K.RAMAKRISHNAN
COLLEGE OF ENGINEERING
An Autonomous Institution
Permanently Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015, 14001:2015 certified institution, Accredited by NBA and with A grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.

A PROJECT REPORT

on

LAND RECORD MANAGEMENT SYSTEM

Submitted in partial fulfillment of requirements for the award of the course of

ECA1121 – PYTHON PROGRAMMING

Under the guidance of

M.INDHU.,ME

Assistant Professor/ECE

Submitted By

SHAYANA R (8115U23EC100)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION AND ENGINERRING

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(An Autonomous Institution, affiliated to Anna University Chennai and
Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

MAY 2024

K.RAMAKRISHNAN
COLLEGE OF ENGINEERING
An Autonomous Institution
Permanently Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015, 14001:2015 certified institution, Accredited by NBA and with A grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project report titled **"LAND RECORD MANAGEMENT SYSTEM "** is the bonafide work of **SHAYAMNA R(8115U23EC100)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr.M.MAHESWARI,Ph.D.,	**M.INDHU.ME.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
ASSISTANT PROFESSOR	**ASSISTANT PROFESSOR,**
Department Electronics and Communication Engineering,	Department of Electronics and Communication Engineering,,
K. Ramakrishnan College of Engineering (Autonomous)	K. Ramakrishnan College of Engineering (Autonomous)
Samayapuram – 621 112	Samayapuram – 621 112

Submitted for the End Semester Examination held on …………….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I jointly declare that the project report on **“LAND RECORD MANAGEMENT SYSTEM USING PYTHON** “is the result of original work done by us and best of our knowledge, similar work has not been submitted to “ANNA UNIVERSITY CHENNAI” for the requirement of Degree of BACHELOR OF ENGINEERING. This project report is submitted on the partial fulfillment of the requirement of the award of degree of BACHELOR OF ENGINEERING.

Signature

SHAYANA R

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtedness to our institution, "**K. Ramakrishnan College of Engineering (Autonomous)**", for providing us with the opportunity to do this project.

We extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering an adequate duration to complete it.

We would like to thank **Dr. D. SRINIVASAN, B.E, M.E., Ph.D.,** Principal, who gave the opportunity to frame the project to full satisfaction.

We thank **Dr. M . MAHESWARI M.tech.,Ph.D.,** Head of the Department of **ELECTRONICS AND COMMUNICATION ENGINEERING**, for providing her encouragement in pursuing this project.

We wish to convey our profound and heartfelt gratitude to our esteemed project guide **M.INDHU.ME.,** Department of **ELECTRONICS AND COMMUNICATION ENGINEERING,** for her
incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

We render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

<u>DEPARTMENT OF ECE</u>

<u>VISION</u>

To be distinguished as a prominent program in Electronics and Communication Engineering Studies by preparing students for Industrial Competitiveness and Societal Challenges.

<u>MISSION</u>

M1. To equip the students with latest technical, analytical and practical knowledge

M2. To provide vibrant academic environment and Innovative Research culture

M3. To provide opportunities for students to get Industrial Skills and Internships to meet out the challenges of the society.

<u>PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)</u>

PEO1: Graduates will become experts in providing solution for the Engineering problems in Industries, Government and other organizations where they are employed.

PEO2: Graduates will provide innovative ideas and management skills to enhance the standards of the society by individual and with team works through the acquired Engineering knowledge.

PEO3: Graduates will be successful professionals through lifelong learning and contribute to the society technically and professionally.

PROGRAM SPECIFIC OUTCOMES (PSO'S)

PSO1: Students will qualify in National level Competitive Examinations for Employment and Higher studies.

PSO2: Students will have expertise in the design and development of Hardware and Software tools to solve complex Electronics and Communication Engine erring problems in the domains like analog and digital electronics, embedded and communication systems.

PROGRAM OUTCOME

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriatetechniques, resources, and modern engineering and IT tools including

PROGRAM SPECIFIC OUTCOMES (PSO'S)

prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large. Some of the mare, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Lifelong learning: Recognize the need for, and have the preparation and lifelong learning in the broadest context of technological change.

