Property Verification and Tax Payment

Ms. Anjali S, Assistant Professor

CHAPTER-1

1.1 Project Overview

INTRODUCTION

Architecture and execution property tax concerns are little researched. This is particularly true in developing countries, where the property market is largely unorganized and the valuations are therefore incredibly difficult, knowledge available to tax authorities is restricted, extent of decentralization is restricted. The vicious circle of low-service quality, bad tax enforcement, and inadequate local public services persists in most industrialised nations. As property tax is the most important means of local government income, its implementation is necessary to break the vicious cycle.

To validate land transactions and prevent fraud, an economic system needs precise and efficient land record management. Urbanization complicates land transactions. Urban land transactions are plagued by phoney paperwork, "benami" and fraudulent land transfers, unnaturally high demand for urban property, and inadequate governmental oversight of land transactions. Controlling these issues will reduce public insecurity. Most Indian registrations do not provide genuine ownership titles. Thus, 70% of legal issues involve land. a computerised urban land verification system that verifies property titles.

The app can be utilized by municipal corporations, local governments, or city administrations to provide a convenient platform for property owners within their jurisdiction to pay property taxes online. It allows property owners to easily access and manage their tax payments, view tax details, receive notifications, and make payments securely.

Objectives of the app are as follows:

- Start an accurate property survey.
- Give each property a permanent number.
- Issue urban household titles.

CHAPTER-2

2.1 Existing System

LITERATURE SURVEY

ULBs use "Khata" to track property tax information. "Khata" is a property ownership document. ULBs collect taxes from urban residents using "Khatas," which are not Records of Rights. Current property verification and tax payment is manual.

Drawbacks

- Property owner has to visit the government centers for payment of property tax
- Due to human intervention cheating can be done by preparing fake documents.
- New Construction and development in real estate business may decrease

- Less property tax collection due to bribe
- Multiple owners for single property
- Increase of litigation property

2.2 Proposed System

The proposed android mobile app maintains property ownership data to promote liquidity and transactions and allow property owners to easily monetise their assets. Urban property record systems save land litigation costs and time.

Advantages

- Security for Property
- Protection from unauthorized acquisition
- Exact property details
- Facilitates seeking of legal assistance from courts in case of disputes
- Enables in availing bank loans
- Facilitates access to water and electricity services
- Helps in obtaining building plan approval
- Allotment of a permanent and specific number to each individual property prevents the occurrence of frauds in the process of buying and leasing of properties.

2.3 Feasibility Study

A crucial stage that must be completed An evaluation of the viability of a certain project or endeavour is doable and has the potential to be lucrative is the feasibility analysis. The project's technical, economic, legal, operational, and scheduling aspects are examined to see whether it's worth continuing.

1. Economic Feasibility

Economic feasibility addresses project finance. This project is economically viable since its financial advantages and The cost is less than those of the current system. Open-source software reduces hardware and software costs. Android smartphones and Windows 7+ laptops are needed.

2. Technical Feasibility

Technical Feasibility is all about the execution of the project and the set of Software and Toolsincluded in the project.

Android Studio develops this project. The project's front end is XML and its back end is Core Java. Since all development tools and technologies are open source, the project is theoreticallypossible without large expenses.

Mobile app development: The technical expertise and resources required for developing an Android application are readily available. The app can leverage existing frameworks and libraries to streamline development.

3. Operational Feasibility

Operational feasibility focuses on whether the system will be utilised after development and implementation. Operational Feasibility ensures application functionality.

End customers will like the new method since the app saves time and effort. Tools and Technologies

1. Java Programming

Java, a flexible programming language, allows developers to create code once and reuse it countless times. This saves developers time and effort by allowing them to use existing code for tasks beyond their expertise. Reusing code allows developers to easily complete complicated tasks without reinventing the wheel.
The emergence of the Android platform predates the rich history of the Java programming language. The object-oriented programming paradigm, which emphasises the organisation and manipulation of data through objects and classes, has been widely used and recognised. Modular and maintainable code is promoted by this approach, making it easier for software projects to be designed, implemented, and managed by developers.
The need for a platform-independent language that could be employed to write code for various consumer electronic devices was one of the primary motivations behind the creation of Java. Developers were allowed to write code once and deploy it on different hardware and operating system configurations because the language was designed to be architecture-neutral. Software development was revolutionised by this platform independence, as it enabled developers to target multiple platforms without the need for extensive modifications or rewriting of code.
Java's adaptability and platform freedom have helped it gain popularity. Its stability, scalability, and comprehensive library support make it perfect for corporate systems, web development, and Android app development. The Java community and documentation help developers learn, collaborate, and share code.

2. XML

Markup specifies documents. XML is used to create structured information documents (Extensible Markup Language). HTML displays web data whereas XML defines it (Hypertext Markup Language). XML complements HTML, not replacing it. XML lets authors customise document markups and data representation. HTML tags mark up web pages and organise HTML documents. Unlike HTML, XML has no tag sets or inherent meanings. This allows writers to customise tags and structural connections. XML

allows bespoke and domain-specific document formats, enabling data transmission across systems and applications.

3. Android Operating System

Android, based on Linux, was created for smartphones and tablets. Google led its collaborative development via the Open Handset Alliance.It is found crucial by programmers to design applications for the Android platform because a standardised and consistent approach to managing mobile app development is offered by it. A diverse range of Android-powered devices can have their applications created by developers to work seamlessly, ensuring a wider reach and user base.
Google released the Android SDK beta to developers in 2007, followed by Android 1.0 in September of the same year. Android becomes a comprehensive mobile operating system. Because Android is open-source, developers may access its source code. This open approach lets Android developers customise, innovate, and collaborate. Open source licencing has generated a healthy developer community, helping the Android platform grow and improve. Android's open-source nature encourages third-party app and modification creation, adding to its popularity and adaptability. Developers may design custom Android experiences, resulting in a wide diversity of applications and features on the platform.

4. Android Studio

Google created Android Studio, the official Android app development IDE. It enables Android developers to build several apps. Android Studio's sophisticated tools simplify development. Its user-friendly interface simplifies coding, debugging, and project administration. With auto-completion and code restructuring, its powerful code editor helps developers create and modify code effectively. The IDE's strong debugger helps developers find and repair bugs. The Android SDK is completely integrated with Android Studio, giving developers access to a huge library of libraries, APIs, and resources. Developers may create feature-rich Android apps with this connection. Android Studio runs code on emulators and physical devices. Developers may test and debug their apps in numerous situations and settings to optimise performance and user experience across devices. Android Studio offers tutorials, documentation, and a vibrant development community. These tools help app developers learn, cooperate, and solve problems.

5. SQLITE Database Service

SQLite is a popular relational database management system. While client server database engines operates the aforementioned databases, SQLite differs in that a self-contained, serverless database engine is operated by it. Despite a significant portion of the SQL standard being adhered to by SQLite, the features of a relational database can still be leveraged by developers. A SQLite database can be incorporated within your software to utilise the capabilities of a relational database management system. Its ease of use and portability are notable advantages of SQLite. Developers are allowed to seamlessly integrate SQLite databases into their applications as it is included as part of the complete software package. Moreover, both Android and iOS platforms have preinstalled SQLite, making it readily accessible for developers targeting these mobile operating systems. Various types of data, including device contacts and media files, are extensively stored and retrieved using SQLite in the context of Android. These datasets are managed reliably and efficiently within the android ecosystem by it.

CHAPTER -3

SOFTWARE REQUIREMENT SPECIFICATION

Software system requirements—both functional and non-functional—are gathered and documented in the Software Requirement Specifications (SRS) document. All stakeholders inthe software development process—clients, developers, and testers—must communicate regularly to ensure project success.

A consensus is established on the capabilities, constraints, and overall objectives of the software programme by the SRS document. The document can be utilised effectively due to this consensus. A legally enforceable agreement is created by the SRS document, including all parties, such as the client or customer and the development team. The specific requirements and objectives that must be met for the software development process towards accomplishing success are understood by everyone involved, thanks to this agreement. This requires informing all stakeholders about the criteria and goals

The Functional requirements list the software system's characteristics and capabilities (SRS). Whatthe programme should do is described by these requirements. Various techniques such as providing examples, working through scenarios, and sharing user stories are often employed to effectively explain these requirements. Different contexts can demonstrate how the system should respond optimally by applying these flexible approaches. Well-defined functional requirements have essential attributes of clarity, testability, and specificity. Among these, the utmost importance is held by clarity.

3.1 Functional Requirements

Functional requirements govern how a software system behaves, features, and works. Functional requirements describe the software's capabilities and responses to different inputs and contexts. Specific information regarding the interactions between users and the system along with the anticipated results and responses from the system, is offered by these requirements. These criteria detail user-system interactions and system responses.

Admin

This system needs admin. The admin may access the app using a predefined password and user name. The admin may add prisons and jailors to app after logging in. He may allocate jailors to prisons.

- Add city
- Add area
- Add inspector
- Assign area to inspector

Property Owner

The Property owner module manages a particular Property details. The administrator adds him to the app.. He is responsible for adding details of all the property details, After id generated by inspection owner he can view the property details and tax details.

- Register
- Add property details
- Property id generated by inspection Owner
- View Property details
- Tax Details
- Pay tax
- Receipt generated

Area Inspector

Area inspector can inspect the property which are uploaded by property owners. Inspector can approve/reject with the comments.

- Inspect Property
- Approve/Reject with comments
- Generate Property id

3.2 Non-Functional Requirements

These requirements define system operating standards rather than particular actions. These are system restrictions. Non-functional requirements describe system performance.

- **Performance:** An estimate for the reaction times, throughput, and resource utilisation that are expected to be associated with a particular system be arrived at by using the performance requirements for that system. The pace at which users interact with the system, the system's efficiency in processing information, network connection time,and the system's capacity to handle more work must be considered.

- **Reliability:** The app should should operate properly. The gauge output should never show inaccurate or obsolete information without warning the user. The software minimises data entry mistakes. Invalid data generates suitable error messages.

- **Usability:** The software must be straightforward and quick to use. App states should change rapidly.

- **Maintainability:** Portability Write software plainly and simply. Well-documented code. Modular software design will simplify maintaience. This programme will support Android 4.0 or above. This app will support all Android OS versions.

- **Availability:** If the Android smartphone works, the app will always be accessible criteria.
- **Security:** "Requirements" are the protections and measures needed to secure a system and data against future unauthorised access, breaches, or malicious activity. These steps ensure system and data integrity. This paper discusses security standards, authentication, authorisation, encryption, and more. Topics include encryption methods.

- **Response Time:** the system must finish user tasks quickly. The programme must do the operation immediately.

3.3 Hardware and Software Requirements

Software Requirements Operating System: Windows 10 Frontend: XML
Backend: Java Database: SQlite

Hardware Requirements Processor: Intel Core i5 or higher RAM: 8GB or higher
Storage: 10GB or higher

CHAPTER-4

4.1 Introduction

SYSTEM DESIGN

Software system design involves creating its behaviour, structure, components, and relationships. It covers everything and throughout the software development lifecycle. Based on requirements, the system design phase creates a detailed strategy for development and execution. System design focuses on designing a solution that meets functionality, performance, reliability, and scalability requirements. Designing the software system's structure, linkages, and data flows. The design phase creates a software system blueprint to guide development.

