowners = [{'name': "Landowner A", 'help_needed': "irrigation", 'district':
            "", 'city': ""}]
        self.assertEqual(matched_landowners, expected_landowners)
```

Test Result



```
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
----------------------------------------------------------------------
Ran 4 tests in 0.200s

OK
Destroying test database for alias 'default'...
```

Figure 5.5: **System Test Image**

The figure 5.5 depicts results got from executing the system testing on the webapp.

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The AgriSkill web application is designed to be an efficient, intuitive place for easy interactions among the farmers, agricultural experts, and all stakeholders. With the implementation of TF-IDF combined with k-NN for match-finding, AgriSkill comes with the ability to execute highly accurate search and recommendations. This strong algorithmic mix allows the platform to make judgments regarding the appropriateness of keywords and user-entered information and to return results that are faster and closer to the requirements of users. This precision in searching saves time users spend sifting through information, hence boosting productivity as it gives them quicker access to crucial agricultural insights.Apart from search functionality, expert matching is just calibrated to ensure that a person is connected with experts who share relevance in topic and are neighbors within proximity. This location-aware, context-aware matching ensures quality links to enable timely access to locality-specific advice on agricultural practices, including crop management, pest control, and soil care. Locality-specific recommendations enable a system to respond quicker and promptly resolve problems.

The central idea of the architecture in AgriSkill is scalability and performance. The application uses efficient techniques for indexing data, hence ensuring robust performance even while scaling the amount of data. Also, using a lightweight stack of technology such as HTML, CSS, JavaScript at the front end and Python with Django on the back-end ensures it can handle its server load without latency. This design is also meant to ensure effective running even with minimal computational powers; thereby, the solution should make it accessible in the remote areas where there are infrastructure limitations. System architecture overall is flexible in accommodating additional enhancements in recommendation algorithms and real-time communications with advanced data analytics. Therefore, AgriSkill would still meet current users' demands but can accommodate their growing needs in its operational efficacy and functionality.

6.2 Comparison of Existing and Proposed System

Existing system

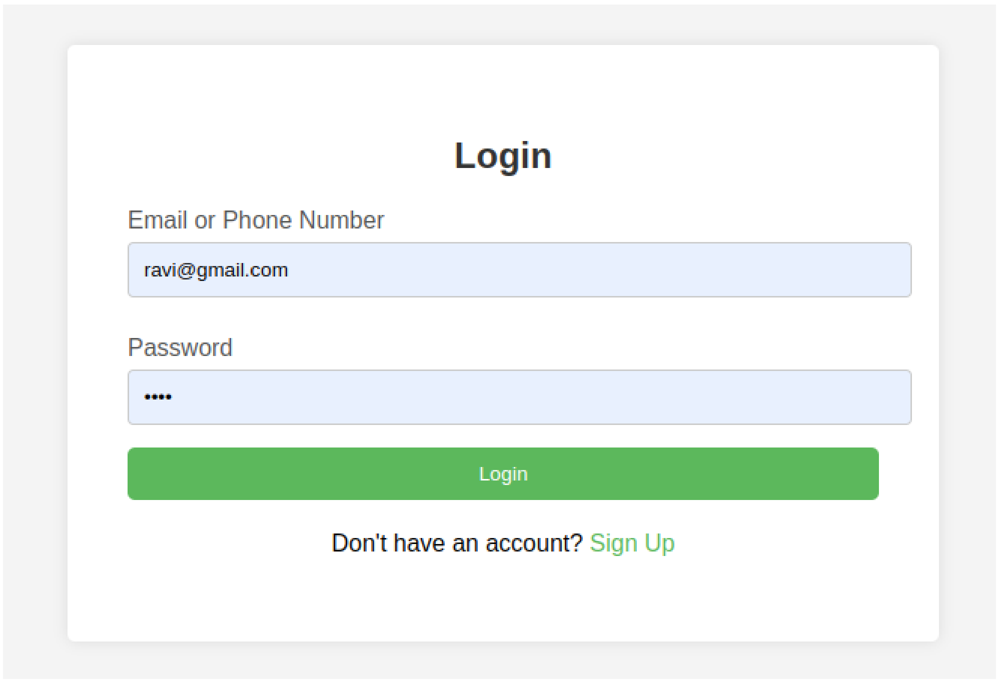
The proposed AgriSkill system revolutionizes the knowledge of agriculture by combining modern technology with advanced algorithms in designing an interactive and responsive web-based platform. AgriSkill differs from the old systems because it applies the TF-IDF with k-nearest neighbors (k-NN) to implement accurate, relevance-based searching and expert recommendation for information retrieval by farmers and other agricultural experts, quickly to the problems in hand.With AgriSkill's dynamic search and recommendation functionalities, the advice is personalized because recommendations are tailored according to crops, soil types, or even localized agricultural challenges. Such is quite the opposite of generalized advice seen in many existing systems. Besides, the platform includes an expert matching system based on considerations of the professional's expertise and the proximity geographically, thus simplifying how farmers can connect with appropriate professionals who could provide relevant support in the context.

Scalability is one strength of AgriSkill since it can handle vast amounts of agricultural data without performance degradation, thanks to its efficient data indexing and lightweight architecture; all in HTML, CSS, JavaScript frontend; and Python and Django backend. Its accessible design also ensures that AgriSkill will work reliably even in areas with the lowest level of infrastructure.Finally, AgriSkill encourages greater user interaction by its friendly interface and community-based features such as discussion forums and real-time messaging. The engagement-based approach ensures that the users are always active on the platform, thus facilitating an ongoing knowledge exchange that improves agricultural productivity at the individual and community levels.

Proposed system(AgriSkill Webapp)

AgriSkill webapp takes all the disadvantage of previously agricultural information system by taking up the strong algorithms for retrieving TF-IDF, and k -nearest neighbour to get excellent personalized searches relevant to query. Thus Agri Skill is entirely different from many other such sites, using expert-matching technology using user questions and professional years of work along with any geographical place which helps user connect on a timely level with actual farming experts with more effects. The platform's architecture is robust enough to support scalability and efficient data handling, thus ensuring that performance is maintained with a growing user base and dataset sizes. AgriSkill also enables real-time collaboration and knowledge sharing across various regions, creating a supporting community for farmers as well as experts. This allows AgriSkill to be an accessible and powerful tool in addressing agricultural challenges, enhancing productivity, and transferring knowledge across the agricultural sector.

Output

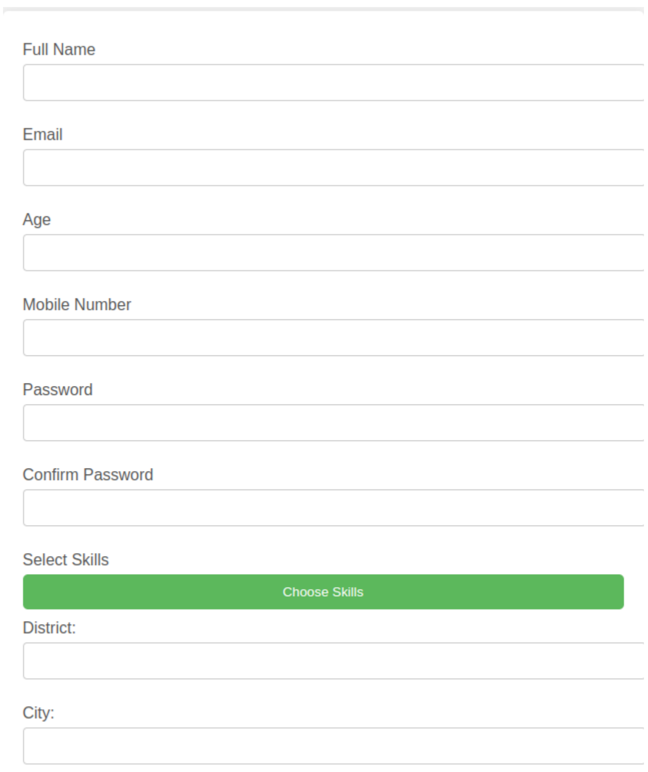


ravi@gmail.com

••••

Figure 6.1: **Login page**

The figure 6.1 is an image shows a simple screen of the Agriskill. The user is asked to enter their email or phone number and password to log in. There is also a link to sign up for a new account if the user does not have one.


Full Name

Email

Age

Mobile Number

Password

Confirm Password

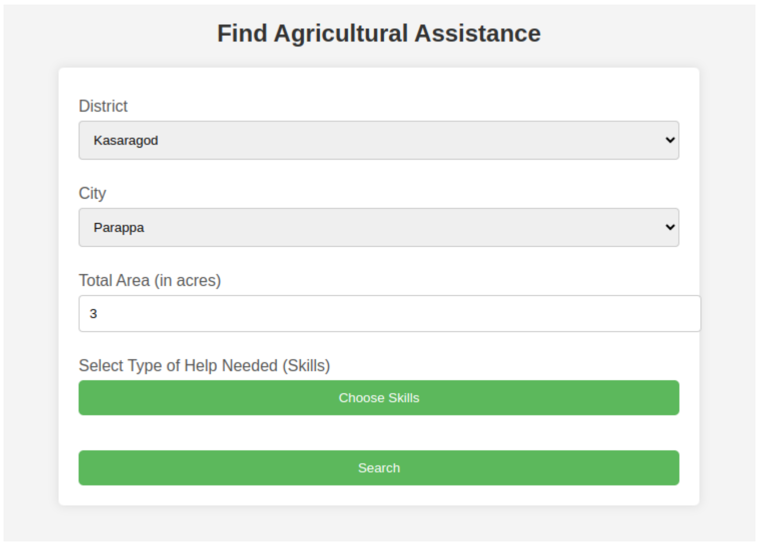
Select Skills

District:

City:

Figure 6.2: **Signup page**