ABSTRACT

The Land Record Management System (LRMS) is a software solution developed to efficiently manage and maintain land records, replacing traditional manual record keeping practices. Built using Python, the system offers a user-friendly text-based interface for creating, retrieving, updating, and deleting land records. Key functions include adding new records with details like owner name, location, size, and value; viewing all existing records; updating current records; and deleting outdated or incorrect records. Designed for use by government agencies, real estate firms, and other organizations involved in land administration, the LRMS enhances efficiency, accuracy, and transparency in land record management. This system ensures data integrity and facilitates easy maintenance and retrieval of records, making it a scalable and reliable solution for modern land administration needs.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction:

In today's ever-evolving digital landscape, efficient management of land records is crucial for maintaining transparency, reducing disputes, and facilitating smooth transactions. Enter the Land Record Management System, a comprehensive solution designed to revolutionize how land data is collected, stored, and accessed. This system offers a centralized platform where all pertinent information regarding land ownership, boundaries, transactions, and history can be securely managed and updated in real-time. With features such as digitized records, GIS mapping, and online accessibility, the Land Record Management System empowers governments, land administrators, and citizens alike to streamline processes, enhance accountability, and foster sustainable land governance. Join us as we embark on a journey towards a more transparent, efficient, and equitable land management system.

1.2 Purpose and Importance

Purpose and Importance of the Land Record Management System:

The Land Record Management System serves as a fundamental tool for modernizing land administration practices and promoting effective land governance. Its purpose lies in addressing the following key objectives.

1. Transparency and Accountability: By digitizing land records and making them easily accessible, the system enhances transparency in land transactions, reducing the potential for fraud and corruption. This fosters trust among stakeholders and ensures accountability in land management

2. Efficiency and Convenience: Streamlining the management of land records through a centralized digital platform eliminates the need for manual paperwork and cumbersome administrative procedures. This results in faster processing times for land-related transactions, saving time and resources for both government agencies and citizens.

3. Dispute Resolution: Accurate and up-to-date land records facilitate quick and reliable resolution of land disputes. With comprehensive data on property ownership, boundaries, and history readily available, conflicts can be resolved more efficiently, minimizing legal complexities and associated costs.

4. Land Use Planning and Development: The system provides valuable data for urban planning, infrastructure development, and environmental management initiatives. By analyzing spatial information on land use patterns and trends, policymakers can make informed decisions to promote sustainable development and optimize resource allocation.

5. Citizen Empowerment: Accessible online portals and interactive maps empower citizens to actively participate in land-related processes, such as property registration, land valuation, and taxation. This promotes inclusivity and democratic governance, enabling individuals to exercise their rights and responsibilities as landowners or tenants.

In essence, the Land Record Management System plays a vital role in modernizing land administration, promoting good governance, and supporting sustainable development initiatives. Its importance cannot be overstated, as it serves as the foundation for fair, transparent, and efficient land management practices in both rural and urban contexts.

1.3 Objectives

1. Digitization
2. Accuracy and Integrity
3. Transparency and Accountability
4. Efficiency and Streamlining
5. Capacity Building

1.3 Project Summarization

The Land Record Management System is a transformative project aimed at modernizing land administration practices and promoting effective governance. Its core objectives include digitizing land records, ensuring accuracy and integrity, enhancing transparency and accountability, streamlining processes, facilitating dispute resolution, supporting spatial planning and development, empowering citizens, ensuring security and privacy, promoting interoperability, and capacity building. By centralizing land data, providing open access to records, and leveraging technology, the system seeks to foster trust, efficiency, and sustainability in land management,

thereby driving economic growth and social development.

CHAPTER 2

PROJECT METHODOLOGY

1.1 Introduction to System Architecture

Introduction to Land Record Management System Architecture:

The Land Record Management System (LRMS) architecture is the backbone of a comprehensive solution designed to revolutionize the management of land records. At its core, the architecture encompasses a range of components, technologies, and functionalities, all working seamlessly together to digitize, organize, and maintain land-related data.

Overall, the Land Record Management System architecture is designed to provide a robust, scalable, and interoperable framework for managing land records effectively. By leveraging modern technologies and best practices.