4.2 Architecture Design

Software system architecture is designed during architectural design. Its links, information flow, and control mechanisms are defined by many stages.

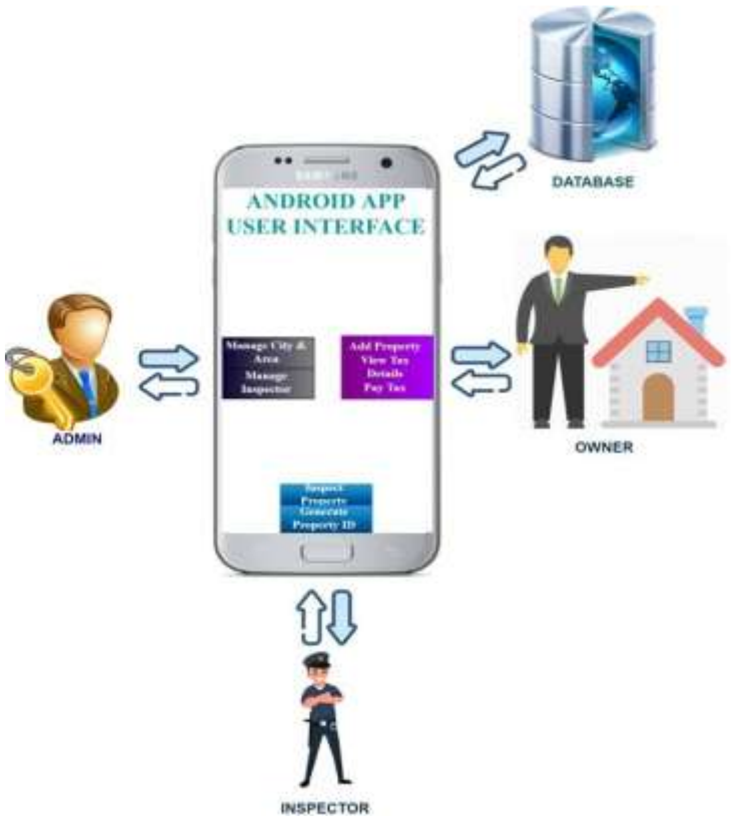


Fig 4.2.0: Architecture Design

CHAPTER-05

DETAILED DESIGN

5.1 Data Flow Diagram

DFD show information flow for any process or system. It shows data inputs, outputs, storage spots, and paths between destinations using rectangles, circles, arrows, and brief text labels. Data flowcharts may vary from basic, hand-drawn process overviews to multi-level DFDs that go deeper into data handling. They can examine and model systems. DFDs, like thegreatest diagrams and charts, can graphically "express" things that are hard to describe in wordsand work for both technical and nontechnical audiences, from developers to CEOs.

DFD rules and tips

- Processes need inputs and outputs
- Data flows should enter and leave each data repository
- A system processes data
- All DFD processes move to another process or data

DFD levels

Levels and layers help focus a data flow diagram on a specific item. DFD levels are 0, 1, 2, and sometimes even 3. Your goal decides how much information you need.

Context Diagrams are DFD Level 0. A fundamental overview of the system or process being investigated or modelled. It shows the system as a single high-level process with its external entities at a quick glimpse. Developers, stakeholders, business analysts, and data analysts should understand it.

DFD Level 1 details Context Level Diagram elements. As you deconstruct the Context Diagram's high-level process, you'll highlight the system's major functions.

DFD Level 2 then delves farther into Level 1. To explain the mechanism, extra text may be needed.

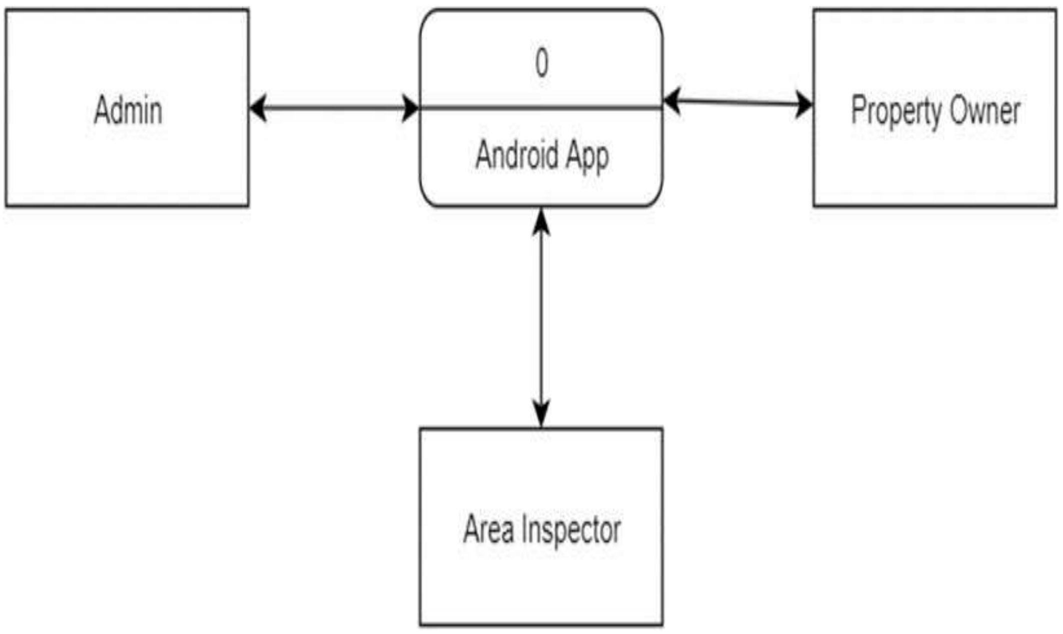


Fig 5.1.0: App Overview

Above figure shows the app overview of data flow diagram (DFD), it contains three blocks they are Admin, Property Owner and Area Inspector.

Admin helps to login to Users and add Area, city and Area Inspector.

Property owner helps to register or login to the application then owner will upload the details of the property after adding inspector inspect the property if all details are correct generate the id then owner can easily view the details of tax and pay the tax.

Area Inspector verify the property and provide clearance if property is legitimate. Property Owner register the property with all necessary property details.

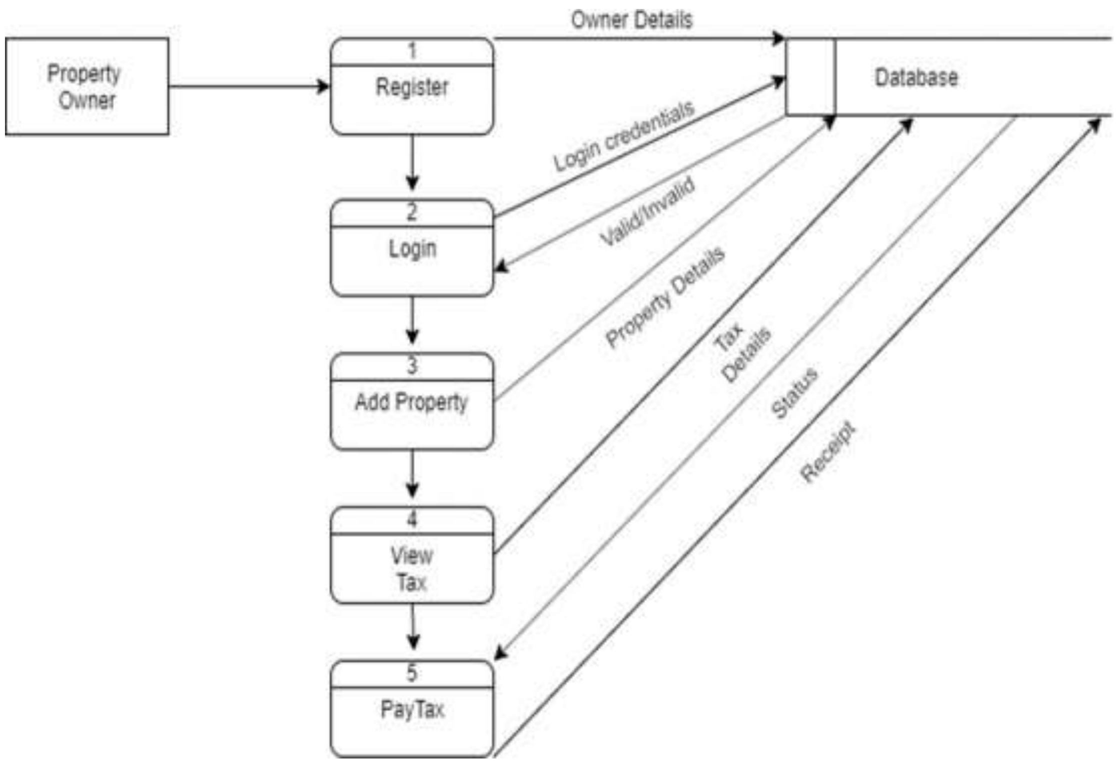


Fig 5.1.1: Property Owner

Property Verification: This is the system's primary feature, where the identity of the property owner and specifics of the property are confirmed.

Android App: The Android app provides users with the ability to utilize the online platform for properties verification and remittance of taxes.

User Login and Registration: To gain entry to the system and its features, users can log in using their credentials or register as new users.

Authenticated users can enter their own property facts, including the address, ownership details, and pertinent tax data.

Tax Payment: Via the app, users can access their tax information and submit payments. Property Data: This component maintains and saves data pertaining to properties, such as tax records and land specifics.