The figure 6.2 is an image shows a registration form for skilled professionals. The form requires users to enter their full name, email address, age, mobile number, password, confirm password, select their skills, district, and city.

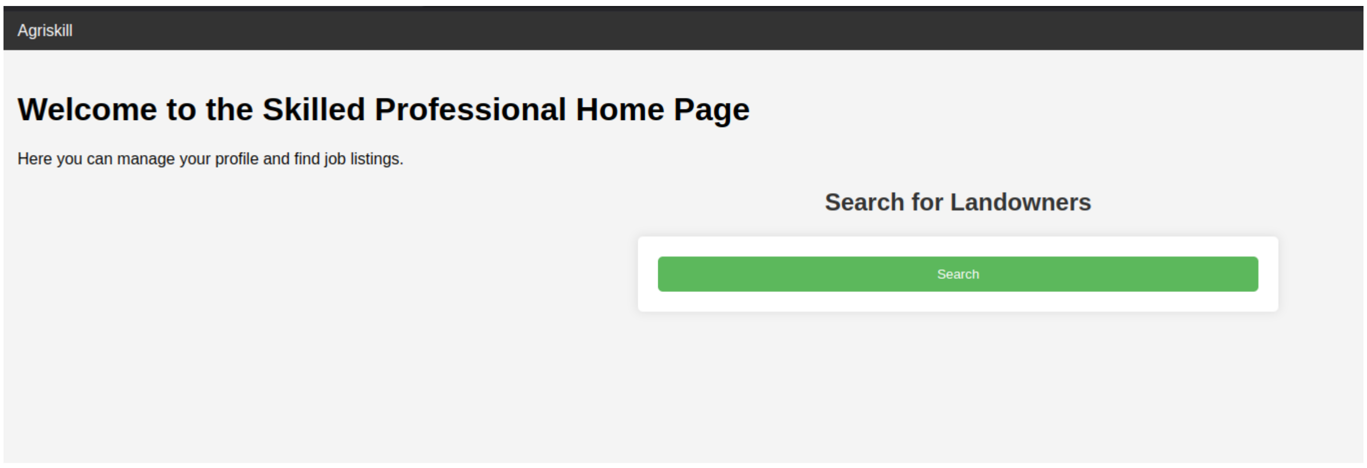

Find Agricultural Assistance

District	Kasaragod
City	Parappa
Total Area (in acres)	3

Select Type of Help Needed (Skills)

Figure 6.3: **Landowner home page**

The figure 6.3 is an image shows a form to find agricultural assistance. The form requires the user to enter the district, city, total area of land, and select the type of help needed (skills). Once all information is filled in, the user can click the ”Search” button to find assistance.


Agriskill

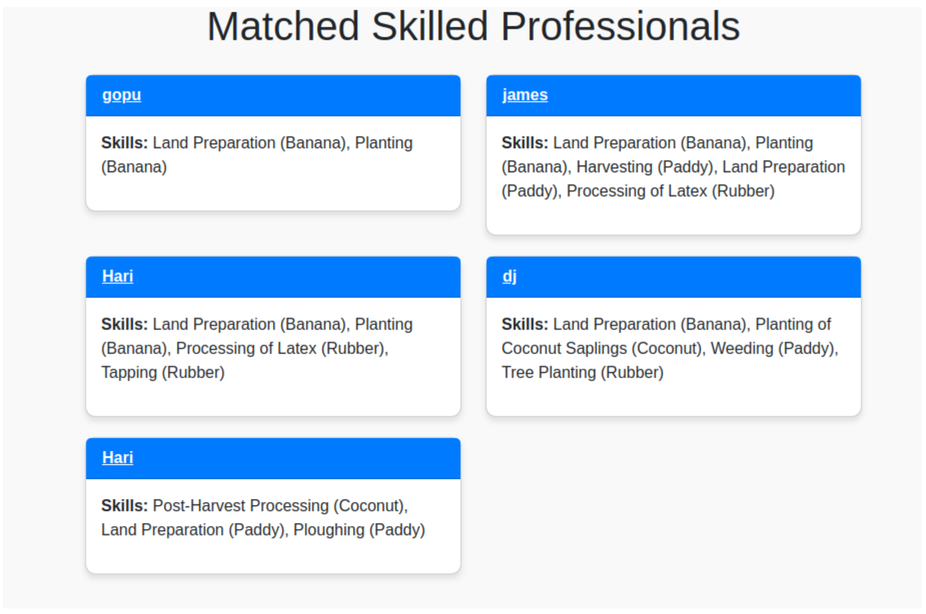
Welcome to the Skilled Professional Home Page

Here you can manage your profile and find job listings.

Search for Landowners

Figure 6.4: **Skilled professional home page**

The figure 6.4 is the homepage for skilled professionals on the Agriskill platform. It provides a welcome message, a brief description of its functionality, and a search bar to find landowners.


Matched Skilled Professionals

gopu

Skills: Land Preparation (Banana), Planting (Banana)

james

Skills: Land Preparation (Banana), Planting (Banana), Harvesting (Paddy), Land Preparation (Paddy), Processing of Latex (Rubber)

Hari

Skills: Land Preparation (Banana), Planting (Banana), Processing of Latex (Rubber), Tapping (Rubber)

dj

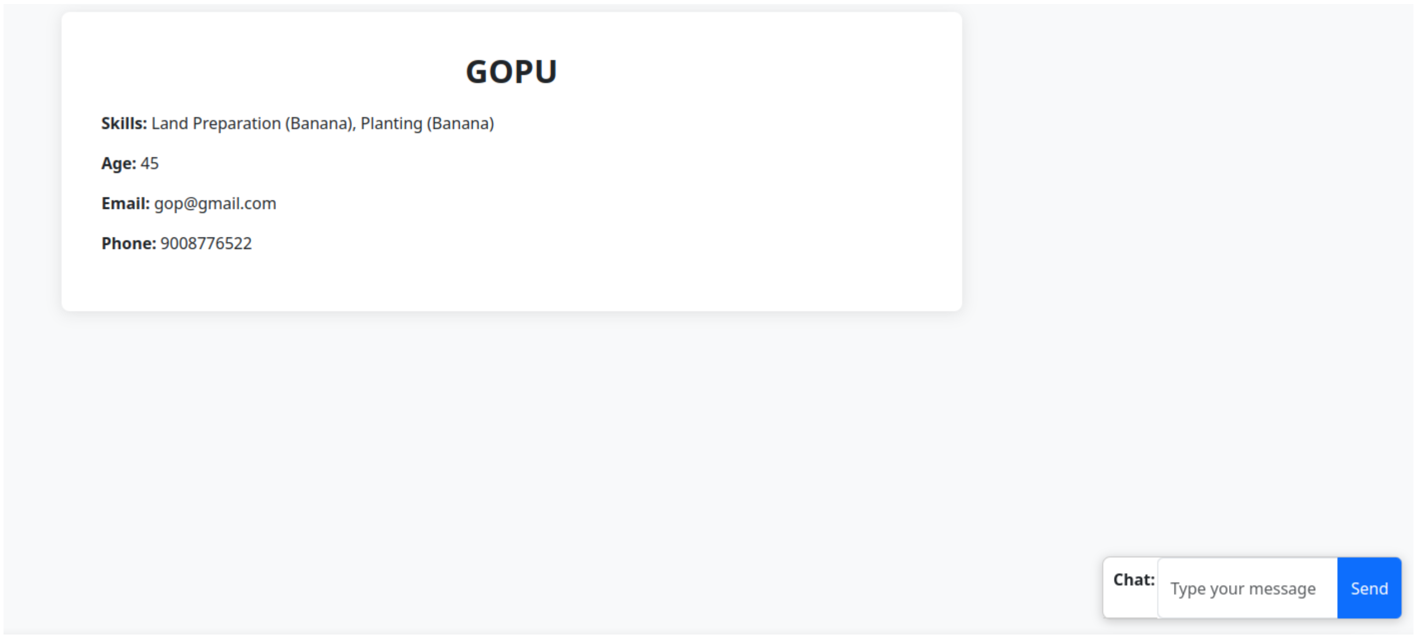
Skills: Land Preparation (Banana), Planting of Coconut Saplings (Coconut), Weeding (Paddy), Tree Planting (Rubber)

Hari

Skills: Post-Harvest Processing (Coconut), Land Preparation (Paddy), Ploughing (Paddy)

Figure 6.5: **List of skilled professionals**

The figure 6.5 is an image shows a list of matched skilled professionals for a specific agricultural task. Displaying each professionals name and skills.



GOPU

Skills: Land Preparation (Banana), Planting (Banana)

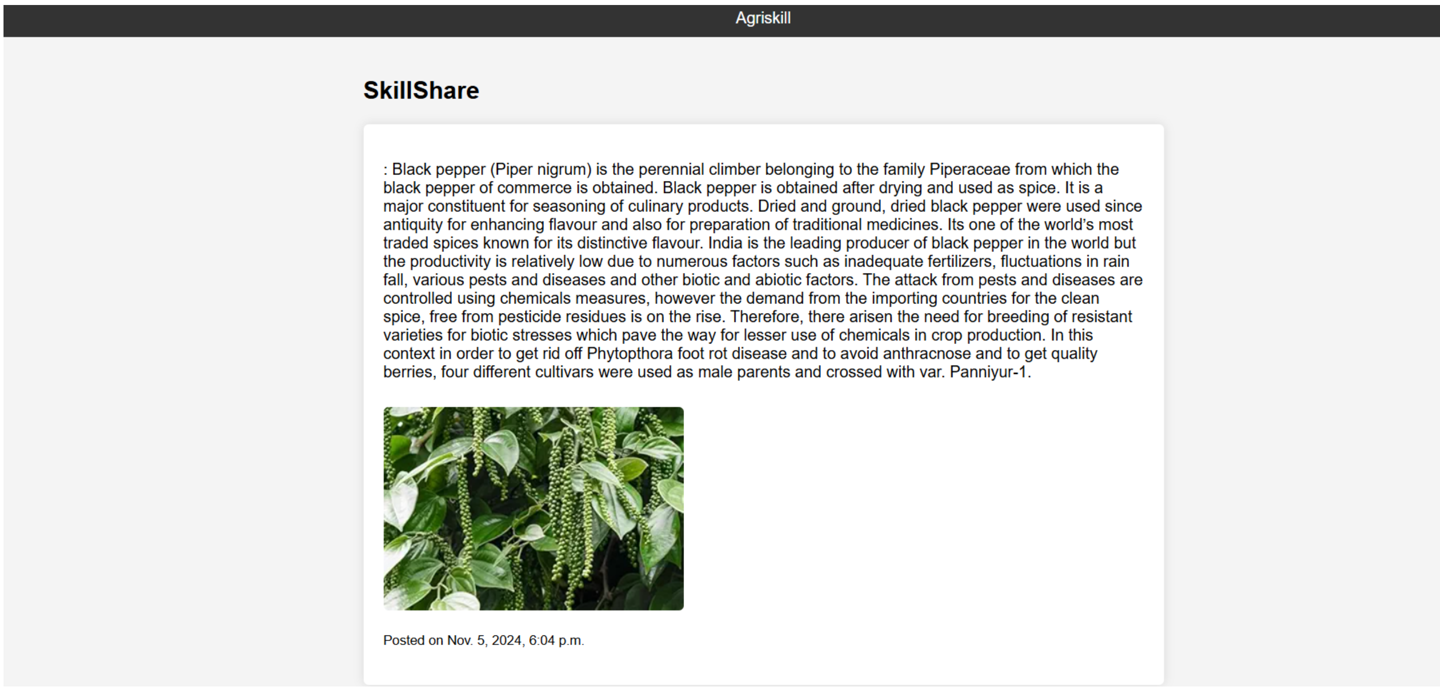
Age: 45

Email: gop@gmail.com

Phone: 9008776522

Figure 6.6: **Selected user profile**

The figure 6.6 is an image shows a profile of a skilled professional. The profile displays his skills, age, email address, and phone number.



SkillShare

: Black pepper (Piper nigrum) is the perennial climber belonging to the family Piperaceae from which the black pepper of commerce is obtained. Black pepper is obtained after drying and used as spice. It is a major constituent for seasoning of culinary products. Dried and ground, dried black pepper were used since antiquity for enhancing flavour and also for preparation of traditional medicines. Its one of the world's most traded spices known for its distinctive flavour. India is the leading producer of black pepper in the world but the productivity is relatively low due to numerous factors such as inadequate fertilizers, fluctuations in rain fall, various pests and diseases and other biotic and abiotic factors. The attack from pests and diseases are controlled using chemicals measures, however the demand from the importing countries for the clean spice, free from pesticide residues is on the rise. Therefore, there arisen the need for breeding of resistant varieties for biotic stresses which pave the way for lesser use of chemicals in crop production. In this context in order to get rid off Phytopthora foot rot disease and to avoid anthracnose and to get quality berries, four different cultivars were used as male parents and crossed with var. Panniyur-1.

Posted on Nov. 5, 2024, 6:04 p.m.

Figure 6.7: **Discussion form**

The figure 6.6 is an image shows the discussion form to share their advanced insights about agriculture.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The AgriSkill web application is set to revolutionize the sharing of agricultural knowledge in rural farming communities. The site offers an interactive platform and connects farmers with agricultural experts, advisors, and peers where they access tailored information and practical insights. Advanced algorithms such as TF-IDF and k-nearest neighbors (k-NN) are applied in matching and ensure users are matched up with the content and other individuals with specific needs. This custom method assists the farmers to cope with various problems they are facing by crop yield, resource efficiency, and sustainable use of resources.

This includes direct solving of shortages in skilled labor for rural agriculture through fostering a collaborative ecosystem. It promotes sharing the knowledge among the members in the community, valuable exchange, and learning among the various individuals within the group so that there is an actual bridging of the gaps in knowledge, thus maximizing agricultural efficiency. By practicing sustainable farming techniques like permaculture and water conservation, AgriSkill provides farmers with an opportunity to embrace sustainable operations, therefore contributing to long-term sustainability.

Besides, the AgriSkill web application gives great social and economic advantages to rural areas by connecting farmers with mentors and peers and industry experts thereby reducing feelings of loneliness and isolation and creates an atmosphere of community. Stimulating local entrepreneurship and creation of jobs, it directly adds to the broader economical development of rural regions. Going beyond an agricultural tool, AgriSkill is a building solution into a future resilient in farming communities that are sustainable.

7.2 Future Enhancements

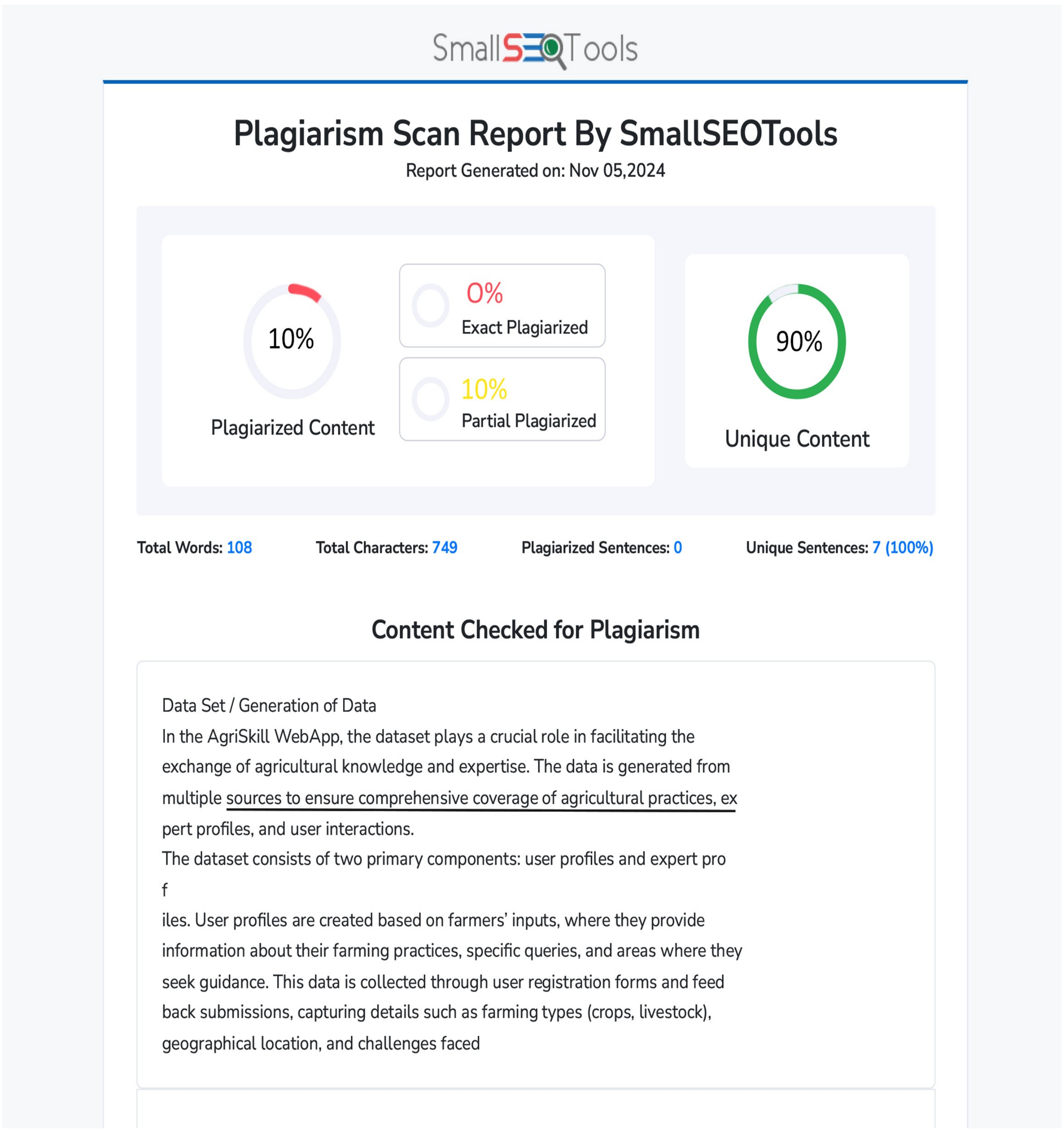
The AgriSkill web application has massive scope for further development that would significantly enhance the impact upon rural agricultural communities. For sure, one of the areas of significant development involves real-time data analytics and predictive modeling. AgriSkill could incorporate Internet of Things (IoT) devices and sensors into agriculture and provide timely information regarding soil moisture levels, weather conditions, and overall crop health. This data-driven approach would equip farmers with decisions, optimization of resources used, and waste reduction. Furthermore, the application of machine learning algorithms would be able to allow the system to present personalized recommendations in terms of historical data that users could have for particular farming environments, thus increasing relevance and effectiveness of shared knowledge.