1.1.1 High-Level System Architecture

The high-level system architecture for the Land Record Management system typically consists of several key components:

(i) User Interface (UI)

(ii) Application Logic

1.1.2 Components of the System Architecture

In the Land Record Management System (LRMS) project, the User Interface:

a. User Interface (UI):

The UI presents user-friendly forms for data entry, allowing users to input or update land record information. These forms are designed with intuitive input fields, dropdown menus, and validation checks to ensure accurate and complete data entry.

b. Application Logic:

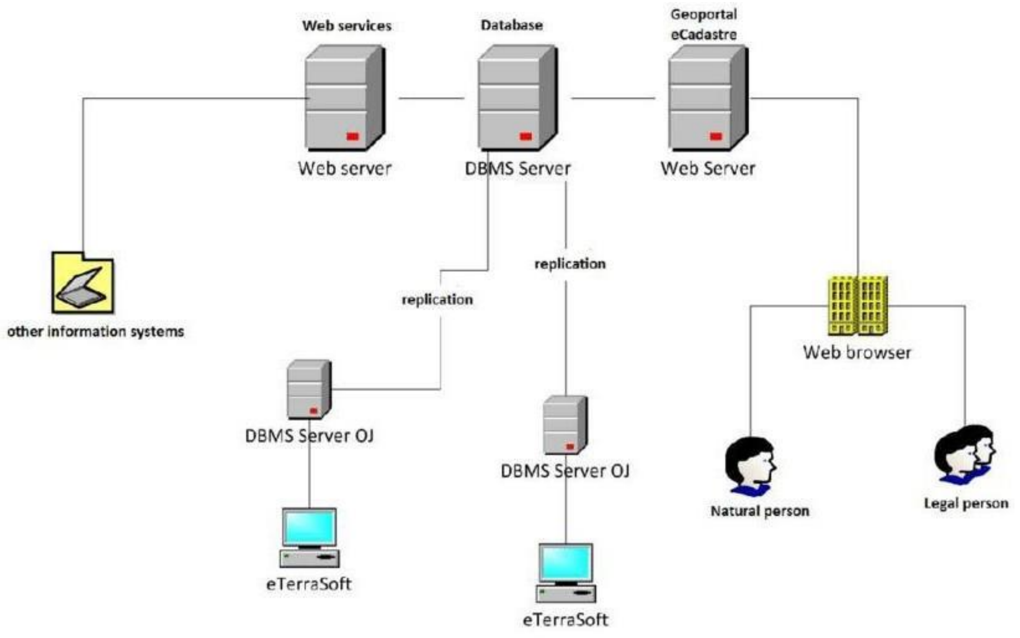
The application logic validates user inputs and ensures data integrity by implementing validation rules and checks on data entered through the UI. This prevents incorrect or incomplete data from being saved to the database.

c. Data management layer :

Creating a data management layer for a Land Record Management System (LRMS) involves designing classes and methods that interact with a database to manage land records.

1.2 Detailed System Flow chart Diagram

Creating a visual representation of the architecture of a Land Record Management System (LRMS) involves illustrating the various components and their interactions. Here's a simplified flow chart diagram for an LRMS:



Architecture Diagram

CHAPTER 3
PYTHON PREFERENCE

1. Ease of Development: Python's simple and readable syntax makes the easy to develop and maintain code, reducing development time

2.Rich Ecosystem: Python has a vast ecosystem of libraries and frameworks, such as SQLAlchemy for database interaction, Flask or Django for web development, and Pandas for data manipulation, which can accelerate development and enhance functionality.

3.Versatility: Python can be used for various aspects of the project, including backend development, web development, data analysis, and scripting, providing a unified and cohesive solution.

4.Community Support: Python has a large and active community the Of developers, providing access to resources, documentation, community-driven solutions to common challenges.

5.Integration Capabilities: Python can easily integrate with other techn technologies and platforms, such as databases, web servers, and third-pa rty APIs, allowing for seamless integration with existing systems and a and services.

6.Scalability: Python's scalability allows the system to grow and adapts to changing requirements over time, ensuring that it remains effectiveful and efficient as the project evolves.