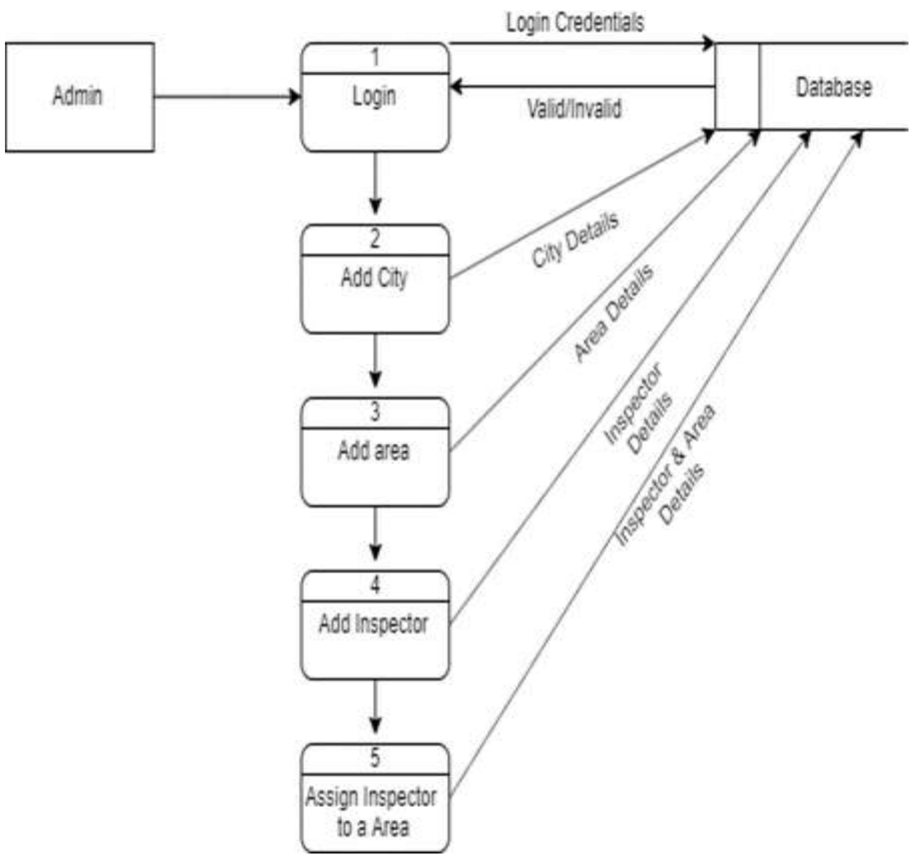


Fig 5.1.2: Admin DFD

Administrator: This term refers to the administrative position in charge of the system for verifying ownership and paying taxes. The administrator has the power to add cities, areas, inspectors, and allocate officers to certain regions.

The system user can add additional cities using this component, "Add City."

Add Area: The administrator can add additional areas to a city through this component.

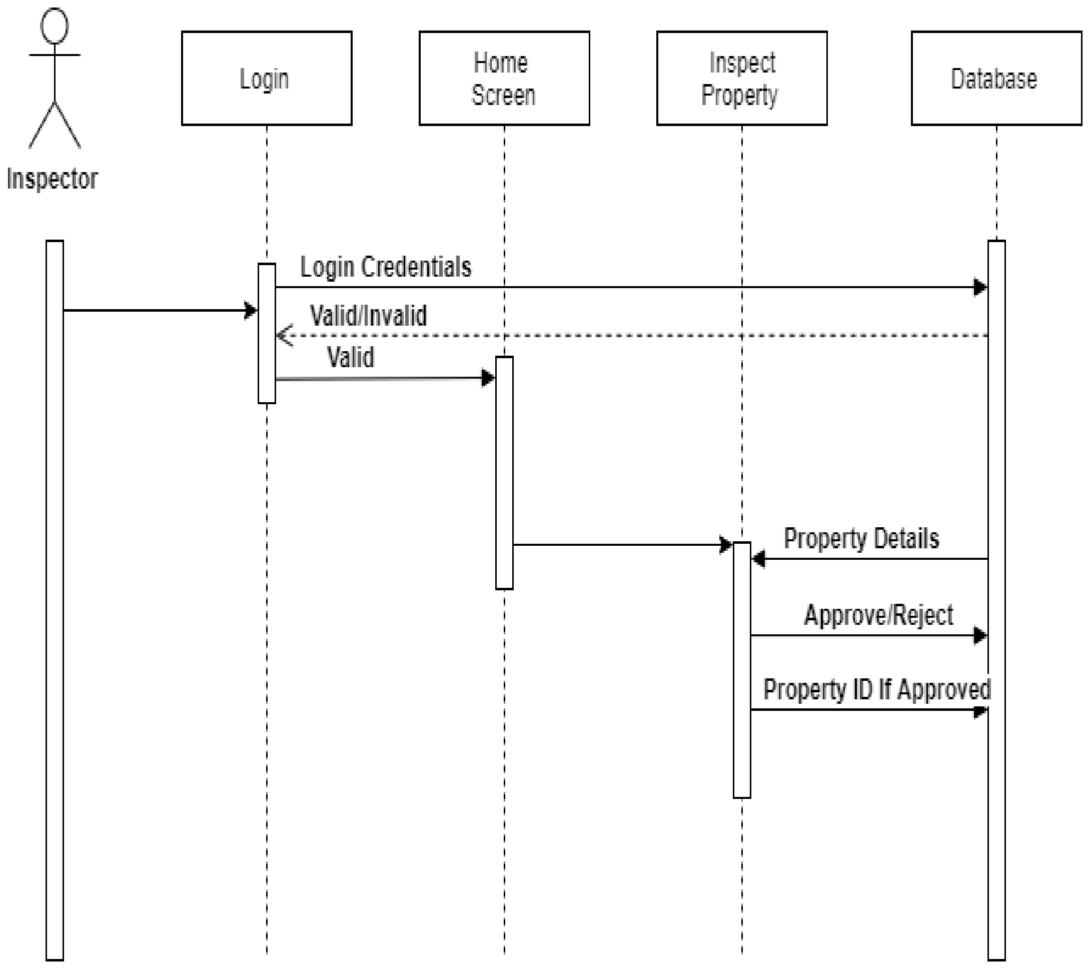
Inspectors can be selected by the an administrator, and their duties will be in charge of checking properties in particular regions.

Assign Area: The administrator can assign a region to an inspector, indicating that inspectors as the one in charge of inspecting the properties in that region.

5.2 Sequence Diagram

It illustrate object interactions in order. Sequence diagrams demonstrate time-ordered object interactions. It shows the scenario's objects and classes and the messages they exchange to work. The Logical the system's perspective under development links sequence diagrams with use case realisations. Event scenarios are sequence diagrams. Sequence diagrams help visualise and verify runtime situations. These may forecast system behaviour and reveal class responsibilities while modelling a new system.

Fig5.2.0: Sequence Diagram For Inspector



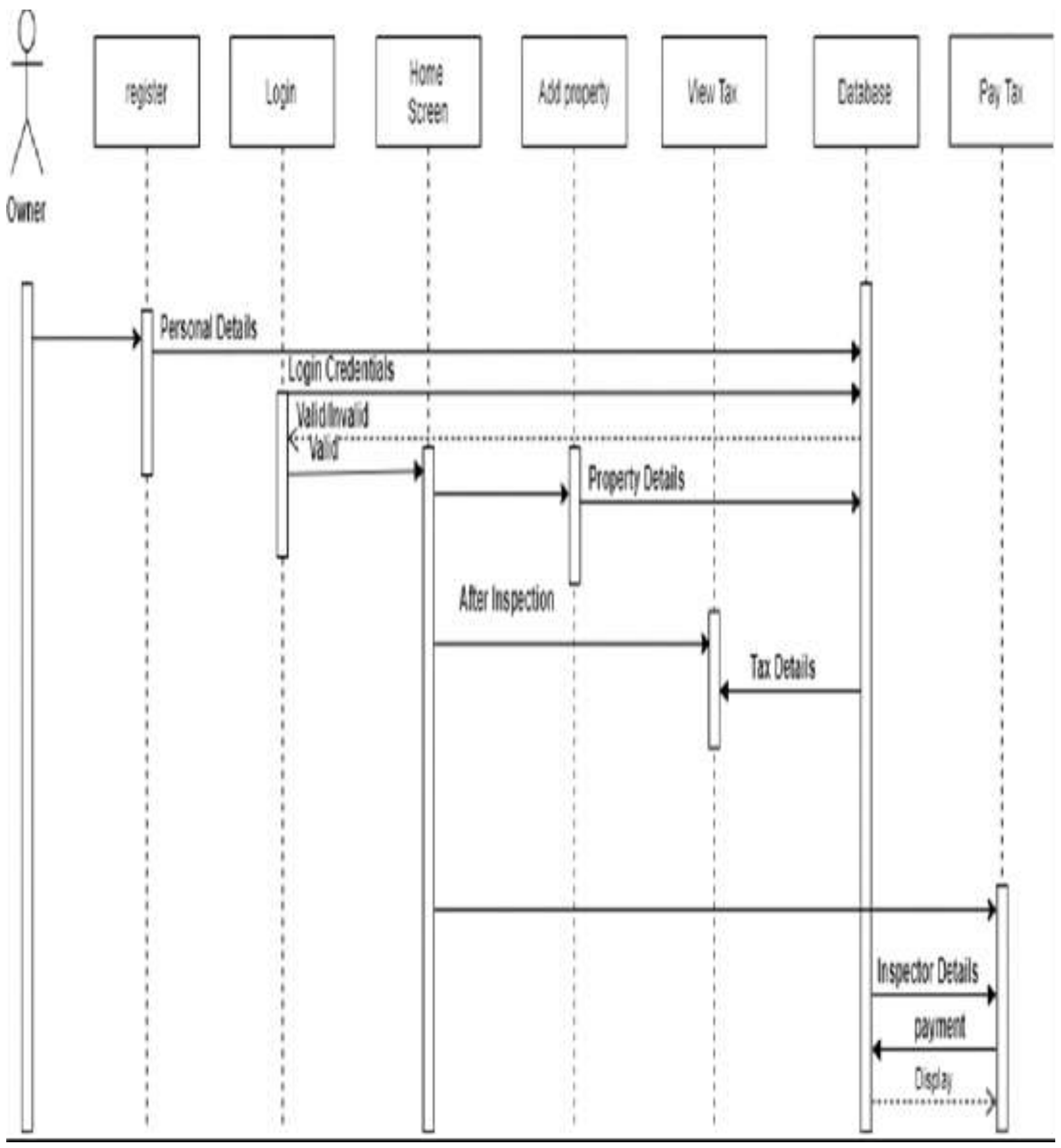


Fig5.2.1: Sequence Diagram For Property Owner

5.3 ER Diagram

Entity Relationship (ER) Diagrams, are flowcharts that show how "entities" like people, things, and ideas interact in a system. Software engineering, corporate information systems, education, and research typically employ ER Diagrams to develop or troubleshoot relational databases. ERDs, also known as ER Models, employ rectangles, diamonds, ovals, and connecting lines to show entities, relationships, and their properties. Relationships are verbs while entities are nouns.

