APARTMENT MANAGEMENT SYSTEM (AMS)

○ CONTRIBUTORS

1. Khawaja Abdul Moiz
2. Muhammad Harim
3. Areeb Alvi

➢ Purpose:

The purpose of an "Apartment Management System" is to facilitate efficient and organized management of various tasks and processes related to the administration and maintenance of apartment complexes or residential buildings. The system is designed to streamline operations for property managers, tenants, and other stakeholders. Here are some common purposes of an Apartment Management System:

Tenant Management:

- Keep track of tenant information, including contact details, lease agreements, and payment history.
- Automate the process of tenant onboarding and off boarding.

Rent and Billing:

- Manage rent collection, generate invoices, and keep a record of payment transactions.
- Provide automated reminders for upcoming rent payments.

Maintenance Requests:

- Allow tenants to submit maintenance requests through the system.
- Streamline the process of assigning and tracking maintenance tasks.
- Amenities and Facilities Management:
- Keep a record of available amenities and facilities in the apartment complex.
- Schedule maintenance and cleaning for shared spaces such as gyms, pools, and common areas.

Communication:
- Facilitate communication between property managers and tenants.
- Provide a platform for announcements, notifications, and updates.

Security and Access Control:
- Implement features for managing access control, such as keyless entry systems or visitor management.
- Keep track of security-related incidents and measures.

Document Management:
- Store and manage important documents related to property ownership, contracts, and legal agreements.

Financial Reporting:
- Generate financial reports for property owners and managers.
- Keep track of expenses, revenue, and overall financial health of the property.

Occupancy and Vacancy Tracking:
- Monitor and manage the occupancy status of individual units.
- Plan for upcoming vacancies and coordinate leasing efforts.

Compliance and Regulations:
- Ensure compliance with local regulations and housing laws.
- Store relevant compliance documentation.

By addressing these aspects, an Apartment Management System aims to enhance the overall efficiency of property management, improve communication between stakeholders, and create a more convenient and enjoyable living experience for tenants.

➢ Functionality:

The functionality of an Apartment Management System encompasses a wide range of features to address the needs of property managers, tenants, and other stakeholders involved in the management of residential complexes. Here are the key functionalities typically found in such a system:

1. User Authentication and Authorization
2. Tenant Information Management
3. Rent and Billing Management
4. Maintenance Request System
5. Amenities and Facilities Management
6. Communication Platform
7. Security and Access Control
8. Document Storage and Management
9. Financial Reporting
10. Occupancy and Vacancy Tracking
11. Compliance Monitoring
12. Analytics and Reporting
13. Data Security and Privacy

By incorporating these functionalities, an Apartment Management System aims to automate and streamline various aspects of property management, enhancing the overall efficiency and experience for both property managers and tenants.

➢ Usage Instructions for AMS:

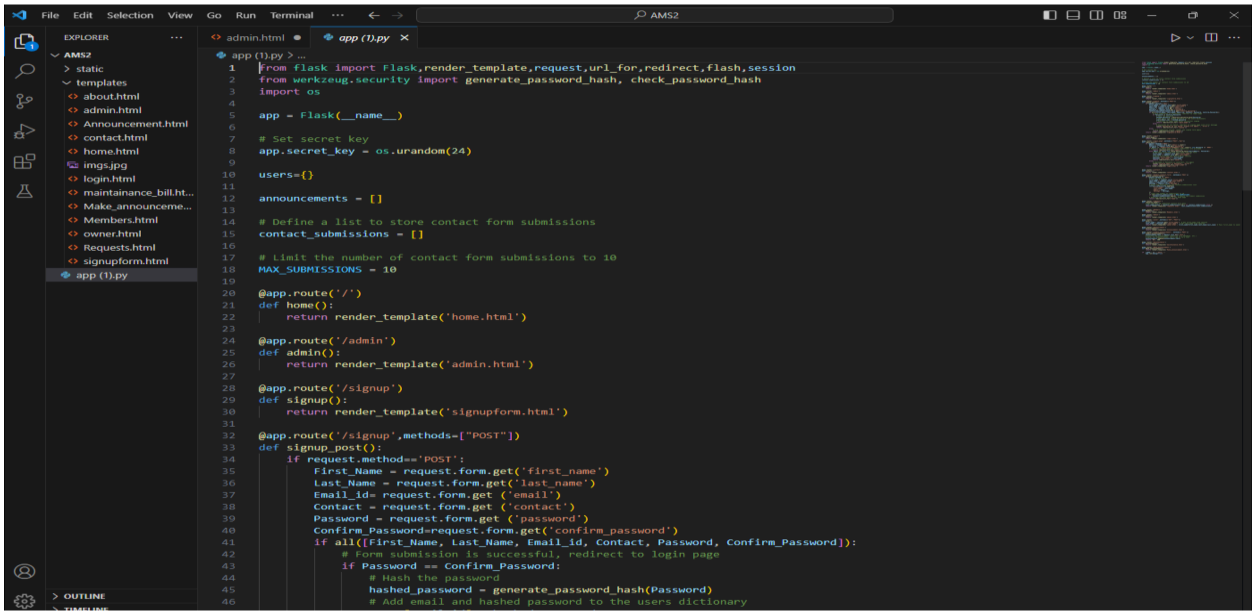
Usage instructions for an Apartment Management System. Note that these are general guidelines, and the specifics may vary based on the actual implementation and technology stack used in the project.