Along with these technological advancements, the social feature extension on the platform must take place so that a robust feeling of community can be achieved within its users. For example, discussion forums, live QA sessions, and virtual workshops will encourage higher degrees of interaction among the farmer and agricultural experts' level of engagement. These shared values will help provide an excellent user experience since farmers share knowledge, support, and the best practices. There are social networking capabilities, which encompass user profiles and connectivity capabilities to find mentors and work collaboratively with like-minded individuals, significantly enhancing the value of the platform.

In conclusion, gamification will be a good motivator for AgriSkill. Features such as badges, leaderboards, and rewards for active contributors motivate the users to engage deeper within the platform and share expertise. These enhancements not only make learning fun but also contribute to a thriving knowledge-sharing ecosystem that benefits the whole agricultural community. Through continuous evolution and expansion in features, AgriSkill can be instrumental in the advancement of sustainable agricultural practices, spurring economic growth in rural areas, and enabling farmers with the necessary skills to excel in this changing landscape.

Chapter 8

PLAGIARISM REPORT



SmallSEOTools

Plagiarism Scan Report By SmallSEOTools

Report Generated on: Nov 05,2024

10% Plagiarized Content

0% Exact Plagiarized

10% Partial Plagiarized

90% Unique Content

Total Words: 108 Total Characters: 749 Plagiarized Sentences: 0 Unique Sentences: 7 (100%)

Content Checked for Plagiarism

Data Set / Generation of Data
In the AgriSkill WebApp, the dataset plays a crucial role in facilitating the exchange of agricultural knowledge and expertise. The data is generated from multiple sources to ensure comprehensive coverage of agricultural practices, ex
pert profiles, and user interactions.
The dataset consists of two primary components: user profiles and expert pro
f
iles. User profiles are created based on farmers' inputs, where they provide information about their farming practices, specific queries, and areas where they seek guidance. This data is collected through user registration forms and feed
back submissions, capturing details such as farming types (crops, livestock), geographical location, and challenges faced

Figure 8.1: **Plagiarism report**

Appendices

Appendix A

Complete Data / Sample Data / Sample Source Code / etc

A.1 Complete Data

A.1.1 Dataset Overview

This project relies on a rich dataset of agricultural skills and professional expertise profiles to train and evaluate an intelligent recommendation system in AgriSkill for efficiently matching landowners with skilled professionals based on location and specific agricultural needs. It draws its data from numerous agricultural databases and available profiles, giving rich detail into professional skills, areas of expertise, and geolocations. It contains 5,000 entries covering a wide range of agricultural specialties, which means AgriSkill can provide the user with highly relevant expertise.

A.1.2 Agricultural Skills Categories

The dataset comprises the following categories of skills that are crucial to training the recommendation engine so that it can match landowners with the right skilled professionals:

- Paddy Cultivation: This skill includes expertise in all the different stages such as preparation of land, planting, irrigation, and harvesting.
- Coconut Cultivation: They have cultivation skills such as saplings planting, fertilization, pest control, and the maintenance of trees.
- Tapping of Rubber: They have skills in tapping latex, processing, and taking care of plantations.
- Vegetable Cultivation: They know how to grow common vegetables and how the soil is prepared and on rotation practices.

- Orchard Management: They have cultivation skills in fruit trees by pruning, disease control, and harvesting.
- Fertilizer Application: They can test the soil, make fertilizers, and give safety measures in applying.
- Knowledge on the most common pests and diseases affecting local crops and their control methods.
- Soil and Water Management Skills related to soil health, water management, and irrigation techniques.
- Organic Farming Proficiency in sustainable practices that include composting, natural pest control, and organic crop growing.

The structured dataset allows the recommendation engine at AgriSkill to generate accurate and relevant matches such that landowners end up being matched with professionals whose skills suit their specific agricultural requirements.

A.2 Sample Data

User Type	Name	Location	Skills/Help Needed	Number
Skilled Professional	Ramesh Kumar	Thrissur, Kerala	Paddy Cultivation, Soil Preparation, Irrigation	9876543210
Skilled Professional	Anjali Patel	Palakkad, Kerala	Coconut Farming, Tree Planting, Pest Control	9876543211
Skilled Professional	Vinod Nair	Malappuram, Kerala	Rubber Tapping, Latex Processing, Plantation Mgmt	9876543212
Skilled Professional	Seetha Menon	Kollam, Kerala	Pest and Disease Control, Soil Health Management	9876543213
Skilled Professional	Rajeev Menon	Kannur, Kerala	Vegetable Farming, Crop Rotation, Irrigation	9876543214
Landowner	John Mathew	Alappuzha, Kerala	Land Preparation (Banana), Tree Planting (Coconut)	9876543215
Landowner	Lakshmi Pillai	Kozhikode, Kerala	Weeding (Paddy), Soil Preparation	9876543216
Landowner	Manohar Singh	Ernakulam, Kerala	Planting of Coconut Saplings, Organic Farming	9876543217
Landowner	Priya Varghese	Pathanamthitta, Kerala	Pest Control (Vegetables), Fertilizer Application	9876543218
Landowner	Ajay Das	Kasaragod, Kerala	Rubber Plantation Setup, Fertilizer Application	9876543219

Table A.1: Sample Data for AgriSkill

A.3 Sample Source Code