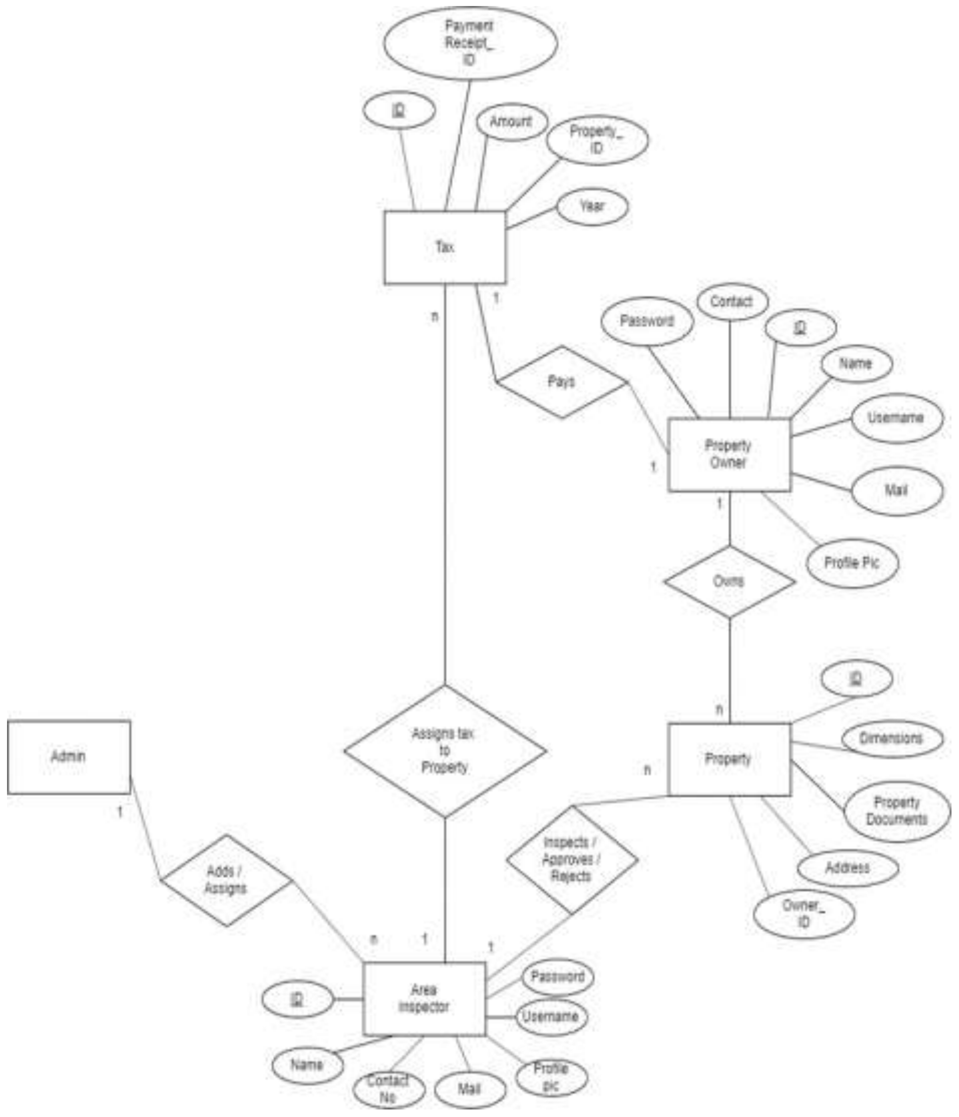


Fig 5.3.0: ER Diagram

5.4 Use Case Diagram

Use case diagrams in UML are based on behavioral representations. on use-case analysis. It provides a visual representation of a system's actors, use cases, and dependencies. Use case diagrams indicate which actors execute system functions. System actors may be portrayed.

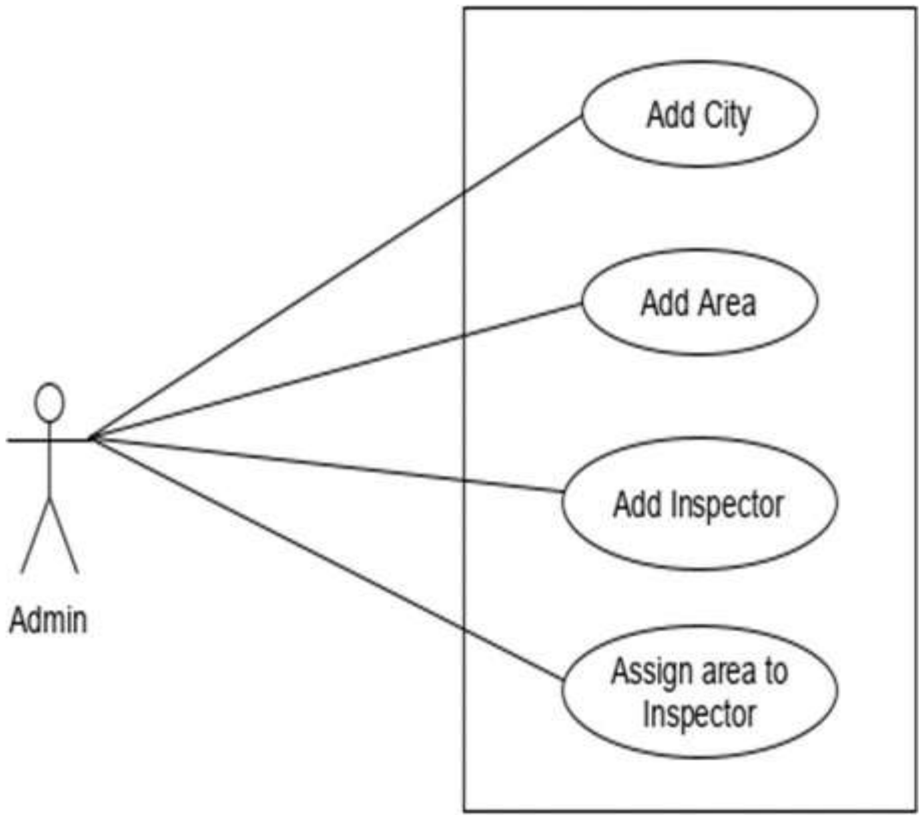


Fig 5.4.0: Administration

Above figure Shows the Admin activity, Admin helps to Add the all details of user to add City, Area and Inspector and also Admin helps to assign the area to Inspector to check the property information.

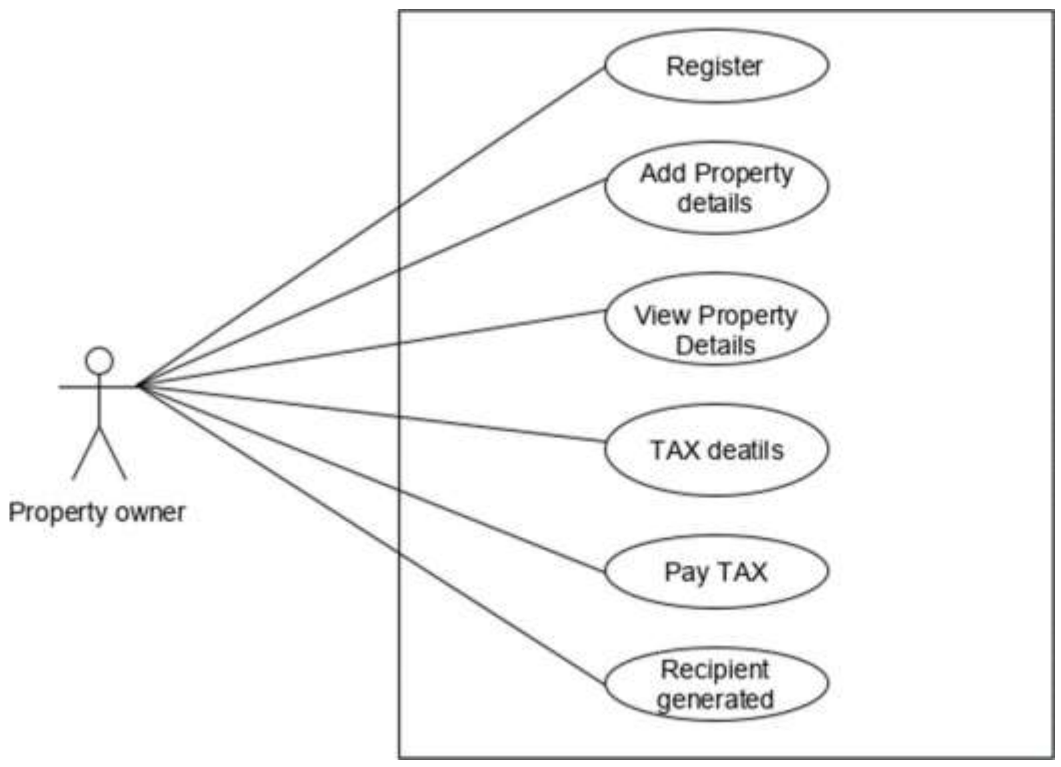


Fig 5.4.1: property owner exclusivity

Sign in : For using the capabilities for property verification and making tax payments, the property's holder must sign in to the app. This use case is a requirement for additional uses, hence it is included in the "Property Confirmation and Tax Payment" use case.

Register Account: The Property Owner has the option of opening a new account even though they don't yet have one.

Add Property History: The property owner may include details regarding their property, including the address, ownership information, and any additional relevant information needed for verification.

Add Tax data: The property's Owner may add tax-related information required for paying tax after providing property data.

Pay Tax: Using the app, the property owner is able to view their tax information and pay their property taxes.

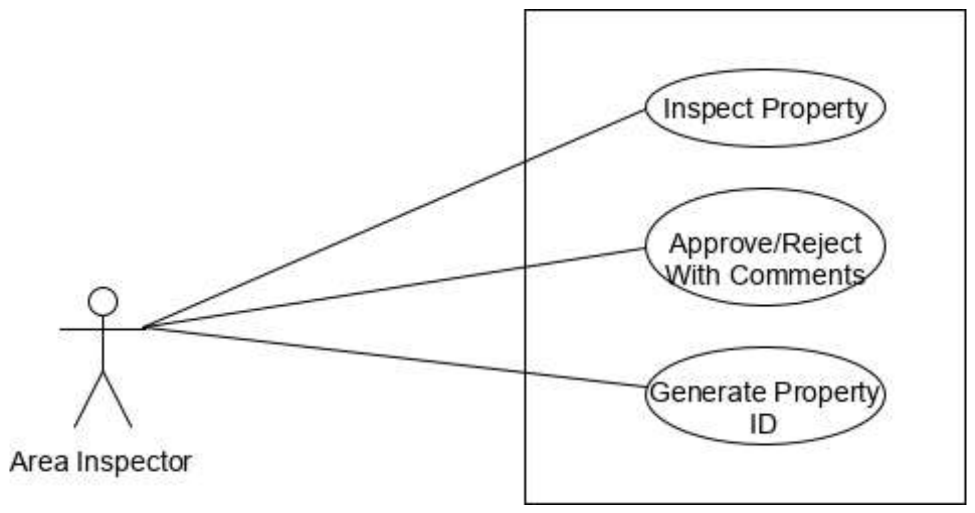


Fig 5.4.2: Area inspector confidentiality

Logging in as Inspector (include): In order to perform the features they offer, the Area Inspect needs to log into the program using their account information. This use case is a requirement for additional purposes.

Property Evaluation: The Local Controller can do a property inspection by going to the site and examining the property's specifications and regulatory compliance.

Property approval: If a property passes all review conditions and has been confirmed, the Area Inspector might authorize it.

Reject Property: The Area Inspector has the authority to disapprove of a property if it fails to comply to the required specifications or doesn't have the necessary paperwork.

Create Inspection Report: For every single property they control, the Area Inspector has the opportunity to create an extensive assessment report that includes details on the property's condition and any related findings or notes.

5.5 Activity Diagram

Activity maps illustrate system behavior. Activity diagrams outline the control flow, including decision paths, from beginning to end.