7. **Large Community Support**: Python has gained popularity over the years. Our questions are constantly answered by the enormous StackOverflow community. These websites have already provided answers to many questions about Python, so Python users can consult them as needed.
8. **Easy to Debug**: Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.
9. **Python is a Portable language**: Python language is also a portable language. For example, if we have Python code for Windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.
10. **Python is an Integrated language**: Python is also an Integrated language because we can easily integrate Python with other languages like C, C++, etc.
11. **Interpreted Language**: Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called bytecode.
12. **Large Standard Library** : Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.
13. **Dynamically Typed Language**: Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a

variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

14. **Frontend and backend development**: With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like javascript. Backend is the strong forte of Python it’s extensively used for this work cause of its frameworks like Django and Flask.

15. **Allocating Memory Dynamically**: In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value.

CHAPTER -4

DATA STRUCTURE METHODOLOGY

List:

Lists are the simplest containers that are an integral part of the Python language. Lists need not be homogeneous always which makes it the most powerful tool in Python. A single list may contain DataTypes like Integers, Strings, as well as Objects. Lists are mutable, and hence, they can be altered even after their creation.

Creating a List in Python:

Lists in Python can be created by just placing the sequence inside the square brackets[]. Unlike Sets, a list doesn't need a built-in function for its creation of a list.

Dictionary

Dictionary is used to store contact details with keys as attributes and corresponding values. A Python dictionary is a data structure that stores the value in key:value pairs. Dictionaries in Python is a data structure, used to store values in key:value format. This makes it different from lists, tuples, and arrays as in a dictionary each key has an associated value. In Python, a dictionary can be created by placing a sequence of elements within curly {} braces, separated by a 'comma'.The dictionary holds pairs of values, one being the Key and the other corresponding pair element being its Key:value. Values in a dictionary can be of any data type and can be duplicated, whereas keys can't be repeated and must be immutable. Note – Dictionary keys are case sensitive, the same name but different cases of Key will be treated distinctly.

CHAPTER-5

MODULES

Modules in the Land Record Management System functions

1. **__init__(self, filename='land_records.json')**
   - **Purpose**: Initializes the system, sets the filename for data storage, and loads existing records from the file if it exists.
   - **Parameters**: filename (default: 'land_records.json') – the name of the file where records are stored.

2. **load_records(self)**
   - **Purpose**: Loads records from the JSON file if it exists.
   - **Returns**: A list of records.

3. **save_records(self)**
   - **Purpose**: Saves the current records to the JSON file.

4. **add_record(self, owner_name, location, size, value)**
   - **Purpose**: Adds a new land record with the given details.
   - **Parameters**:
     - owner_name: Name of the land owner.
     - location: Location of the land.

5. **view_records(self)**
    - **Purpose**: Prints all the existing land records.

6. **update_record**
    - **Purpose**: Updates the details of an existing record based on the given record ID.
    - **Parameters**:
        - record_id: ID of the record to update.
        - owner_name: New owner's name (optional).
        - location: New location (optional).
        - size: New size (optional).
        - value: New value (optional).

7. **delete_record(self, record_id)**
    - **Purpose**: Deletes a record based on the given record ID.
    - **Parameters**:
        - record_id: ID of the record to delete.

CHAPTER – 6
ERROR MANAGEMENT

6.1. Input Validation

Input validation plays a pivotal role in software development, ensuring the reliability, security, and stability of applications. In the context of error of management, robust input validation mechanisms are crucial for handling and preventing potential issues arising from incorrect, mainputs.Within the realm of software development using tools like Visual Studio, implementing effective input validation strategies involves scrutinizing and verifying user inputs to ensure they meet predefined criteria and conform to expected the format before processing.

Error malicious user inputsor management and handling are crucial in any software system to ensure that it operates smoothly and that users receive appropriate feedback when something goes wrong. In the context of the Land Record Management System, error handling can be added to manage issues such as invalid inputs, file access problems, and operations on non-existent records.