```
#models.py
from django.db import models
from django.contrib.auth.hashers import make_password, check_password as check_hashed_password
from django.core.validators import MinValueValidator, RegexValidator
from django.contrib.auth.models import User
# Base model with shared fields
class UserBase(models.Model):
    full_name = models.CharField(max_length=80)
    email = models.EmailField(unique=True)
    age = models.IntegerField(null=True, blank=True, validators=[MinValueValidator(0)])
    mobile_number = models.CharField(max_length=20, unique=True, validators=[RegexValidator(r'
        ^\+?1?\d{9,15}$')])
    password = models.CharField(max_length=128)
    district = models.CharField(max_length=100, null=True, blank=True)
    city = models.CharField(max_length=100, null=True, blank=True)

    class Meta:
        abstract = True

    def save(self, *args, **kwargs):
        # Hash the password before saving
        if not self.password.startswith('pbkdf2_sha256$'):
            self.password = make_password(self.password)
        super().save(*args, **kwargs)

    def check_password(self, raw_password):
        return check_hashed_password(raw_password, self.password)

# Landowner model
class Landowner(UserBase):
    total_area = models.DecimalField(max_digits=10, decimal_places=2, null=True, blank=True)
    help_needed = models.CharField(max_length=255, null=True, blank=True)

    def __str__(self):
        return f"Landowner: {self.full_name}"

# Skilled Professional model
class SkilledProfessional(UserBase):
    skills = models.CharField(max_length=255, null=True, blank=True)

    def __str__(self):
        return f"Skilled Professional: {self.full_name}"\

class SkillSharePost(models.Model):
    text = models.TextField(blank=True)
    image = models.ImageField(upload_to='skillshare_images/', blank=True, null=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Post {self.id} - {self.text[:30]}"


#urls.py
from django.template.context_processors import static
```

```python
from django.urls import path
from numpy.f2py.crackfortran import namepattern
from . import views  # Import views from the current app
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('role-selection/', views.role_selection, name='role_selection'),  # Role Selection
    path('signup/skilled-professional/', views.skilled_professional_signup, name='skilled_professional_signup'),  # Skilled Professional Signup
    path('signup/landowner/', views.landowner_signup, name='landowner_signup'),  # Landowner Signup
    path('skilled-professional/', views.skilled_professional_home, name='skilled_professional_home'),  # Skilled Professional Home
    path('job-listings/', views.job_listings, name='job_listings'),  # Job Listings
    path('user-messages/', views.user_messages, name='user_messages'),  # User Messages
    path('settings/', views.settings, name='settings'),  # User Settings
    path('landowner-home/', views.landowner_home, name='landowner_home'),  # Landowner Home
    path('select-professional/<int:id>/', views.select_professional, name='select_professional'),  # Select Professional
    path('update-work-location/', views.update_work_location, name='update_work_location'),  # Update Work Location
    path('logout/', views.custom_logout, name='logout'),  # Logout
    path('profile/', views.profile, name='profile'),
    path('matched_professionals/', views.matched_professionals, name='matched_professionals'),
    path('professional/<str:professional_name>/', views.landowner_results, name='landowner_results'),
    path('matched_landowners/', views.matched_landowners, name='matched_landowners'),
    path('skillshare/', views.skillshare_view, name='post_page'),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)


#settings.py
"""
Django settings for Agriskill project.

Generated by 'django-admin startproject' using Django 5.1.1.

For more information on this file, see
https://docs.djangoproject.com/en/5.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.1/ref/settings/
"""

from pathlib import Path
import os

```

```
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-%j@d!ugb5=(cfip^m#yl9+xjpt!@p+-r_ur)9$04wa76*^*30v'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

# Set your allowed hosts
ALLOWED_HOSTS = ['localhost', '127.0.0.1']

# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',  # Include Django REST framework if you are using it
    'Agriskill',  # Your application
    'channels'
]

# Middleware configuration
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'mysite.urls'

# Template configuration
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],  # Add your global templates directory here
        'APP_DIRS': True,  # Enable app directories
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'mysite.wsgi.application'

# Database configuration (Use MySQL or any database you prefer)
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',  # or 'django.db.backends.sqlite3'
        'NAME': 'Agriskill',
        'USER': 'root',
        'PASSWORD': 'Hari#5252',
        'HOST': 'localhost',  # or your database host
        'PORT': '3306',  # or the port your database is using
    }
}
# Password validation settings
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
ASGI_APPLICATION = "myproject.asgi.application"
CHANNEL_LAYERS = {
    "default": {
        "BACKEND": "channels_redis.core.RedisChannelLayer",
        "CONFIG": {
            "hosts": [("127.0.0.1", 6379)],
        },
    },
}
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')


# Internationalization settings
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
STATIC_URL = '/static/'

# Default primary key field type
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

#views.py
from django.contrib.auth.decorators import login_required
from django.db.models import Q
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from django.contrib.auth.hashers import make_password, check_password
from rest_framework.generics import get_object_or_404
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors
from .models import Landowner, SkilledProfessional
import numpy as np
import logging
from .models import SkillSharePost
from django.core.files.storage import FileSystemStorage






# Login View
def user_login(request):
    if request.method == 'POST':
        # Get the login input which could be email or mobile number
        identifier = request.POST.get('identifier')  # 'identifier' for either email or mobile
        password = request.POST.get('password')

        # Attempt to fetch user based on email or mobile number
        user = None
        if identifier:
            # Check both models for user existence
            user = Landowner.objects.filter(email=identifier).first() or Landowner.objects.
                filter(
                mobile_number=identifier).first()
            if not user:  # If not found in Landowner, check SkilledProfessional
                user = SkilledProfessional.objects.filter(
                    email=identifier).first() or SkilledProfessional.objects.filter(
                        mobile_number=identifier).first()

        # Debugging output
        print(f"Identifier: {identifier}")
        print(f"User found: {user}")  # See what user is fetched
        print(f"Password entered: {password}")

        # Check if user exists and verify the password
        if user:
            if user.check_password(password):  # Use the method from UserBase
                # Store user details in session
                request.session['user_email'] = user.email

                # Store role based on user type
                if isinstance(user, Landowner):
                    request.session['user_role'] = 'landowner'
                    return redirect('landowner_home')
                else:  # Assuming it's a SkilledProfessional
                    request.session['user_role'] = 'skilled_professional'
                    return redirect('skilled_professional')
            else:
                messages.error(request, 'Invalid password.')
                print(f"Password check failed for user: {user.email}")  # Debugging output
        else:
            messages.error(request, 'Invalid email or phone number.')

    return render(request, 'login.html')

# Role Selection View
def role_selection(request):
    if request.method == 'POST':
        selected_role = request.POST.get('role')
        if selected_role:
            request.session['selected_role'] = selected_role
            return redirect(
                'skilled_professional_signup' if selected_role == 'skilled_professional' else '
                    landowner_signup'
            )
        messages.error(request, "Please select a role.")
    return render(request, 'role_selection.html')


# Signup View for Skilled Professional
def skilled_professional_signup(request):
    # Initialize the temporary storage dictionary outside of POST
    if 'temporary_skilled_professionals' not in request.session:
        request.session['temporary_skilled_professionals'] = {}

    if request.method == 'POST':
        # Collect input data
        full_name = request.POST.get('full_name')
        email = request.POST.get('email')
        age = request.POST.get('age')
        mobile_number = request.POST.get('mobile_number')
        password = request.POST.get('password')
        confirm_password = request.POST.get('confirm_password')
        skills = request.POST.get('skills')  # Skills are collected here
        district = request.POST.get('district')
        city = request.POST.get('city')

        # Verify passwords match
        if password != confirm_password:
            return render(request, 'signup_skilled_professional.html', {'error': 'Passwords do
                not match'})

        # Temporary storage logic
        temporary_skilled_professionals = request.session['temporary_skilled_professionals']
        temporary_skilled_professionals[email] = {  # Use email as a key
            'full_name': full_name,
            'age': age,
            'mobile_number': mobile_number,
            'skills': skills,
            'district': district,
            'city': city
        }
        print(temporary_skilled_professionals)
        # Save the updated temporary skilled professionals back to the session
        request.session['temporary_skilled_professionals'] = temporary_skilled_professionals

        # Attempt to save the data to the database
        try:
            new_professional = SkilledProfessional(
                full_name=full_name,
                email=email,
                age=age,
                mobile_number=mobile_number,
                password=password,
                skills=skills,  # Skills should be saved here
                district=district,
                city=city
            )
            new_professional.save()
        except Exception as e:
            print(f"Failed to save to DB: {e}")
            return render(request, 'signup_skilled_professional.html', {'error': 'Failed to
                save your information.'})

        return redirect('user_login')  # Redirect to a success page or login

    return render(request, 'signup_skilled_professional.html')

def landowner_signup(request):
    # Check if the selected role is 'landowner'
    if request.session.get('selected_role') != 'landowner':
        return redirect('role_selection')

    if request.method == 'POST':
        full_name = request.POST.get('full_name')
        email = request.POST.get('email')
        phone_number = request.POST.get('mobile_number')
        age = request.POST.get('age')  # Capture the age from the form
        password = request.POST.get('password')
        confirm_password = request.POST.get('confirm_password')

        # Validate that all fields are filled
        if not all([full_name, email, phone_number, age, password, confirm_password]):
            messages.error(request, "All fields are required.")
            return render(request, 'signup_landowner.html')

        # Check if passwords match
        if password != confirm_password:
            messages.error(request, "Passwords do not match.")
            return render(request, 'signup_landowner.html')

        # Hash the password
        hashed_password = make_password(password)

        # Create a new Landowner instance with the age included
        landowner = Landowner(
            full_name=full_name,
            email=email,
            mobile_number=phone_number,
            age=age,  # Add the age field here
            password=hashed_password
        )

        try:
            # Attempt to save the landowner instance
            landowner.save()
            messages.success(request, "Signup successful for Landowner!")
            return redirect('user_login')
        except Exception as e:
            messages.error(request, f"An error occurred: {str(e)}")

    return render(request, 'signup_landowner.html')

# Landowner Home View - Search for Skilled Professionals using TF-IDF and KNN
@login_required
def landowner_home(request):
    if request.method == 'POST':
        # Collecting form data from POST request
        total_area = request.POST.get('total_area')
        skills = request.POST.get('skills')

        # Split the landowner's skills into a list
        landowner_skills = [skill.strip() for skill in skills.split(',') if skill.strip()]

        # Filter all skilled professionals and prepare data for TF-IDF
        skilled_professionals = SkilledProfessional.objects.all()
        professional_skills = [professional.skills for professional in skilled_professionals]
        all_skills = professional_skills + [', '.join(landowner_skills)]

        # Calculate TF-IDF and KNN for matching professionals
        vectorizer = TfidfVectorizer()
        tfidf_matrix = vectorizer.fit_transform(all_skills)
        knn = NearestNeighbors(n_neighbors=5, metric='cosine')
        knn.fit(tfidf_matrix[:-1])

        landowner_vector = tfidf_matrix[-1]
        distances, indices = knn.kneighbors(landowner_vector)
        matching_professionals = [skilled_professionals[int(i)] for i in indices.flatten()]

        # Store matched professionals in session
        request.session['matched_professionals'] = [
            {'name': prof.full_name, 'skills': prof.skills}
            for prof in matching_professionals
        ]

        return redirect('matched_professionals')

    return render(request, 'landowner_home.html')


# Skilled Professional Home View (Merged with Update Work Location)
@login_required
def skilled_professional_home(request):
    user_email = request.session.get('user_email')