Activity diagrams show sequential procedures including choice, iteration, and concurrency. Activity diagrams in the Unified Modeling Language may explain system components business and operational processes. Activity diagrams demonstrate control flow.

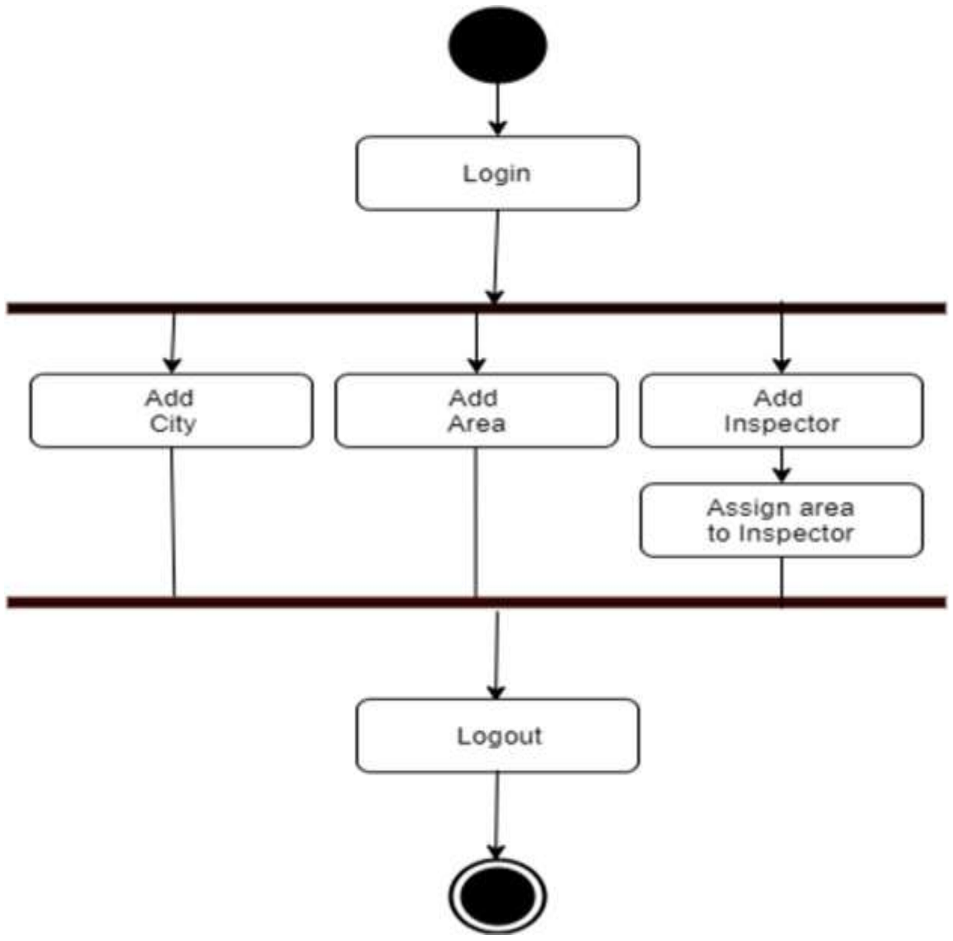


Fig 5.5.0: Admin Option

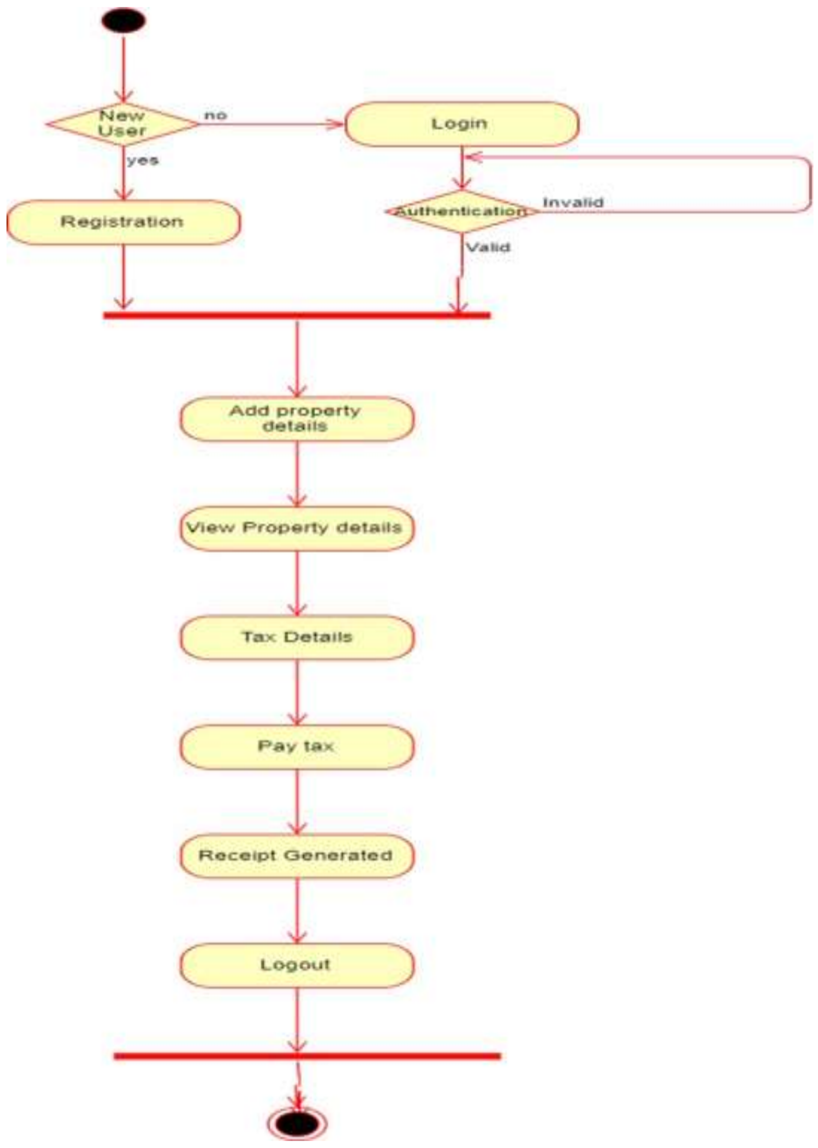


Fig 5.5.1: Customer Option

The above figure show the Customers options when the new user visit the application they need to register or else they need to login and if the criteria match user will get access to add property and view the details and get an access to pay the tax, when the tax payed successfully user will get an appropriate receipt .

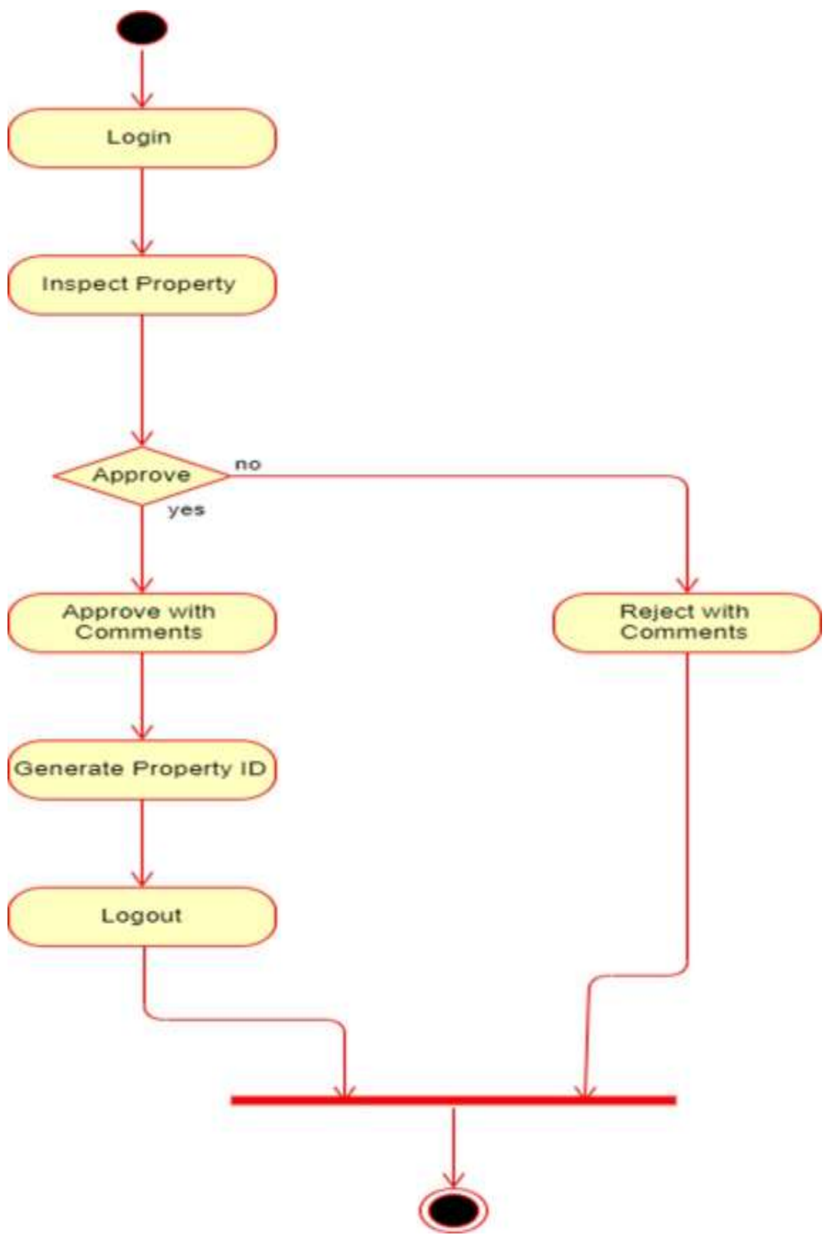


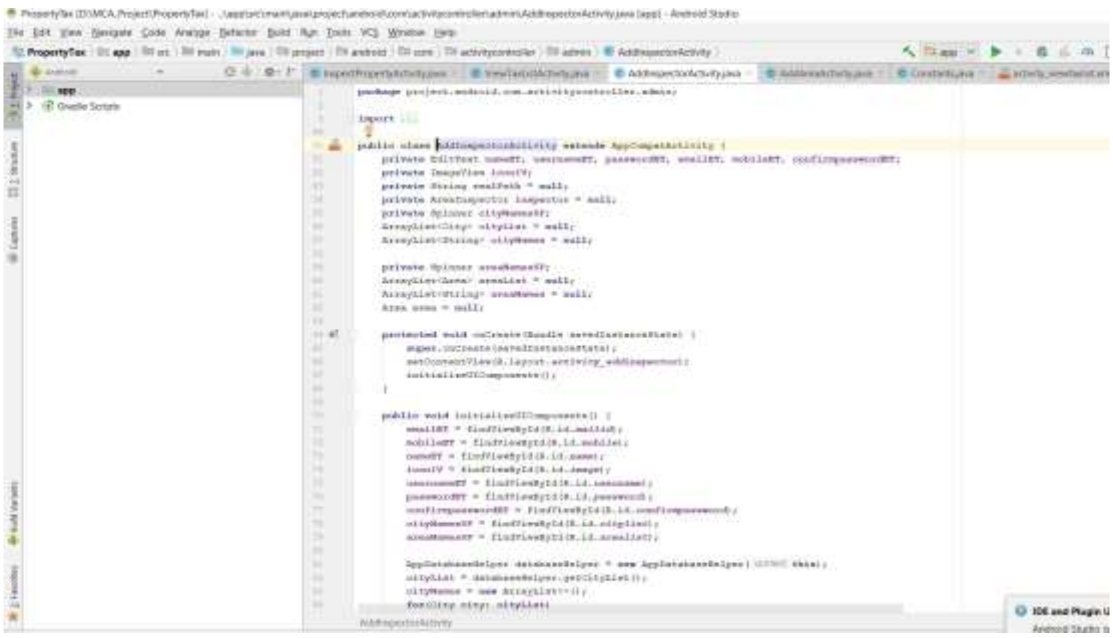
Fig 5.5.2: Activity Diagram – Inspector

The above diagram represent the Inspector options the inspector will Approve or Reject the property based user details and criteria , if the property is approved the user will get an unique ID and submit to user.