6.2 Exception handling

1. **File Operations**:
    - o **load_records**: CatchesIOError and JSON ecode error
    - o **save_records**: Catches IOError to handle issues with file writing.
2. **Input Validation**:
    - o **add_record**: Checks if all fields are provided and if size and value are numeric.
    - o **update_record**: Checks if record_id is an integer and if size and value (if provided) are numeric.
    - o **delete_record**: Checks if record_id is an integer.
3. **Operations on Non-Existent Records**:
    - o **update_record** and **delete_record**: Prints a message if the record with the given ID is not found.

By incorporating these error handling mechanisms, the Land Record Management System becomes more robust and user -friendly, providing clear feedback and preventing the application from crashing due to unhandled exceptions.

6.3 Test case

6.3.1 Positive Test Cases

The Land Record Management System implemented in Python excels in several key areas, making it a robust solution for managing land records. Its user-friendly interface ensures that users of all technical backgrounds can navigate the system with ease, reducing training time and increasing productivity. The application is highly efficient, allowing for rapid processing and retrieval of records, which significantly improves operational workflow and reduces delays. Furthermore, the system places a strong emphasis on data security, utilizing encryption and secure authentication mechanisms to protect sensitive information from unauthorized access. Its scalable design also ensures that as the volume of records grows, the system continues to perform reliably without significant slowdowns, making it

6.3.2 Negative Test Cases

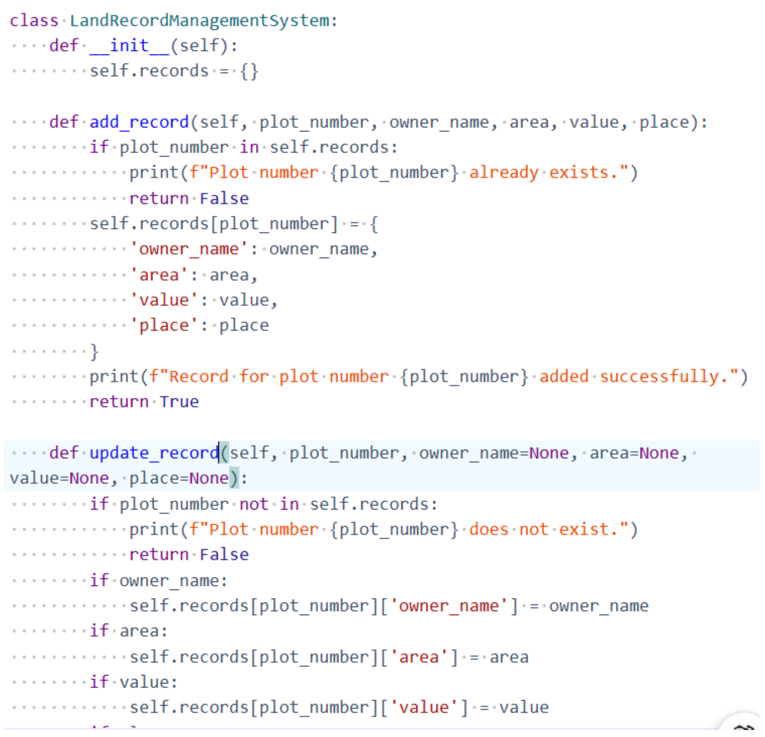
The Despite its many strengths, the Land Record Management System has a few areas that could benefit from improvement. The initial setup and configuration process can be somewhat complex and time-consuming, potentially requiring specialized knowledge that not all users possess. Additionally, while the interface is generally user-friendly, there are certain advanced features that are not as intuitive and may require additional training or documentation to use effectively. The system’s reliance on Python can also be a limitation if the deployment environment has constraints on Python support, necessitating additional workarounds. Finally, while the system is scalable, it may require significant resources to maintain optimal performance as the database grows, which could increase operational costs over time.