    # Check if skilled professional exists
    try:
        skilled_professional = SkilledProfessional.objects.get(email=user_email)
    except SkilledProfessional.DoesNotExist:
        messages.error(request, "Skilled professional not found.")
        return redirect('login')

    if request.method == 'POST':
        # Retrieve the skilled professional's skills
        skilled_professional_skills = skilled_professional.skills

        # Debug: Print skills
        print(f"[DEBUG] Skilled Professional Skills: {skilled_professional_skills}")

        if not skilled_professional_skills:
            messages.warning(request, "Please specify your skills.")
            return redirect('skilled_professional')

        # Split skills into a list
        professional_skills_list = [skill.strip() for skill in skilled_professional_skills.split(',') if skill.strip()]

        # Step 1: Filter landowners by help_needed
        landowners_by_skill = Landowner.objects.filter(
            Q(help_needed__icontains=professional_skills_list[0])  # Start with the first skill for matching
        )

        # Debug: Check landowners filtered by the first skill
        print(f"[DEBUG] Landowners after skill filter: {[
            {'name': l.full_name, 'help_needed': l.help_needed}
            for l in landowners_by_skill
        ]}")

        # Step 2: Further filter using the rest of the skills
        for skill in professional_skills_list[1:]:  # Skip the first skill since it's already included
            landowners_by_skill = landowners_by_skill.filter(help_needed__icontains=skill)

        # Debug: Final check of matching landowners
        print(f"[DEBUG] Final Matching Landowners: {[
            {'name': l.full_name, 'help_needed': l.help_needed}
            for l in landowners_by_skill
        ]}")

        # Store matched landowners in session
        request.session['matched_landowners'] = [
            {
                'name': landowner.full_name,
                'help_needed': landowner.help_needed,
                'district': landowner.district,
                'city': landowner.city
            }
            for landowner in landowners_by_skill
        ]

        # Redirect to matched landowners page after storing results in session
        return redirect('matched_landowners')

    return render(request, 'skilled_professional.html', {'user': skilled_professional})


def custom_logout(request):
    logout(request)
    messages.success(request, "You have been logged out successfully.")
    return redirect('user_login')  # Redirect to login page


def job_listings():
    return None


def profile(request):
    # Check if the user is logged in by verifying session data
    if 'user_email' not in request.session or 'user_role' not in request.session:
        return redirect('user_login')

    user_email = request.session['user_email']
    user_role = request.session['user_role']

    # Fetch the user details based on role
    user = None
    if user_role == 'landowner':
        user = Landowner.objects.filter(email=user_email).first()
    elif user_role == 'skilled_professional':
        user = SkilledProfessional.objects.filter(email=user_email).first()

    # Render profile page with user data
    context = {
        'user': user,
        'role': user_role
    }
    return render(request, 'profile.html', context)

def matched_professionals(request):
    # Retrieve matched professionals from session
    matched_professionals = request.session.get('matched_professionals', [])

    return render(request, 'matched_professionals.html', {'skilled_professionals':
        matched_professionals
    })

def landowner_results(request, professional_name):
    # Fetch the professional by full_name (name field)
    professional = get_object_or_404(SkilledProfessional, full_name=professional_name)

    return render(request, 'landowner_results.html', {
        'professional': professional,
    })
def matched_landowners(request):
    return render(request, 'matched_landowner.html')
# views.py

@login_required
def skillshare_view(request):
    if request.method == 'POST':
        # Handle form submission for new post
        text = request.POST.get('postText', '')
        image = request.FILES.get('postImage')

        # Save post to the database
        post = SkillSharePost(text=text, image=image)
        post.save()

        # Redirect to the same page to display the new post
        return redirect('skillshare')

    # Fetch all posts to display on the SkillShare page
    posts = SkillSharePost.objects.all().order_by('-created_at')
    return render(request, 'skillshare.html', {'posts': posts})

#consumer.py
# consumers.py
import json
from channels.generic.websocket import AsyncWebsocketConsumer

class ChatConsumer(AsyncWebsocketConsumer):
    async def connect(self):
        self.room_name = self.scope['url_route']['kwargs']['room_name']
        self.room_group_name = f"chat_{self.room_name}"

        # Join room group
        await self.channel_layer.group_add(
            self.room_group_name,
            self.channel_name
        )

        await self.accept()

    async def disconnect(self, close_code):
        # Leave room group
        await self.channel_layer.group_discard(
            self.room_group_name,
            self.channel_name
        )

    # Receive message from WebSocket
    async def receive(self, text_data):
        text_data_json = json.loads(text_data)
        message = text_data_json['message']

        # Send message to room group
        await self.channel_layer.group_send(
            self.room_group_name,
            {
                'type': 'chat_message',
                'message': message
            }
        )

    # Receive message from room group
    async def chat_message(self, event):
        message = event['message']

        # Send message to WebSocket
        await self.send(text_data=json.dumps({
            'message': message
        }))
#HTML Pages
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
        }
        .login-container {
            background: white;
            padding: 40px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            max-width: 500px;
            margin: auto;
            margin-top: 100px;
        }
        h2 {
            text-align: center;
            color: #333;
        }
        label {
            display: block;
            margin: 10px 0 5px;
            color: #555;
        }
        input[type="text"],
        input[type="password"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        button {
            background-color: #5cb85c;
            color: white;
            border: none;
            padding: 10px 15px;
            border-radius: 5px;
            cursor: pointer;
            width: 100%;
        }
        button:hover {
            background-color: #4cae4c;
        }
        .signup-link {
            display: block;
            text-align: center;
            margin-top: 20px;
        }
        .signup-link a {
            text-decoration: none;
            color: #5cb85c;
        }
        .signup-link a:hover {
            text-decoration: underline;
        }
        /* Message styles */
        .message {
            margin-bottom: 20px;
            padding: 10px;
            border-radius: 5px;
            text-align: center;
        }
        .message.error {
            background-color: #f8d7da;
            color: #721c24;
        }
        .message.success {
            background-color: #d4edda;
            color: #155724;
        }
    </style>
</head>
<body>

<div class="login-container">
    <h2>Login</h2>
    <!-- Display any messages (like errors or success messages) -->
    {% if messages %}
```

```
        {% for message in messages %}
            <div class="message {% if message.tags %}{{ message.tags }}{% endif %}">
                {{ message }}
            </div>
        {% endfor %}
    {% endif %}

    <form method="POST" action="{% url 'user_login' %}">
        {% csrf_token %}
        <label for="identifier">Email or Phone Number</label>
        <input type="text" id="identifier" name="identifier" required>

        <label for="password">Password</label>
        <input type="password" id="password" name="password" required>

        <button type="submit">Login</button>
    </form>

    <div class="signup-link">
        <p>Don't have an account? <a href="{% url 'role_selection' %}">Sign Up</a></p>
    </div>
</div>

</body>
</html>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Select Your Role</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .role-container {
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
            width: 300px;
            text-align: center;
        }
        h2 {
            color: #333;
            margin-bottom: 20px;
        }
        label {
            display: block;
            margin: 10px 0;
            font-size: 16px;
            text-align: left;
        }
        button {
            width: 100%;
            background-color: #28a745;
            color: white;
            padding: 10px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            margin-top: 20px;
            font-size: 16px;
        }
        button:hover {
            background-color: #218838;
        }
    </style>
</head>
<body>
    <div class="role-container">
        <h2>Select Your Role</h2>
        <form method="POST" action="{% url 'role_selection' %}">
            {% csrf_token %}
            <label>
                <input type="radio" name="role" value="skilled_professional" required>
                Skilled Professional
            </label>
            <label>
                <input type="radio" name="role" value="landowner" required>
                Landowner
            </label>
            <button type="submit">Continue</button>
        </form>
    </div>
</body>
</html>


<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Landowner Signup</title>
    {% load static %}
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }

        .container {
            max-width: 500px;
            margin: 50px auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }

        h1 {
            text-align: center;
            color: #333;
        }

        .form-group {
            margin-bottom: 15px;
        }

        label {
            display: block;
            margin-bottom: 5px;
            font-weight: bold;
        }

        input[type="text"],
        input[type="email"],
        input[type="password"],
        input[type="number"] {
            width: 100%;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }

        button {
            width: 100%;
            padding: 10px;
            background-color: #28a745;
            color: #fff;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }

        button:hover {
            background-color: #218838;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Landowner Signup</h1>
        <form method="POST">
            {% csrf_token %}
            <div class="form-group">
                <label for="full_name">Full Name:</label>
                <input type="text" id="full_name" name="full_name" required>
            </div>
            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" id="email" name="email" required>
            </div>
            <div class="form-group">
                <label for="mobile_number">Mobile Number:</label>
                <input type="text" id="mobile_number" name="mobile_number" required>
            </div>
            <div class="form-group">
                <label for="age">Age:</label>
                <input type="number" id="age" name="age" min="0" required>
            </div>
            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" id="password" name="password" required>
            </div>
            <div class="form-group">
                <label for="confirm_password">Confirm Password:</label>
                <input type="password" id="confirm_password" name="confirm_password" required>
            </div>
            <button type="submit">Signup</button>
        </form>
    </div>
</body>
</html>


<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Signup Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
        }
        form {
            background: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            max-width: 600px;
            margin: auto;
        }
        h2 {
            text-align: center;
            color: #333;
        }
        label {
            display: block;
            margin: 10px 0 5px;
            color: #555;
        }
        input[type="text"],
        input[type="email"],
        input[type="password"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        button {
            background-color: #5cb85c;
            color: white;
            border: none;
            padding: 10px 15px;
            border-radius: 5px;
            cursor: pointer;
            width: 100%;
        }
        button:hover {
            background-color: #4cae4c;
        }
        /* Modal Styles */
        .modal {
                display: none; /* Hidden by default */
                position: fixed;
                z-index: 1;
                left: 0;
                top: 0;
                width: 100%;
                height: 100%;
                overflow: auto;
                background-color: rgb(0,0,0);
                background-color: rgba(0,0,0,0.4);
                padding-top: 60px;
            }
            .modal-content {
                background-color: #fefefe;
                margin: 5% auto;
                padding: 20px;
                border: 1px solid #888;
                width: 80%;
                border-radius: 5px;
            }
            .close {
                color: #aaa;
                float: right;
                font-size: 28px;
                font-weight: bold;
            }
            .close:hover,
            .close:focus {
                color: black;
                text-decoration: none;
                cursor: pointer;
            }
            .skill-category {
                margin: 15px 0;
            }
            .skill-category h3 {
                margin-bottom: 10px;
            }
            .skills {
                display: flex;
                flex-wrap: wrap;
            }
            .skills label {
                margin-right: 15px;
            }
        </style>
</head>
<body>

    <!-- Signup Form -->
    <h2>Signup as Skilled Professional</h2>
    <form method="POST" action="{% url 'skilled_professional_signup' %}">
        {% csrf_token %}
        <label for="full_name">Full Name</label>
        <input type="text" id="full_name" name="full_name" required>

        <label for="email">Email</label>
        <input type="email" id="email" name="email" required>

        <label for="age">Age</label>
        <input type="text" id="age" name="age" required>

        <label for="mobile_number">Mobile Number</label>
        <input type="text" id="mobile_number" name="mobile_number" required>

        <label for="password">Password</label>
        <input type="password" id="password" name="password" required>

        <label for="confirm_password">Confirm Password</label>
        <input type="password" id="confirm_password" name="confirm_password" required>