CHAPTER-6

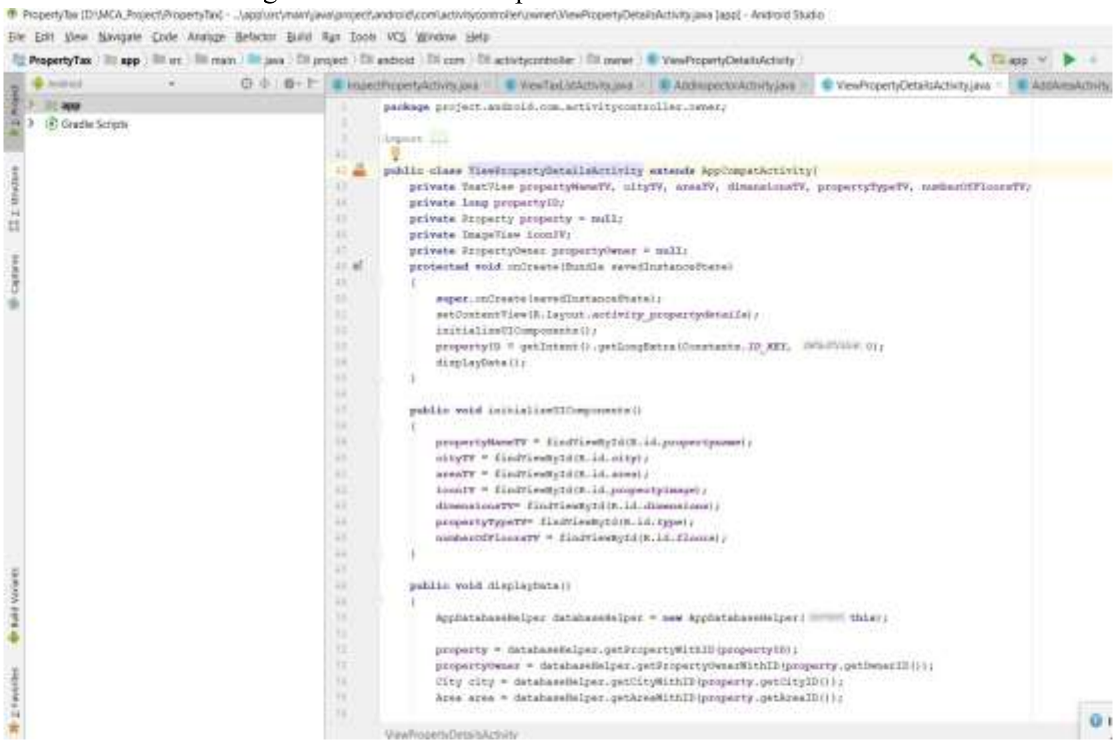
6.1 Code Snippets

IMPLEMENTATION



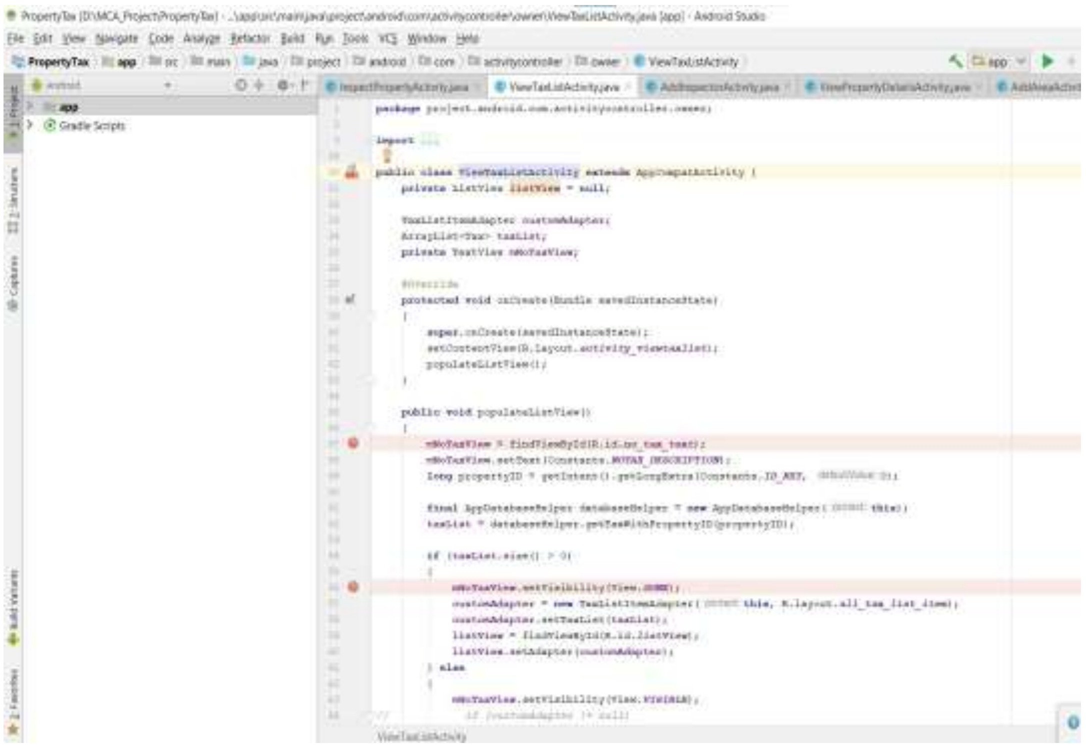
```java
package project.android.com.activitycontroller.admin;

import ...

public class AddInspectorActivity extends AppCompatActivity {
    private EditText nameET, usernameET, passwordET, emailET, mobileET, confirmpasswordET;
    private ImageView iconIV;
    private String realPath = null;
    private AreaInspector inspector = null;
    private Spinner cityNamesSP;
    ArrayList<City> cityList = null;
    ArrayList<String> cityNames = null;

    private Spinner areaNamesSP;
    ArrayList<Area> areaList = null;
    ArrayList<String> areaNames = null;
    Area area = null;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_addinspector);
        initializeUIComponents();
    }

    public void initializeUIComponents() {
        emailET = findViewById(R.id.emailid);
        mobileET = findViewById(R.id.mobileno);
        nameET = findViewById(R.id.name);
        iconIV = findViewById(R.id.image);
        usernameET = findViewById(R.id.username);
        passwordET = findViewById(R.id.password);
        confirmpasswordET = findViewById(R.id.confirmpassword);
        cityNamesSP = findViewById(R.id.citylist);
        areaNamesSP = findViewById(R.id.arealist);

        AppDatabaseHelper databaseHelper = new AppDatabaseHelper( context: this);
        cityList = databaseHelper.getCityList();
        cityNames = new ArrayList<>();
        for(City city: cityList)
```

Fig 6.1.0: Add Area Inspector



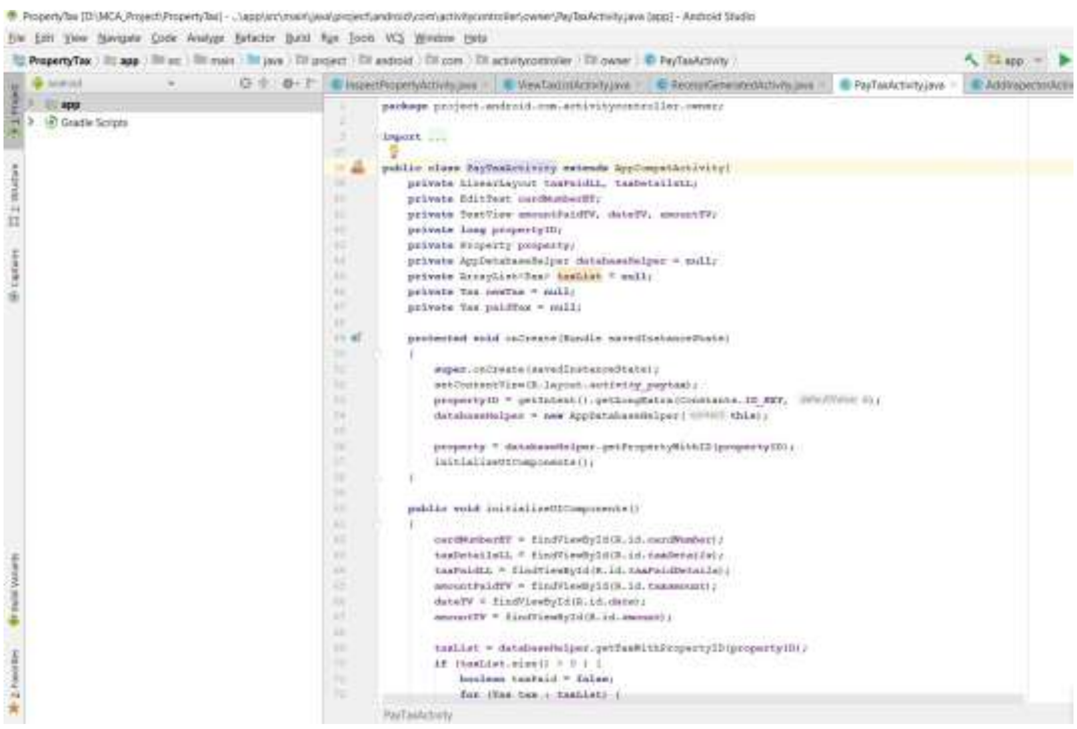
```java
package project.android.com.activitycontroller.owner;

import ...

public class ViewPropertyDetailsActivity extends AppCompatActivity{
    private TextView propertyNameTV, cityTV, areaTV, dimensionsTV, propertyTypeTV, numberOfFloorsTV;
    private long propertyID;
    private Property property = null;
    private ImageView iconIV;
    private PropertyOwner propertyOwner = null;
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_propertydetails);
        initializeUIComponents();
        propertyID = getIntent().getLongExtra(Constants.ID_KEY, defaultValue: 0);
        displayData();
    }

    public void initializeUIComponents()
    {
        propertyNameTV = findViewById(R.id.propertyname);
        cityTV = findViewById(R.id.city);
        areaTV = findViewById(R.id.area);
        iconIV = findViewById(R.id.propertyimage);
        dimensionsTV= findViewById(R.id.dimensions);
        propertyTypeTV= findViewById(R.id.type);
        numberOfFloorsTV = findViewById(R.id.floors);
    }

    public void displayData()
    {
        AppDatabaseHelper databaseHelper = new AppDatabaseHelper( context: this);

        property = databaseHelper.getPropertyWithID(propertyID);
        propertyOwner = databaseHelper.getPropertyOwnerWithID(property.getOwnerID());
        City city = databaseHelper.getCityWithID(property.getCityID());
        Area area = databaseHelper.getAreaWithID(property.getAreaID());
```