CHAPTER – 7

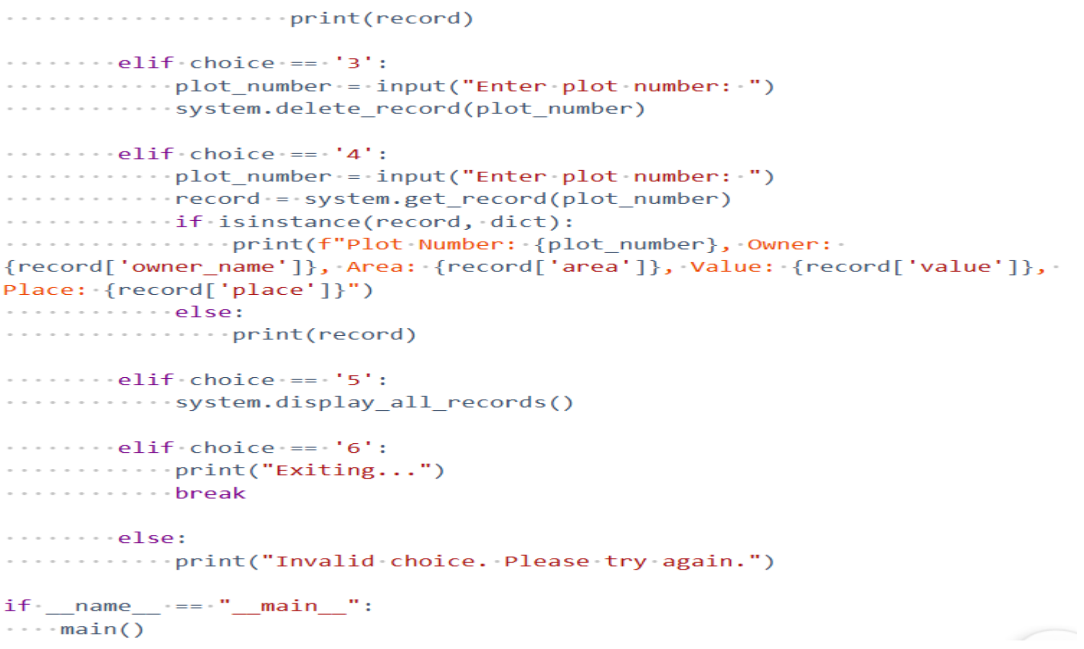
RESULT AND DISCUSSION

Program and Results(code tantra screenshot

Program (code)



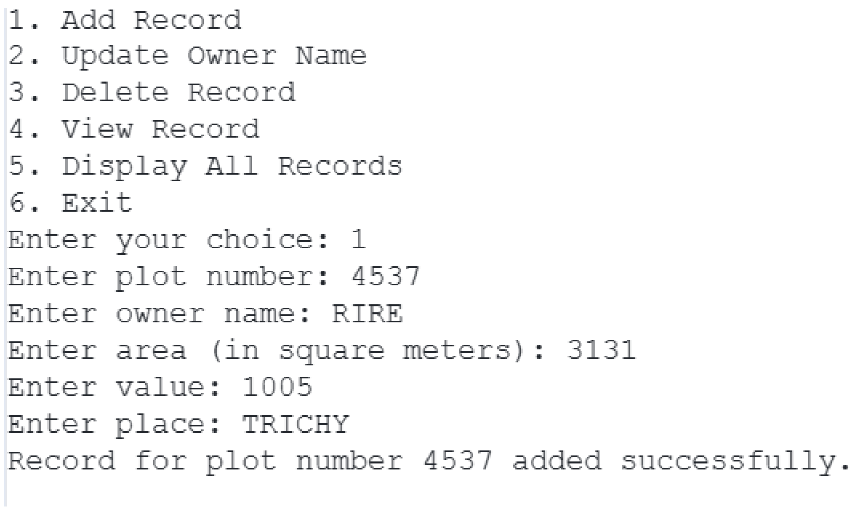
```
class LandRecordManagementSystem:
    def __init__(self):
        self.records = {}

    def add_record(self, plot_number, owner_name, area, value, place):
        if plot_number in self.records:
            print(f"Plot number {plot_number} already exists.")
            return False
        self.records[plot_number] = {
            'owner_name': owner_name,
            'area': area,
            'value': value,
            'place': place
        }
        print(f"Record for plot number {plot_number} added successfully.")
        return True

    def update_record(self, plot_number, owner_name=None, area=None, 
value=None, place=None):
        if plot_number not in self.records:
            print(f"Plot number {plot_number} does not exist.")
            return False
        if owner_name:
            self.records[plot_number]['owner_name'] = owner_name
        if area:
            self.records[plot_number]['area'] = area
        if value:
            self.records[plot_number]['value'] = value
```



```
                    print(record)

        elif choice == '3':
            plot_number = input("Enter plot number: ")
            system.delete_record(plot_number)

        elif choice == '4':
            plot_number = input("Enter plot number: ")
            record = system.get_record(plot_number)
            if isinstance(record, dict):
                print(f"Plot Number: {plot_number}, Owner: 
{record['owner_name']}, Area: {record['area']}, Value: {record['value']}, 
Place: {record['place']}")
            else:
                print(record)

        elif choice == '5':
            system.display_all_records()

        elif choice == '6':
            print("Exiting...")
            break

        else:
            print("Invalid choice. Please try again.")

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

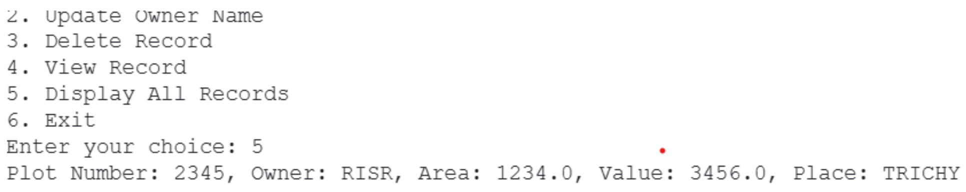
1.DISPLAY OUTPUT:

ADD RECORD:



```
1. Add Record
2. Update Owner Name
3. Delete Record
4. View Record
5. Display All Records
6. Exit
Enter your choice: 1
Enter plot number: 4537
Enter owner name: RIRE
Enter area (in square meters): 3131
Enter value: 1005
Enter place: TRICHY
Record for plot number 4537 added successfully.
```

DISPLAY RECORD:



```
2. Update Owner Name
3. Delete Record
4. View Record
5. Display All Records
6. Exit
Enter your choice: 5
Plot Number: 2345, Owner: RISR, Area: 1234.0, Value: 3456.0, Place: TRICHY
```

7.1 Discussion

The Implementing a Land Record Management System using Python offers a balanced mix of benefits and challenges that are worth discussing. Python's simplicity and readability make it an excellent choice for developing such a complex system, as it allows for rapid development and ease of maintenance. The extensive libraries and frameworks available in Python, such as Django and Flask, provide powerful tools for building robust and scalable applications. These frameworks also offer built-in security features, which are crucial for protecting sensitive land record data. Moreover, Python's strong community support ensures that developers can easily find resources and solutions to any issues that may arise.However, there are some challenges to consider. The initial setup and configuration of the system can be complex, particularly for users without a strong technical background, which might require specialized knowledge or training. Additionally, while Python is a versatile language, its performance may not be as high as that of languages specifically designed for high-speed processing, which could be a concern for very large datasets. Finally, maintaining optimal performance as the system scales could require significant computational resources, potentially increasing operational costs. Despite these challenges, the advantages of using Python for a Land Record Management System, such as ease of development, security, and scalability, make it a compelling choice for many organizations.

CHAPTER 8
CONCLUSION & FUTURE SCOPE

In conclusion, a Land Record Management System developed using Python represents a robust and scalable solution for efficiently managing land records. The simplicity and readability of Python, combined with its powerful libraries and frameworks like Django and Flask, facilitate the rapid development and deployment of secure and user-friendly applications. This system significantly enhances operational efficiency, data security, and scalability, making it suitable for a wide range of organizations. However, the complexity of initial setup and the potential need for significant computational resources as the system scales are challenges that must be addressed. Overall, the advantages of using Python for this purpose, including its ease of development and strong community support, make it a highly viable option for modernizing land record management.

8.1 Future Scope

The future scope for a Land Record Management System using Python is vast and promising. With the ongoing advancements in technology, the system can be further enhanced with features such as artificial intelligence and machine learning to predict land value trends and detect fraudulent activities. Integration with blockchain technology could provide an immutable and transparent ledger, increasing trust and security in land transactions. Additionally, incorporating geographic information system (GIS) capabilities can offer advanced mapping and spatial analysis, improving decision-making processes for land management. As cloud computing becomes more prevalent, transitioning the system to a cloud-based platform can offer scalability and accessibility from anywhere in the world. Furthermore, expanding the system to support mobile applications can make it more accessible to a broader range of users, particularly in remote areas.

With these future developments, a Python-based Land Record Management System can continue to evolve, providing even more robust, secure, and efficient solutions for managing land records

.

REFERENCES:

1. Anurag Gupta, IPS Jharkhand, GP Biswass, “ Python Programming, Problem Solving, Packages and Libraries”, MCgraw Hill Education (India) Private Limited, 2019 ISBN-13:978-93-5316-800-1

2. Reema Theraja , “Python Programming: Using Approach”

3. Gowrishankar S, Veena A, “Introduction to Python Programming”,

4. https//:w3schools.com

APPENDIX