        <!-- Hidden field to store selected skills -->
        <input type="hidden" id="skills" name="skills">

        <label for="skills">Select Skills</label>
        <button type="button" id="openModal">Choose Skills</button>
        <label for="district">District:</label>
        <input type="text" id="district" name="district" required>
        <label for="city">City:</label>
        <input type="text" id="city" name="city" required>


    <div id="selectedSkills" style="margin: 15px 0;"></div>
        <button type="submit" style="margin-top: 20px;">Sign Up</button>
    </form>
<!-- The Modal -->
<div id="skillsModal" class="modal">
    <div class="modal-content">
        <span class="close">&times;</span>
        <h2>Select Your Skills</h2>

        <div class="skill-category">
            <h3>Banana Cultivation</h3>
            <div class="skills">
                <label><input type="checkbox" name="skill_checkbox" value="Harvesting (Banana)"
                    > Harvesting</label>
                <label><input type="checkbox" name="skill_checkbox" value="Land Preparation (
                    Banana)"> Land Preparation</label>
                <label><input type="checkbox" name="skill_checkbox" value="Planting (Banana)">
                    Planting</label>
                <label><input type="checkbox" name="skill_checkbox" value="Weeding (Banana)"> Weeding</label>
            </div>
        </div>

        <div class="skill-category">
            <h3>Coconut Farming</h3>
            <div class="skills">
                <label><input type="checkbox" name="skill_checkbox" value="Harvesting (Coconut)"> Harvesting</label>
                <label><input type="checkbox" name="skill_checkbox" value="Land Preparation (Coconut)"> Land Preparation</label>
                <label><input type="checkbox" name="skill_checkbox" value="Planting of Coconut Saplings (Coconut)"> Planting of Coconut Saplings</label>
                <label><input type="checkbox" name="skill_checkbox" value="Post-Harvest Processing (Coconut)"> Post-Harvest Processing</label>
            </div>
        </div>

        <div class="skill-category">
            <h3>Paddy Cultivation</h3>
            <div class="skills">
                <label><input type="checkbox" name="skill_checkbox" value="Harvesting (Paddy)"> Harvesting</label>
                <label><input type="checkbox" name="skill_checkbox" value="Land Preparation (Paddy)"> Land Preparation</label>
                <label><input type="checkbox" name="skill_checkbox" value="Ploughing (Paddy)"> Ploughing</label>
                <label><input type="checkbox" name="skill_checkbox" value="Weeding (Paddy)"> Weeding</label>
            </div>
        </div>

        <div class="skill-category">
            <h3>Rubber Tapping</h3>
            <div class="skills">
                <label><input type="checkbox" name="skill_checkbox" value="Latex Collection (Rubber)"> Latex Collection</label>
                <label><input type="checkbox" name="skill_checkbox" value="Processing of Latex (Rubber)"> Processing of Latex</label>
                <label><input type="checkbox" name="skill_checkbox" value="Tapping (Rubber)"> Tapping</label>
                <label><input type="checkbox" name="skill_checkbox" value="Tree Planting (Rubber)"> Tree Planting</label>
            </div>
        </div>
        <button id="saveSkills">Save Skills</button>
    </div>
</div>

```

```
<script>


    const modal = document.getElementById("skillsModal");
    const btn = document.getElementById("openModal");
    const span = document.getElementsByClassName("close")[0];

    btn.onclick = function() {
        modal.style.display = "block";
    }

    span.onclick = function() {
        modal.style.display = "none";
    }

    window.onclick = function(event) {
        if (event.target == modal) {
            modal.style.display = "none";
        }
    }

    document.getElementById('saveSkills').onclick = function() {
        const checkboxes = document.querySelectorAll('input[name="skill_checkbox"]:checked');
        const selectedSkills = Array.from(checkboxes).map(checkbox => checkbox.value);

        document.getElementById('selectedSkills').innerText = selectedSkills.join(', ');
        document.getElementById('skills').value = selectedSkills.join(', ');

        modal.style.display = "none";
    };
</script>
</body>
</html>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Skilled Professional Home</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }
        /* Navbar styling */
        .navbar {
            display: flex;
            justify-content: space-between;
            align-items: center;
            background-color: #333;
            color: #fff;
            padding: 10px 20px;
        }
        .navbar a {
            color: white;
            padding: 14px 20px;
            text-decoration: none;
            text-align: center;
        }
        .navbar a:hover {
            background-color: #575757;
            border-radius: 5px;
        }
        /* Form styling */
        .container {
            padding: 20px;
        }
        form {
            background: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            max-width: 600px;
            margin: auto;
        }
        h2 {
            text-align: center;
            color: #333;
        }
        label {
            display: block;
            margin: 10px 0 5px;
            color: #555;
        }
        input[type="text"], select {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        button {
            background-color: #5cb85c;
            color: white;
            border: none;
            padding: 10px 15px;
            border-radius: 5px;
            cursor: pointer;
            width: 100%;
        }
        button:hover {
            background-color: #4cae4c;
        }
    </style>
</head>
<body>
    <div class="navbar">
        <div class="logo">Agriskill</div>
        <nav>
            <a href="{% url 'logout' %}" class="logout-button">Logout</a>
        </nav>
    </div>
    <div class="container">
        <h1>Welcome to the Skilled Professional Home Page</h1>
        <p>Here you can manage your profile and find job listings.</p>

        <h2>Search for Landowners</h2>
        <form method="POST" action="{% url 'skilled_professional' %}">
            {% csrf_token %}

            <button type="submit">Search</button>
        </form>
    </div>
</body>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Landowner Home</title>
</head>
<style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }
        /* Navbar styling */
        .navbar {
            display: flex;
            justify-content: space-between;
            align-items: center;
            background-color: #333;
            color: #fff;
            padding: 10px 20px;
        }
        .navbar a {
            color: white;
            padding: 14px 20px;
            text-decoration: none;
            text-align: center;
        }
        .navbar a:hover {
            background-color: #575757;
            border-radius: 5px;
        }
        /* Form styling */
        .container {
            padding: 20px;
        }
        form {
            background: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            max-width: 600px;
            margin: auto;
        }
        h2 {
            text-align: center;
            color: #333;
        }
        label {
            display: block;
            margin: 10px 0 5px;
            color: #555;
        }
        input[type="text"], select {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        button {
            background-color: #5cb85c;
            color: white;
            border: none;
            padding: 10px 15px;
            border-radius: 5px;
            cursor: pointer;
            width: 100%;
        }
        button:hover {
            background-color: #4cae4c;
        }
        /* Modal styling */
        .modal {
            display: none;
            position: fixed;
            z-index: 1;
            left: 0;
            top: 0;
            width: 100%;
            height: 100%;
            overflow: auto;
            background-color: rgba(0,0,0,0.4);
            padding-top: 60px;
        }
        .modal-content {
            background-color: #fefefe;
            margin: 5% auto;
            padding: 20px;
            border: 1px solid #888;
            width: 80%;
            border-radius: 5px;
        }
        .close {
            color: #aaa;
            float: right;
            font-size: 28px;
            font-weight: bold;
        }
        .close:hover,
        .close:focus {
            color: black;
            text-decoration: none;
            cursor: pointer;
        }
        .skill-category {
            margin: 15px 0;
        }
        .skill-category h3 {
            margin-bottom: 10px;
        }
        .skills {
            display: flex;
            flex-wrap: wrap;
        }
        .skills label {
            margin-right: 15px;
        }
        #resultList {
            display: none;
            background-color: #f9f9f9;
            padding: 10px;
            border-radius: 5px;
            margin-top: 20px;
        }
        .result-item {
            padding: 10px;
            border: 1px solid #ccc;
            margin: 5px 0;
            cursor: pointer;
        }
        .result-item:hover {
            background-color: #f1f1f1;
        }
</style>
<body>
    <div class="navbar">
        <div><a href="#">Agriskill</a></div>
        <div><a href="templates/post_page.html">SkillShare</a></div>
    </div>
        <div class="container">
            <h2>Find Agricultural Assistance</h2>
            <form id="searchForm" method="POST" action="{% url 'matched_professionals' %}">
                {% csrf_token %}
                <label for="district">District</label>
                <select id="district" name="district" required onchange="populateCities()">
                    <option value="Kollam">Kollam</option>
                    <option value="Pathanamthitta">Pathanamthitta</option>
                    <option value="Alappuzha">Alappuzha</option>
                    <option value="Kottayam">Kottayam</option>
                    <option value="Idukki">Idukki</option>
                    <option value="Ernakulam">Ernakulam</option>
                    <option value="Thrissur">Thrissur</option>
                    <option value="Palakkad">Palakkad</option>
                    <option value="Malappuram">Malappuram</option>
                    <option value="Kozhikode">Kozhikode</option>
                    <option value="Wayanad">Wayanad</option>
                    <option value="Kannur">Kannur</option>
                    <option value="Kasaragod">Kasaragod</option>
                </select>

                <label for="city">City</label>
                <select id="city" name="city" required>
                    <option value="">Select City</option>
                </select>
                <label for="total_area">Total Area (in acres)</label>
                <input type="text" id="total_area" name="total_area" required>
                <input type="hidden" id="skills" name="skills">
                <label for="skills">Select Type of Help Needed (Skills)</label>
                <button type="button" id="openModal">Choose Skills</button>
                <div id="selectedSkills" style="margin: 15px 0;"></div>
                <button type="submit" style="margin-top: 20px;">Search</button>
            </form>
            <div id="resultList">
                {% if skilled_professionals %}
                <h3>Matching Skilled Professionals:</h3>
                {% for professional in skilled_professionals %}
                <div class="card mb-3">
                    <div class="card-body">
                        <h5 class="card-title">
                            <!-- Link to landowner_results by name -->
                            <a href="{% url 'landowner_results' professional.name %}">{{
                                professional.name }}</a>
                        </h5>
                        <p><strong>Skills:</strong> {{ professional.skills }}</p>
                    </div>
                    {% endfor %}
                    {% else %}
                    <p>No matching skilled professionals found based on the specified help
                        needed.</p>
                    {% endif %}
                </div>
            </div>
            <!-- Modal for Skill Selection -->
            <div id="skillsModal" class="modal">
                <div class="modal-content">
                    <span class="close">&times;</span>
                    <h2>Select Your Skills</h2>

                    <div class="skill-category">
                        <h3>Banana Cultivation</h3>
                        <div class="skills">
                            <label><input type="checkbox" name="skill_checkbox" value="
                                Harvesting (Banana)"> Harvesting</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Land
                                Preparation (Banana)"> Land Preparation</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Planting
                                 (Banana)"> Planting</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Weeding
                                (Banana)"> Weeding</label>
                        </div>
                    </div>

                    <div class="skill-category">
                        <h3>Coconut Farming</h3>
                        <div class="skills">
                            <label><input type="checkbox" name="skill_checkbox" value="
                                Harvesting (Coconut)"> Harvesting</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Land Preparation (Coconut)"> Land Preparation</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Planting of Coconut Saplings (Coconut)"> Planting of Coconut Saplings</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Post-Harvest Processing (Coconut)"> Post-Harvest Processing</label>
                        </div>
                    </div>

                    <div class="skill-category">
                        <h3>Paddy Cultivation</h3>
                        <div class="skills">
                            <label><input type="checkbox" name="skill_checkbox" value="Harvesting (Paddy)"> Harvesting</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Land Preparation (Paddy)"> Land Preparation</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Ploughing (Paddy)"> Ploughing</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Weeding (Paddy)"> Weeding</label>
                        </div>
                    </div>