Fig 6.1.1: Property details


PropertyTax [D:\MCA_Project\PropertyTax] - ...\app\src\main\java\project\android\com\activitycontroller\owner\ViewTaxListActivity.java [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PropertyTax app src main java project android com activitycontroller owner ViewTaxListActivity
InspectPropertyActivity.java
ViewTaxListActivity.java
AddInspectionActivity.java
ViewPropertyDetailsActivity.java
app
Gradle Scripts
ViewTaxListActivity

```
package project.android.com.activitycontroller.owner;

import ...

public class ViewTaxListActivity extends AppCompatActivity {
    private ListView listView = null;

    TaxListItemAdapter customAdapter;
    ArrayList<Tax> taxList;
    private TextView mNoTaxView;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_viewtaxlist);
        populateListView();
    }

    public void populateListView()
    {
        mNoTaxView = findViewById(R.id.no_tax_text);
        mNoTaxView.setText(Constants.NOTAX_DESCRIPTION);
        long propertyID = getIntent().getLongExtra(Constants.ID_KEY, defaultValue: 0);

        final AppDatabaseHelper databaseHelper = new AppDatabaseHelper( context: this);
        taxList = databaseHelper.getTaxWithPropertyID(propertyID);

        if (taxList.size() > 0)
        {
            mNoTaxView.setVisibility(View.GONE);
            customAdapter = new TaxListItemAdapter( context: this, R.layout.all_tax_list_item);
            customAdapter.setTaxList(taxList);
            listView = findViewById(R.id.listView);
            listView.setAdapter(customAdapter);
        } else
        {
            mNoTaxView.setVisibility(View.VISIBLE);
//            if (customAdapter != null)
```

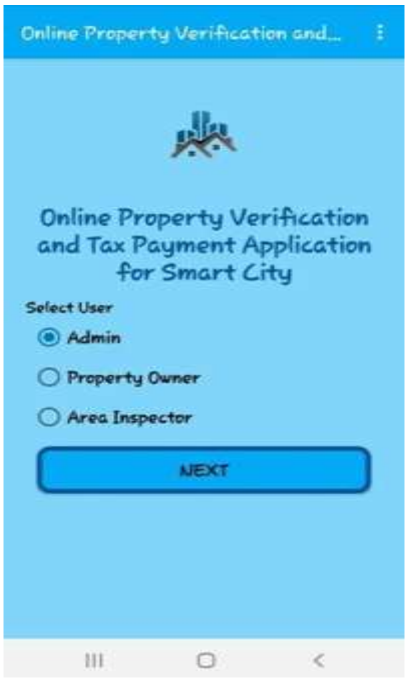
Fig 6.1.2: Tax details


PropertyTax [D:\MCA_Project\PropertyTax] - ...\app\src\main\java\project\android\com\activitycontroller\owner\PayTaxActivity.java [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PropertyTax app src main java project android com activitycontroller owner PayTaxActivity
InspectPropertyActivity.java
ViewTaxListActivity.java
ReceiptGeneratedActivity.java
PayTaxActivity.java
AddInspectionActivity
app
Gradle Scripts
PayTaxActivity

```
package project.android.com.activitycontroller.owner;

import ...

public class PayTaxActivity extends AppCompatActivity{
    private LinearLayout taxPaidLL, taxDetailsLL;
    private EditText cardNumberET;
    private TextView amountPaidTV, dateTV, amountTV;
    private long propertyID;
    private Property property;
    private AppDatabaseHelper databaseHelper = null;
    private ArrayList<Tax> taxList = null;
    private Tax newTax = null;
    private Tax paidTax = null;

    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_paytax);
        propertyID = getIntent().getLongExtra(Constants.ID_KEY, defaultValue: 0);
        databaseHelper = new AppDatabaseHelper( context: this);

        property = databaseHelper.getPropertyWithID(propertyID);
        initialiseUIComponents();
    }

    public void initialiseUIComponents()
    {
        cardNumberET = findViewById(R.id.cardNumber);
        taxDetailsLL = findViewById(R.id.taxDetails);
        taxPaidLL = findViewById(R.id.taxPaidDetails);
        amountPaidTV = findViewById(R.id.taxamount);
        dateTV = findViewById(R.id.date);
        amountTV = findViewById(R.id.amount);

        taxList = databaseHelper.getTaxWithPropertyID(propertyID);
        if (taxList.size() > 0 ) {
            boolean taxPaid = false;
            for (Tax tax : taxList) {
```

Fig 6.1.3: Pay tax activity

6.2 Screenshots

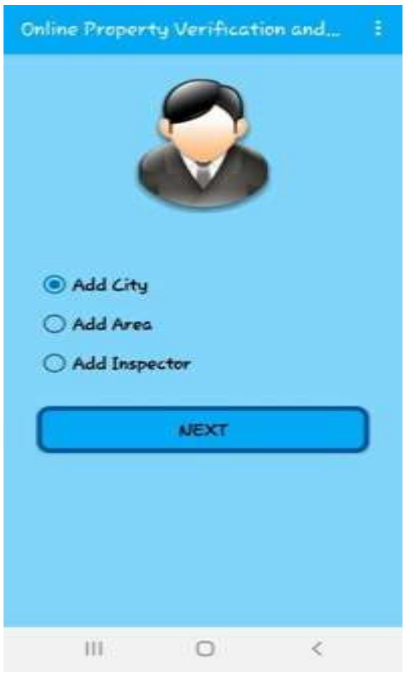

Online Property Verification and...

Online Property Verification and Tax Payment Application for Smart City

Select User
(●) Admin
( ) Property Owner
( ) Area Inspector

NEXT

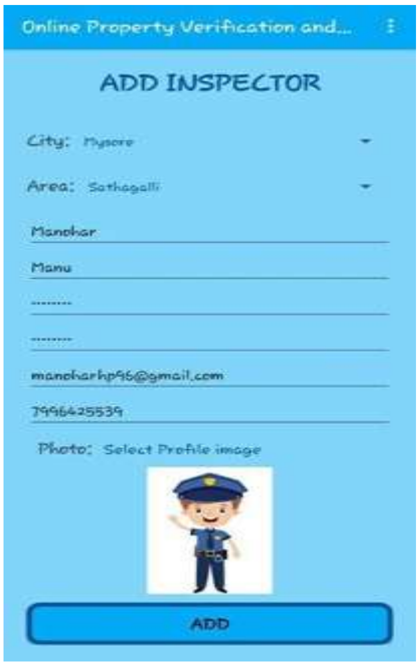
Fig 6.2.0: Main Menu


Online Property Verification and...

(●) Add City
( ) Add Area
( ) Add Inspector

NEXT

Fig 6.2.1: Admin feature


Online Property Verification and...

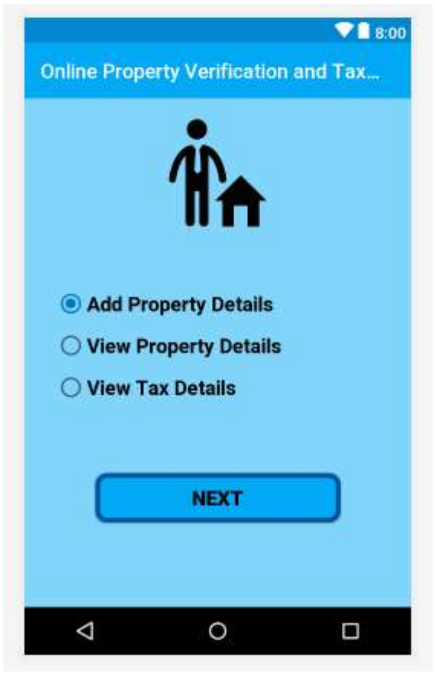
ADD INSPECTOR

City: Mysore
Area: Sathagalli
Manohar
Manu

----
manoharhp96@gmail.com
7996425539
Photo: Select Profile image

ADD

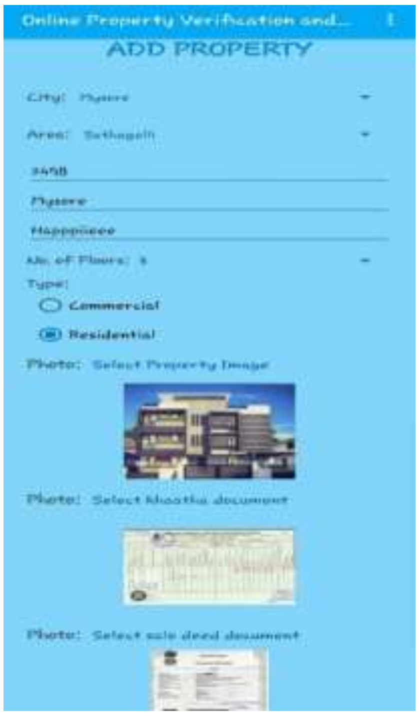
Fig 6.2.2: Add Inspector


8:00
Online Property Verification and Tax...

(●) Add Property Details
( ) View Property Details
( ) View Tax Details

NEXT

Fig 6.2.3: Property Owner Home


Online Property Verification and...

ADD PROPERTY

City: Mysore

Area: Sathagalli

3458

Mysore

Happpiieee

No. of Floors: 5

Type:

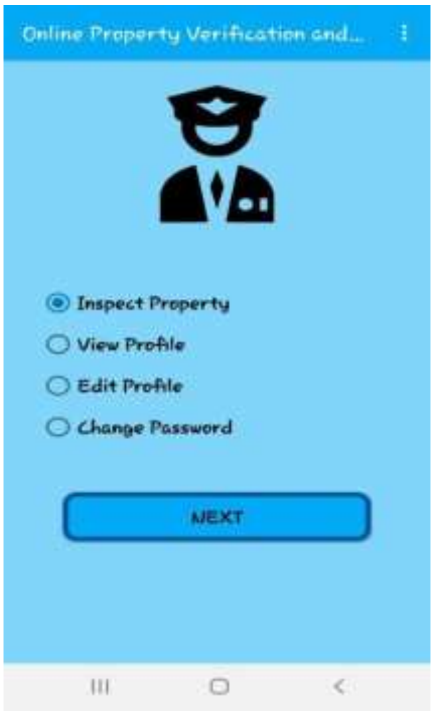
Commercial [✓ Residential]

Photo: Select Property Image

Photo: Select Khaatha document

Photo: Select sale deed document

Fig 6.2.4: Add Property


Online Property Verification and...