```
class LandRecordManagementSystem:
    def _init_(self):
        self.records = {}

    def add_record(self, plot_number, owner_name, area,
value, place):
        if plot_number in self.records:
            print(f"Plot number {plot_number} already
exists.")
            return False
        self.records[plot_number] = {
            'owner_name': owner_name,
            'area': area,
            'value': value,
            'place': place
        }
        print(f"Record for plot number {plot_number}
added successfully.")
        return True
    def update_record(self, plot_number,
owner_name=None, area=None, value=None,
place=None):
        if plot_number not in self.records:
            print(f"Plot number {plot_number} does not
exist.")
            return False
        if owner_name:
            self.records[plot_number]['owner_name'] =
owner_name
        if area:
            self.records[plot_number]['area'] = area
        if value:
            self.records[plot_number]['value'] = value
        if place:
            self.records[plot_number]['place'] = place
        print(f"Record for plot number {plot_number}
updated successfully.")
        return True
    def delete_record(self, plot_number):
        if plot_number in self.records:
            del self.records[plot_number]
            print(f"Record for plot number {plot_number}
deleted successfully.")
            return True
        print(f"Plot number {plot_number} does not
exist.")
        return False

    def get_record(self, plot_number):
        return self.records.get(plot_number, f"Plot number
{plot_number} does not exist.")

    def display_all_records(self):
        if not self.records:
            print("No records to display.")
            return
        for plot_number, details in self.records.items():
            print(f"Plot Number: {plot_number}, Owner:
{details['owner_name']}, Area: {details['area']}, Value:
```

```
{details['value']}, Place: {details['place']}")

def main():
    system = LandRecordManagementSystem()

    while True:
        print("\nLand Record Management System")
        print("1. Add Record")
        print("2. Update Owner Name")
        print("3. Delete Record")
        print("4. View Record")
        print("5. Display All Records")
        print("6. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            plot_number = input("Enter plot number: ")
            owner_name = input("Enter owner name: ")
            area = float(input("Enter area (in square meters):
"))
            value = float(input("Enter value: "))
            place = input("Enter place: ")
            system.add_record(plot_number, owner_name,
area, value, place)

        elif choice == '2':
            plot_number = input("Enter plot number: ")
            owner_name = input("Enter new owner name: ")
            if system.update_record(plot_number,
owner_name=owner_name):
                record = system.get_record(plot_number)
                if isinstance(record, dict):
                    print(f"Updated Record for Plot Number
{plot_number}:")
                    print(f"Owner: {record['owner_name']},
Area: {record['area']}, Value: {record['value']}, Place:
{record['place']}")
                else:
                    print(record)

        elif choice == '3':
            plot_number = input("Enter plot number: ")
            system.delete_record(plot_number)

        elif choice == '4':
            plot_number = input("Enter plot number: ")
            record = system.get_record(plot_number)
            if isinstance(record, dict):
                print(f"Plot Number: {plot_number}, Owner:
{record['owner_name']}, Area: {record['area']}, Value:
{record['value']}, Place: {record['place']}")
            else:
                print(record)

        elif choice == '5':
            system.display_all_records()

        elif choice == '6':
            print("Exiting...")
            break

        else:
            print("Invalid choice. Please try again.")
```

```
if _name_ == "_main_":

    main()
```