                    <div class="skill-category">
                        <h3>Rubber Tapping</h3>
                        <div class="skills">
                            <label><input type="checkbox" name="skill_checkbox" value="Latex Collection (Rubber)"> Latex Collection</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Weeding (Rubber)"> Weeding</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Planting (Rubber)"> Planting</label>
                            <label><input type="checkbox" name="skill_checkbox" value="Tapping (Rubber)"> Tapping</label>
                        </div>
                    </div>

                    <button id="saveSkills">Save Skills</button>
                </div>
            </div>
        </div>
</body>
<script>
            const cities = {
                'Kollam': ['Kollam City', 'Paravoor', 'Kottarakkara'],
                'Pathanamthitta': ['Pathanamthitta City', 'Thiruvalla', 'Kumbazha'],
                'Alappuzha': ['Alappuzha City', 'Cherthala', 'Ambalappuzha'],
                'Kottayam': ['Kottayam City', 'Changanassery', 'Puthuppally'],
                'Idukki': ['Idukki Town', 'Munnar', 'Thodupuzha'],
                'Ernakulam': ['Kochi', 'Aluva', 'Perumbavoor'],
                'Thrissur': ['Thrissur City', 'Chalakudy', 'Irinjalakuda'],
                'Palakkad': ['Palakkad City', 'Kanjirappally', 'Ottapalam'],
                'Malappuram': ['Malappuram City', 'Kondotty', 'Ponnani'],
                'Kozhikode': ['Kozhikode City', 'Koyilandy', 'Vatakara'],
                'Wayanad': ['Kalpetta', 'Mananthavady', 'Vythiri'],
                'Kannur': ['Kannur City', 'Thalassery', 'Payyannur'],
                'Kasaragod': ['Kasaragod Town', 'Parappa', 'Manjeshwaram']
            };

            function populateCities() {
                const districtSelect = document.getElementById('district');
                const citySelect = document.getElementById('city');
                const selectedDistrict = districtSelect.value;

                // Clear previous options
                citySelect.innerHTML = '<option value="">Select City</option>';

                if (selectedDistrict in cities) {
                    cities[selectedDistrict].forEach(function(city) {
                        const option = document.createElement('option');
                        option.value = city;
                        option.textContent = city;
                        citySelect.appendChild(option);
                    });
                }
            }

            document.getElementById('openModal').onclick = function() {
                document.getElementById('skillsModal').style.display = 'block';
            };

            document.getElementsByClassName('close')[0].onclick = function() {
                document.getElementById('skillsModal').style.display = 'none';
            };

            document.getElementById('saveSkills').onclick = function() {
                const checkboxes = document.querySelectorAll('input[name="skill_checkbox"]:
                    checked');
                const selectedSkills = Array.from(checkboxes).map(checkbox => checkbox.value);
                document.getElementById('selectedSkills').innerText = selectedSkills.join(', ')
                    ;
                document.getElementById('skills').value = selectedSkills.join(', ');
                document.getElementById('skillsModal').style.display = 'none';
            };

            window.onclick = function(event) {
                const modal = document.getElementById('skillsModal');
                if (event.target == modal) {
                    modal.style.display = 'none';
```

```
                    }
                };

                function showDetails(name, email, phone, skills) {
                    alert(`Name: ${name}\nEmail: ${email}\nPhone: ${phone}\nSkills: ${skills}`);
                }
</script>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>My Profile</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #f9f9f9;
        }
        h1 {
            color: #333;
        }
        .profile-info {
            background: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            max-width: 500px;
            margin: auto;
        }
        .profile-info p {
            margin: 10px 0;
            color: #555;
        }
        a {
            display: inline-block;
            margin-top: 20px;
            text-decoration: none;
            color: #007BFF;
            font-weight: bold;
        }
        a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <h1>My Profile</h1>
    <div class="profile-info">
        <p><strong>Full Name:</strong> {{ user.full_name }}</p>
        <p><strong>Email:</strong> {{ user.email }}</p>
        <p><strong>Age:</strong> {{ user.age }}</p>
        <p><strong>Mobile Number:</strong> {{ user.mobile_number }}</p>
        <p><strong>District:</strong> {{ user.district }}</p>
        <p><strong>City:</strong> {{ user.city }}</p>

        {% if user|hasattr:"total_area" %}
            <p><strong>Total Area:</strong> {{ user.total_area }} acres</p>
            <p><strong>Help Needed:</strong> {{ user.help_needed }}</p>
        {% elif user|hasattr:"skills" %}
            <p><strong>Skills:</strong> {{ user.skills }}</p>
        {% endif %}
    </div>
    <a href="{% url 'home' %}">Back to Home</a>
</body>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Profile</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
        }
        .profile-container {
            max-width: 600px;
            margin: 0 auto;
            background-color: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            text-align: center;
            color: #333;
        }
        .profile-detail {
            margin: 10px 0;
            font-size: 16px;
            color: #555;
        }
        navbar {
            display: flex;
            justify-content: space-between;
            align-items: center;
            background-color: #333;
            color: #fff;
            padding: 10px 20px;
        }
        .navbar a {
            color: white;
            padding: 14px 20px;
            text-decoration: none;
            text-align: center;
        }
        .navbar a:hover {
            background-color: #575757;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <div class="navbar">
        <div><a href="matched_professionals.html">Agriskill</a></div>
    </div>

<div class="profile-container">
    <h2>User Profile</h2>
    <p class="profile-detail"><strong>Name:</strong> {{ user.full_name }}</p>
    <p class="profile-detail"><strong>Email:</strong> {{ user.email }}</p>
    <p class="profile-detail"><strong>Phone Number:</strong> {{ user.mobile_number }}</p>
    <p class="profile-detail"><strong>Role:</strong> {{ role|title }}</p>

    {% if role == "skilled_professional" %}
        <p class="profile-detail"><strong>Skills:</strong> {{ user.skills }}</p>
    {% endif %}
</div>

</body>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Matched Skilled Professionals</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="
        stylesheet">
    <style>
        /* Custom Styling */
        body {
            background-color: #f9f9f9;
            font-family: Arial, sans-serif;
        }
        .container {
            margin-top: 50px;
            max-width: 800px;
        }
        .card {
            margin-bottom: 20px;
            border-radius: 10px;
            box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
            transition: transform 0.3s;
            cursor: pointer;
        }
        .card:hover {
            transform: scale(1.05);
            box-shadow: 0px 6px 10px rgba(0, 0, 0, 0.2);
        }
        .card-header {
            background-color: #007bff;
            color: white;
            font-weight: bold;
            border-radius: 10px 10px 0 0;
        }
        .card-body {
            padding: 15px;
        }
        .no-results {
            font-size: 1.2rem;
            color: #555;
        }
        navbar {
            display: flex;
            justify-content: space-between;
            align-items: center;
            background-color: #333;
            color: #fff;
            padding: 10px 20px;
        }
        .navbar a {
            color: white;
            padding: 14px 20px;
            text-decoration: none;
            text-align: center;
        }
        .navbar a:hover {
            background-color: #575757;
            border-radius: 5px;
        }
</style>
```

```
</head>
<body>
    <div class="navbar">
        <div><a href="landowner_home.html">Agriskill </a></div>
    </div>
    <div class="container">
        <h1 class="text-center mb-4">Matched Skilled Professionals</h1>

        {% if skilled_professionals %}
            <div class="row">
                {% for professional in skilled_professionals %}
                    <div class="col-md-6">
                        <a href="{% url 'landowner_results' professional.name %}" class="text-
                            white"><style="text-decoration: none;">
                            <div class="card">
                                <div class="card-header">
                                    {{ professional.name }}
                                </div>
                                <div class="card-body">
                                    <p><strong>Skills:</strong> {{ professional.skills }}</p>
                                </div>
                            </div>
                        </a>
                    </div>
                {% endfor %}
            </div>
        {% else %}
            <p class="no-results text-center">No matching skilled professionals found.</p>
        {% endif %}
    </div>
</body>
</html>

<!-- matched_landowner.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Matched Landowners</title>
    <style>
        /* Add your styles here */
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
        }
        .container {
            max-width: 800px;
            margin: auto;
            background: white;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            text-align: center;
            color: #333;
        }
        .landowner {
            border: 1px solid #ccc;
            padding: 10px;
            margin-bottom: 10px;
            border-radius: 5px;
        }
        navbar {
            display: flex;
            justify-content: space-between;
            align-items: center;
            background-color: #333;
            color: #fff;
            padding: 10px 20px;
        }
        .navbar a {
            color: white;
            padding: 14px 20px;
            text-decoration: none;
            text-align: center;
        }
        .navbar a:hover {
            background-color: #575757;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <div class="navbar">
        <div><a href="landowner_home.html">Agriskill</a></div>
    </div>

    <div class="container">
        <h2>Matched Landowners</h2>

        {% if matching_landowners %}
            {% for landowner in matching_landowners %}
                <div class="landowner">
                    <p><strong>Name:</strong> {{ landowner.full_name|title }}</p>
                    <p><strong>Email:</strong> {{ landowner.email }}</p>
                    <p><strong>Help Needed:</strong> {{ landowner.help_needed }}</p>
                    </div>
                {% endfor %}
            {% else %}
                <p>No matching landowners found.</p>
            {% endif %}
        </div>

</body>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Professional Details</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="
        stylesheet">
    <style>
        body {
            background-color: #f8f9fa;
            height: 100vh; /* Full height for background */
            display: flex;
            align-items: center; /* Center vertically */
            justify-content: center; /* Center horizontally */
            padding: 20px; /* Padding around */
        }
        .container {
            background-color: #ffffff;
            border-radius: 8px;
            padding: 40px;
            box-shadow: 0 2px 15px rgba(0, 0, 0, 0.1);
            width: 100%; /* Full width */
            max-width: 900px; /* Max width for larger screens */
        }
        h2 {
            text-align: center;
            font-weight: bold;
            text-transform: uppercase; /* Capitalizing the name */
            margin-bottom: 20px; /* Spacing below the heading */
        }
        #chatbox-container {
            display: none;
            position: fixed;
            bottom: 20px;
            right: 20px;
            width: 300px;
            border: 1px solid #ccc;
            border-radius: 8px;
            background: #ffffff;
            box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);
            z-index: 1000;
        }
        #chatbox {
            max-height: 200px;
            overflow-y: auto;
            padding: 10px;
            border-bottom: 1px solid #ccc;
        }
        #chat-button {
            position: fixed;
            bottom: 20px;
            right: 20px;
            font-size: 24px;
            background: #007bff;
            color: white;
            border: none;
            border-radius: 50%;
            padding: 10px;
            cursor: pointer;
            z-index: 1000;
        }
        #chat-button:hover {
            background: #0056b3;
        }
        navbar {
            display: flex;
            justify-content: space-between;
            align-items: center;
            background-color: #333;
            color: #fff;
            padding: 10px 20px;
        }
        .navbar a {
            color: white;
            padding: 14px 20px;
            text-decoration: none;
            text-align: center;
        }
        .navbar a:hover {
            background-color: #575757;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <div class="navbar">
        <div><a href="landowner_home.html">Agriskill</a></div>
    </div>
    <div class="container">
        <h2>{{ professional.full_name|upper }}</h2> <!-- Capitalizing the name -->
        <p><strong>Skills:</strong> {{ professional.skills }}</p>
```