[✓ Inspect Property]

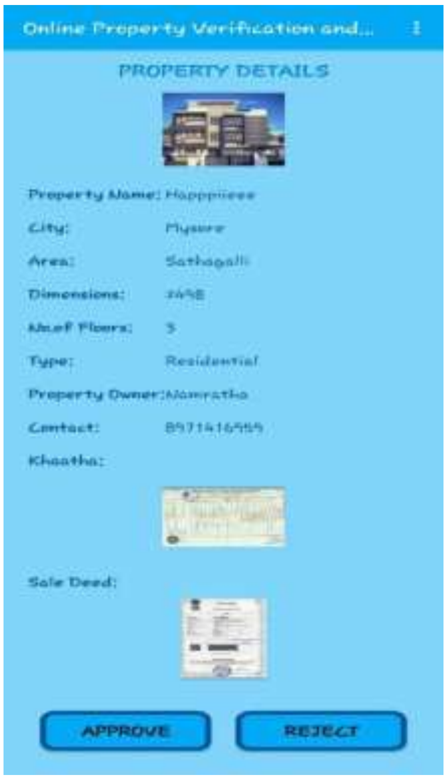
View Profile

Edit Profile

Change Password

NEXT

Fig 6.2.5: Inspector Home

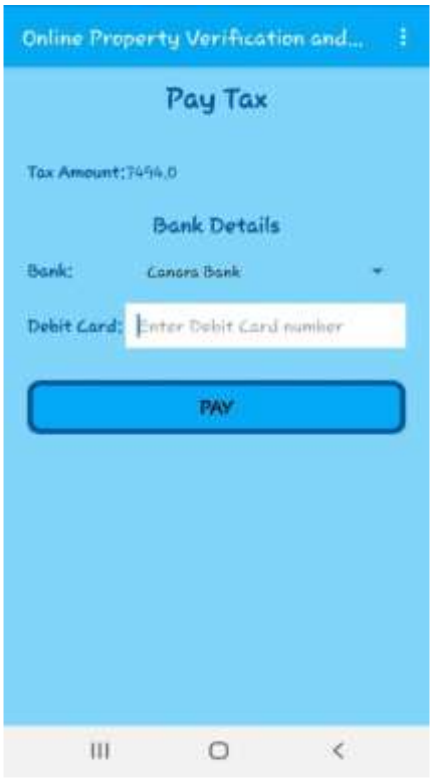

Online Property Verification and...

PROPERTY DETAILS

Property Name:	Happpiieee
City:	Mysore
Area:	Sathagalli
Dimensions:	3458
No.of Floors:	5
Type:	Residential
Property Owner:	Namratha
Contact:	8971416555
Khaatha:	
Sale Deed:	

APPROVE REJECT

Fig 6.2.6: Inspect Property


Online Property Verification and...

Pay Tax

Tax Amount:7494.0

Bank Details

Bank: Canara Bank

Debit Card: Enter Debit Card number

PAY

Fig 6.2.7: Pay Tax

CHAPTER-7

SYSTEM TESTING

A crucial process in software development that aims to assess the quality, functionality, and reliability of a software system is software testing. The software is executed with the intention of identifying defects, errors, or discrepancies between the expected and actual results.

The primary objectives of software testing are:

Defects are found: Uncovering any defects or bugs present in the software system is the main goal of testing. Developers may identify and record these problems to enhance software quality.

Quality enhancement: The overall quality of the software is enhanced by testing, which detects and fixes errors. A reliable, stable software system is ensured, and a satisfactory user experience is delivered.

Risks are mitigated by testing: Potential risks and vulnerabilities in the software system, such as security flaws or performance bottlenecks, are identified by testing. By these issues being addressed, the risk of failure or malfunctioning is minimised.

Requirements validation: Testing ensures that the software system satisfies user demands and meets criteria. Ensure the system works properly in its intended setting.

Various techniques, methodologies, and levels are encompassed by software testing.

Unit testing: To ensure software system functioning, components are evaluated.

Integration testing: Testing system integration and interaction.

System Testing: The full software bundle is evaluated for behaviour and functioning.

Acceptance testing: To meet users' needs, the programme is tested.

7.1 Test Cases

Test Case Id	Test Case Name	Test Case Description	Test Steps				Test Status P/F
			Steps	I/P Given	Expected O/P	Actual O/P	
TC01	Login	To check the client is signed in user name and password are correct.	Client Logged in with Valid username & password	Username and password that are valid	Login successful	Completed login.	Pass
	Login	To check the Client is signed in User name and password are correct	Client logged in with Invalid username & password	Invalid user name and password	Login unsuccessful	Login unsuccessful	Pass
TC02	Add City	Manager will insert city	Admin will Add City details	Valid City Details	Added successfully	Added successfully	Pass
TC03	Add Area	Supervisor to Establish Separate Department	Admin will Add Area details	Valid Area Details	Added successfully	Added successfully	Pass
TC04	Add Inspector	The Inspector Role Will Be Added By Administration.	Admin will Add Inspector details	Valid Inspector Details	Added successfully	Added successfully	Pass
TC05	Add Property Details	Owner of the Property Will Update Information	Details About the Property Will Be Added By the Owner.	Valid Property Details	Added successfully	Added successfully	Pass

TC06	Learn More About This Home	Owner of the Property Can Access Information	Property Owner can view the list of Property Details	List of Property Details	Viewed Successfuly	Viewed Successfuly	Pass
TC07	View Tax Details	Owner of Real Estate Can Check Tax Information	Property Owner can view the list of Tax Details	List of Tax Details	Viewed Successfuly	Viewed Successfuly	Pass
TC08	Registration	In order to confirm that the Property Owner has registered, correct information must be entered.	Enter all valid Property Owner detail	Valid detail	Registered successfully	Registered successfully	Pass
	Registration	To verify that the Property Owner has registered by entering valid detail	If any of the information you submitted is incorrect or missing, an error will occur.	Invalid detail	Registered unsuccessfully	Registered unsuccessfully	Pass
TC09	Inspect Property	Inspectors are able to Conduct Premises Inspections	Everything Must Pass a Strict Inspection	Specifics of a Validly Inspected Property	Passed all inspections	Passed all inspections	Pass
TC10	Edit profile	Information may be altered by the inspector.	Detailed Inspecting and Revised	Inspectoral Specifics Cleaned Upsuccessfully	Successfully edited Inspector details	Corrections made to inspector details	Pass

TC11	View Profile	The profile is seen by the inspector.	The examiner looked at a profile	Profile was successfully viewed by inspector.	Inspector was able to see profile in full	Successful profile inspection by inspector	Pass
TC12	Logout Menu	In case the user selects logout menu option, they should be redirected back to user selection screen	Select logout menu option	Logout option is selected from the options menu	User is taken to user selection screen	The user is sent to a login/signup page.	Pass

Table 7.1.0 : Test Case

CHAPTER-8

CONCLUSION

By providing property owners with an efficient and user-friendly interface, tax information, and convenient payment options, The software may make it easier to calculate and pay real estate taxes. It gives people the ability to pay their fair share of taxes and contribute to the growth of the country.

CHAPTER-9

FUTURE ENHANCEMENT

The app can be further enhanced by the following feature:

1. Past Land Tax Payment Reports for taxes already collected can be viewed here.
2. This form will include payments for utility bills like water and energy.
3. If a vacant property is not managed, fines could be imposed.
4. Tax calculator for real estate

BIBLIOGRAPHY

[1] Kalkundre, P., & Mahajan, V. (2016). Android-based Property Tax Calculation and Payment System. In 2016 International Conference on Inventive Computation Technologies (ICICT) (pp. 1-5). IEEE. DOI: 10.1109/INVENTIVE.2016.7860262

[2] Mamgain, D., Kumar, S., & Bhardwaj, M. (2017). A Mobile Application for Property Tax Calculation and Payment System. International Journal of Engineering Research and General Science, 5(4), 218-221.

[3] Nimbalkar, A., Potdukhe, P., & Chavan, S. (2017). Property Tax Payment System using Android. In 2017 International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 1531-1535). IEEE. DOI: 10.1109/ICCONS.2017.8250802

[4] Patel, D. B., Kadam, P. S., & Shewale, S. S. (2017). An Android Based Property Tax Calculation and Payment System. In 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT) (pp. 334-339). IEEE. DOI: 10.1109/ICICCT1.2017.7975270

[5] Shetty, A., Bhoir, A., Iyer, S., & Nadkarni, V. (2016). An Android Application for Property Tax Payment System. International Journal of Science and Research (IJSR), 5(12), 292-294.

[6] Tripathi, S., Tiwari, D., & Gupta, P. (2016). Android Based Property Tax Payment System. In 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT) (pp. 488-491). IEEE. DOI: 10.1109/icctict.2016.7512059

Installation

USER MANUAL

- The latest version of Android Studio should be downloaded and installed from the official website (https://developer.android.com/studio).
- The installation process should be completed by following the on-screen instructions.
- Android Studio should be launched after installation.
- The Project is being opened.

Android Studio should be opened.

- "Open an existing Android Studio project" should be clicked on or the project folder should be navigated to by going to "File" > "Open".
- The project folder should be selected and "OK" should be clicked.
- The project should be imported and built by Android Studio.

The Android device or emulator is being set up.

- Your computer should be connected to your Android device using a USB cable, or an Android emulator should be set up if you don't have a physical device.

- USB debugging should be enabled on your Android device. "Settings" > "Developer options" (or "Developer settings") should be gone to and "USB debugging" should be toggled on.
- (Note: If your settings do not show "Developer options," "Build number" should be tapped on seven times in "Settings" > "About phone" to activate developer options.)
- If an emulator is being used, it can be set up by going to "Tools" > "AVD Manager" and a new virtual device can be created based on the desired specifications.
- The emulator should be started or your physical device should be connected and recognised by Android Studio.

The App is being run.

- The toolbar at the top of the window should be located in Android Studio.
- The target device should be selected from the device dropdown menu. If an emulator is being used, the virtual device you created should be chosen.
- The app can be launched by clicking on the "Run" button (green triangle icon) or by going to "Run" > "Run 'app'".
- The project will be built by Android Studio, the app will be deployed to the selected device, and it will be launched automatically.
- The app should be waited for to load and start on the device or emulator.
- The app's functionality can be tested by interacting with it on the device or emulator.

Troubleshooting:

- Ensure that the necessary SDK components and dependencies are installed if any build or installation errors are encountered. You will be prompted by Android Studio to install missing components if required.
- Make sure USB debugging is enabled and that your device drivers are up to date if your device is not detected.
- Error messages and stack traces should be checked in the logcat console in Android Studio if the app crashes or behaves unexpectedly. This information can be used to identify and fix issues in your code.