```
            <p><strong>Age:</strong> {{ professional.age }}</p>
            <p><strong>Email:</strong> {{ professional.email }}</p>
            <p><strong>Phone:</strong> {{ professional.mobile_number }}</p>
        </div>

    <button id="chat-button">      </button>

    <div id="chatbox-container">
        <div id="chatbox">
            <p><strong>Chat:</strong></p>
        </div>
        <div class="input-group">
            <input id="message-input" type="text" class="form-control" placeholder="Type your message">
            <button id="send-btn" class="btn btn-primary">Send</button>
        </div>
    </div>

    <script>
        const chatButton = document.getElementById('chat-button');
        const chatboxContainer = document.getElementById('chatbox-container');
        const sendButton = document.getElementById('send-btn');
        const messageInput = document.getElementById('message-input');
        const chatbox = document.getElementById('chatbox');

        chatButton.addEventListener('click', () => {
            chatboxContainer.style.display = chatboxContainer.style.display === 'none' ? 'flex' : 'none';
        });

        // WebSocket URL (replace 'room_name' with dynamic room name from Django)
        const roomName = "{{ professional.full_name }}";
        const chatSocket = new WebSocket(
            `ws://${window.location.host}/ws/chat/${roomName}/`
        );

        // Listen for WebSocket messages
        chatSocket.onmessage = function(e) {
            const data = JSON.parse(e.data);
            const message = data.message;
            const messageElement = document.createElement('p');
            messageElement.textContent = message;
            chatbox.appendChild(messageElement);
            chatbox.scrollTop = chatbox.scrollHeight;
        };

        chatSocket.onclose = function(e) {
            console.error('Chat socket closed unexpectedly');
        };

```

```
            sendButton.addEventListener('click', () => {
                const message = messageInput.value.trim();
                if (message) {
                    chatSocket.send(JSON.stringify({ 'message': "You: " + message }));
                    messageInput.value = '';
                }
            });
        </script>
</body>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Job Listings</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #f9f9f9;
        }
        h1 {
            color: #333;
        }
        a {
            display: inline-block;
            margin-top: 20px;
            text-decoration: none;
            color: #007BFF;
            font-weight: bold;
        }
        a:hover {
            text-decoration: underline;
        }
        navbar {
            display: flex;
            justify-content: space-between;
            align-items: center;
            background-color: #333;
            color: #fff;
            padding: 10px 20px;
        }
        .navbar a {
            color: white;
            padding: 14px 20px;
            text-decoration: none;
            text-align: center;
        }
        .navbar a:hover {
            background-color: #575757;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <div class="navbar">
        <div><a href="landowner_home.html">Agriskill</a></div>
    </div>
    <h1>Job Listings</h1>
    <p>Job listings will be displayed here.</p>
    <a href="{% url 'skilled_professional' %}">Back to Home</a>
</body>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SkillShare</title>
</head>
<style>
    body {
        font-family: Arial, sans-serif;
        background-color: #f4f4f4;
        margin: 0;
        padding: 0;
    }
    .navbar {
        display: flex;
        justify-content: space-between;
        align-items: center;
        background-color: #333;
        color: #fff;
        padding: 10px 20px;
    }
    .navbar a {
        color: white;
        padding: 14px 20px;
        text-decoration: none;
        text-align: center;
    }
    .navbar a:hover {
        background-color: #575757;
        border-radius: 5px;
    }
    .container {
        padding: 20px;
        max-width: 800px;
        margin: auto;
    }
    .post {
        background: white;
        padding: 20px;
        border-radius: 5px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        margin-bottom: 15px;
    }
    .post img {
        max-width: 100%;
        border-radius: 5px;
        margin-top: 10px;
    }
    .add-post-btn {
        display: flex;
        justify-content: flex-end;
        margin-top: 10px;
    }
    .add-post-icon {
        font-size: 24px;
        cursor: pointer;
        background-color: #5cb85c;
        color: white;
        padding: 10px;
        border-radius: 50%;
        text-align: center;
    }
    /* Modal styling */
    .modal {
        display: none;
        position: fixed;
        z-index: 1;
        left: 0;
        top: 0;
        width: 100%;
        height: 100%;
        overflow: auto;
        background-color: rgba(0, 0, 0, 0.4);
        padding-top: 60px;
    }
    .modal-content {
        background-color: #fefefe;
        margin: 5% auto;
        padding: 20px;
        border: 1px solid #888;
        width: 80%;
        max-width: 500px;
        border-radius: 5px;
    }
    .close {
        color: #aaa;
        float: right;
        font-size: 28px;
        font-weight: bold;
    }
    .close:hover, .close:focus {
        color: black;
        text-decoration: none;
        cursor: pointer;
    }
    .form-group {
        margin-bottom: 15px;
    }
    label {
        font-weight: bold;
    }
    textarea, input[type="file"] {
        width: 100%;
        padding: 10px;
        margin-top: 5px;
        border: 1px solid #ccc;
        border-radius: 5px;
    }
    button {
        background-color: #5cb85c;
        color: white;
        padding: 10px;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }
    button:hover {
        background-color: #4cae4c;
    }
</style>
<body>
    <div class="navbar">
        <div><a href="#">Agriskill</a></div>
    </div>

    <div class="container">
        <h2>SkillShare</h2>

        <!-- Posts from other users -->
        <div class="post">
            <p><strong>John Doe</strong>: Sharing tips on sustainable coconut farming.</p>
            <img src="coconut-farming.jpg" alt="Coconut Farming">
        </div>
        <div class="post">
            <p><strong>Mary Smith</strong>: How to efficiently tap rubber trees.</p>
        </div>

        <!-- Add Post Button -->
        <div class="add-post-btn">
            <div class="add-post-icon" onclick="document.getElementById('addPostModal').style.
                display='block'">+</div>
        </div>

        <!-- Modal for Adding New Post -->
        <div id="addPostModal" class="modal">
            <div class="modal-content">
                <span class="close" onclick="document.getElementById('addPostModal').style.
                    display='none'">&times;</span>
                <h3>New Post</h3>
                <form method="POST" enctype="multipart/form-data">
                    <div class="form-group">
                        <label for="postText">Post Text:</label>
                        <textarea id="postText" name="postText" rows="4" placeholder="Write
                            something..."></textarea>
                    </div>
                    <div class="form-group">
                        <label for="postImage">Upload Image:</label>
                        <input type="file" id="postImage" name="postImage" accept="image/*">
                    </div>
                    <button type="submit">Post</button>
                </form>
            </div>
        </div>
    </div>
</body>
<script>
    // Close modal when clicking outside content
    window.onclick = function(event) {
        const modal = document.getElementById('addPostModal');
        if (event.target == modal) {
            modal.style.display = 'none';
        }
    };
</script>
</html>
output
```

References

[1] Samya Pathirage, Athula Ginige, Design of an Online Platform for the Agriculture Centific Knowledge and Foster Sustainability. IEEE International Research Conference on Smart Computing and Systems Engineering (SCSE).

[2] Siddhartha Paul Tiwari, Information and communication technology initiatives for knowledge sharing in agriculture. Nat. Commun., vol. 11, no. 3923, 2020.

[3] Sidi Sanyang, Sibiri Jean-Baptiste Taonda, Julienne Kuiseu, N'Tji Coulibaly,Laban Konate, A paradigm shift in ´ African agricultural research for development: the role of innovation platforms. EEE International Research Conference on Smart Computing and Systems Engineering (SCSE). vol. 54, no. 153-211 2019

[4] Giulio Ermanno Pibiri, Rossano Venturini, Inverted Index Compression. IEEE Conference on Computer Vision and Pattern Recognition (CVP) 2021.

[5] Victor Lempitsky, The inverted multi-index. IEEE Conference on Computer Vision and Pattern Recognition (CVP) 2020.

[6] Cai-zhi Liu, Yan-xiu Sheng, Yong-Quan Yang, Research of Text Classification Based on Improved TF-IDF Algorithm. IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE), 2018.

[7] Prafulla Bafna, Dhanya Pramod, and Anagha Vaidya, Document clustering: TF-IDF approach.International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2022.

[8] Seyyed Mohammad Hossein Dadgar, Mohammad Shirzad Araghi and Morteza Mastery Farahani, A novel text mining approach based on TF-IDF and Support Vector Machine for news classification. IEEE International Conference on Engineering and Technology (ICETECH), 2022.

[9] H. Liang, J. Liu, Y. Yuan, and D. Thalmann, Text Features Extraction based on TF-IDF Associating Semantic. IEEE 4th International Conference on Computer and Communications (ICCC), vol. 50, no. 4, pp. 1833–1844, 2020

[10] Alfirna Rizqi Lahitani, Adhistya Erna Permanasari,Noor Akhmad Setiawan, K-nn algorithm to determine similarity measure: Study case in on-

line essay assessment. 4th International Conference on Cyber and IT Service Management, vol. 70, pp. 683–689, 2015.

[11] Liming Zheng,Ke Jia, and J. S. Liu, Cosine Similarity and K-nn algorithm Based Line Protection for Large-Scale Wind Farms. IEEE Transactions on Industrial Electronics Vol. 68, Issue: 7, 2021.

[12] Lailil Muflikhah,Baharum Baharudin, Document Clustering Using Concept Space and Cosine Similarity Measurement, International Conference on Computer Technology and Development, 2018.

[13] Ari Aulia Hakim, Alva Erwin, Kho I Eng, Maulahikmah Galinium, Automated document classification for news article in Bahasa Indonesia based on term frequency inverse document frequency (TF-IDF) approach. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 6, pp. 947–956, 2021.

[14] A. Dreuw, D. Rybach, T. Deselaers, et al., Implementation of TF-IDF and K-nn Algorithms for Document Classification Based on Abstracts of Scientific Journals . IEEE Conference on Multimodal Corpora: From Models of Natural Interaction to Systems and Applications, pp. 41–46, 2021.

[15] Gisela Yunanda, Dade Nurjanah , and Selly Meliana , Recommendation System from Microsoft News Data using TF-IDF andK-NN Algorithms. Proceedings of the IEEE International Conference on Automatic Face Gesture Recognition, pp. 287–294, 2017.

[16] T. Starner and A. Pentland,Measure the Similarity of Complaint Document Using Cosine Similarity Based on ClassBased Indexing. International Journal of Computer Applications Technology and Research, vol. 7 Issue 08, pp. 292-296,2018.

[17] PV. Snigdha, S.Rahul and M. Naveen, Movie recommendation system using TF-IDF vectorization andK-NN algorithm, IEEE Conference on Multimodal Corpora: From Models of Natural Interaction to Systems and Applications, pp. 411–462, 2021.

[18] Lubab A. Salman,Nurul Hashimah Ahamed, Yu-N Cheah, andAmmar Ismael Kadhim,Feature extraction for cooccurrence-based cosine similarity score of text documents with K-NN Algorithm. IEEE Student Conference on Research and Development, 2022.

[19] Xu, J., Yin, Z., and Zhao, J. ”An Improved Job Recommendation Algorithm Based on Job Matching.” In 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 20-22 Oct. 2017.

[20] Liang, H., and Qian, M. ”A Personalized Job Recommendation System Based on Collaborative Filtering and TFIDF.” In 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 23-25 Nov. 2018.