1. User Registration and Login:

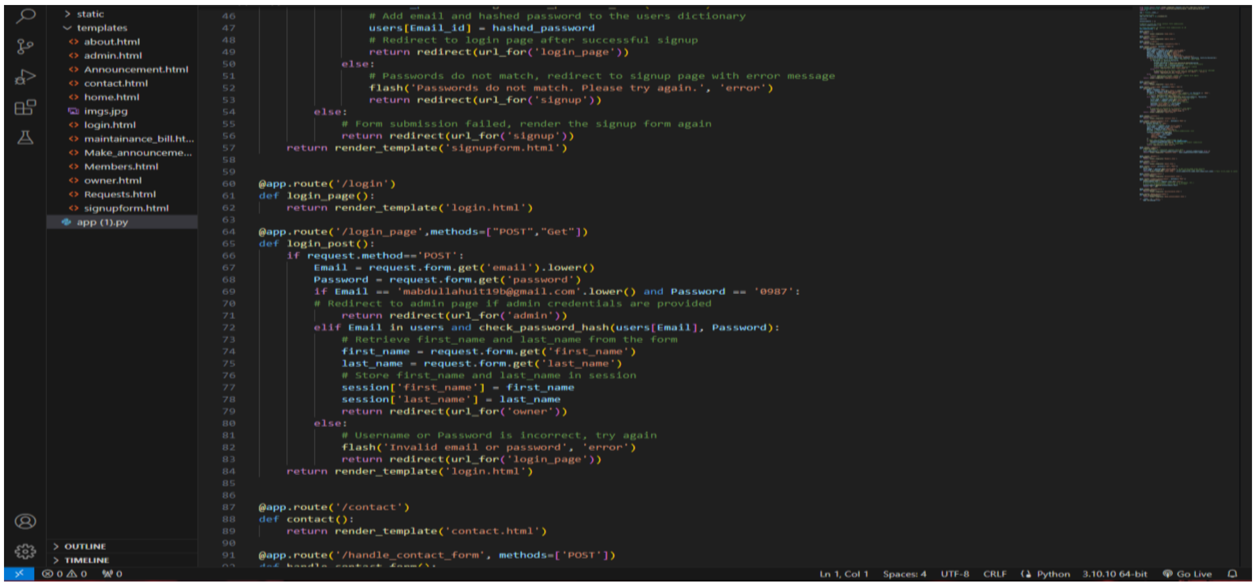
- **For Property Managers:**
  - Register with the system using a valid email address.
  - Log in using the registered credentials.

- **For Tenants:**
  - Receive login credentials from the property manager or use a self-registration feature.
  - Log in to the system.

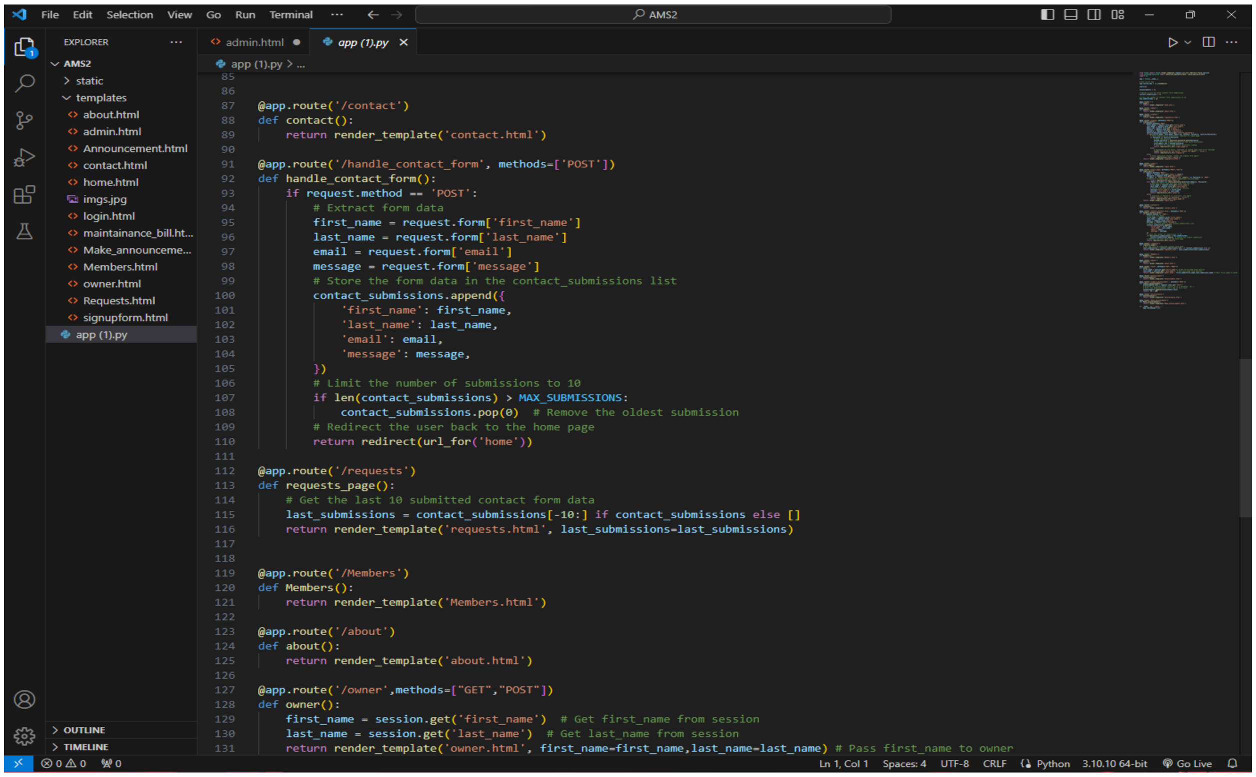
➢ Screen Shots:



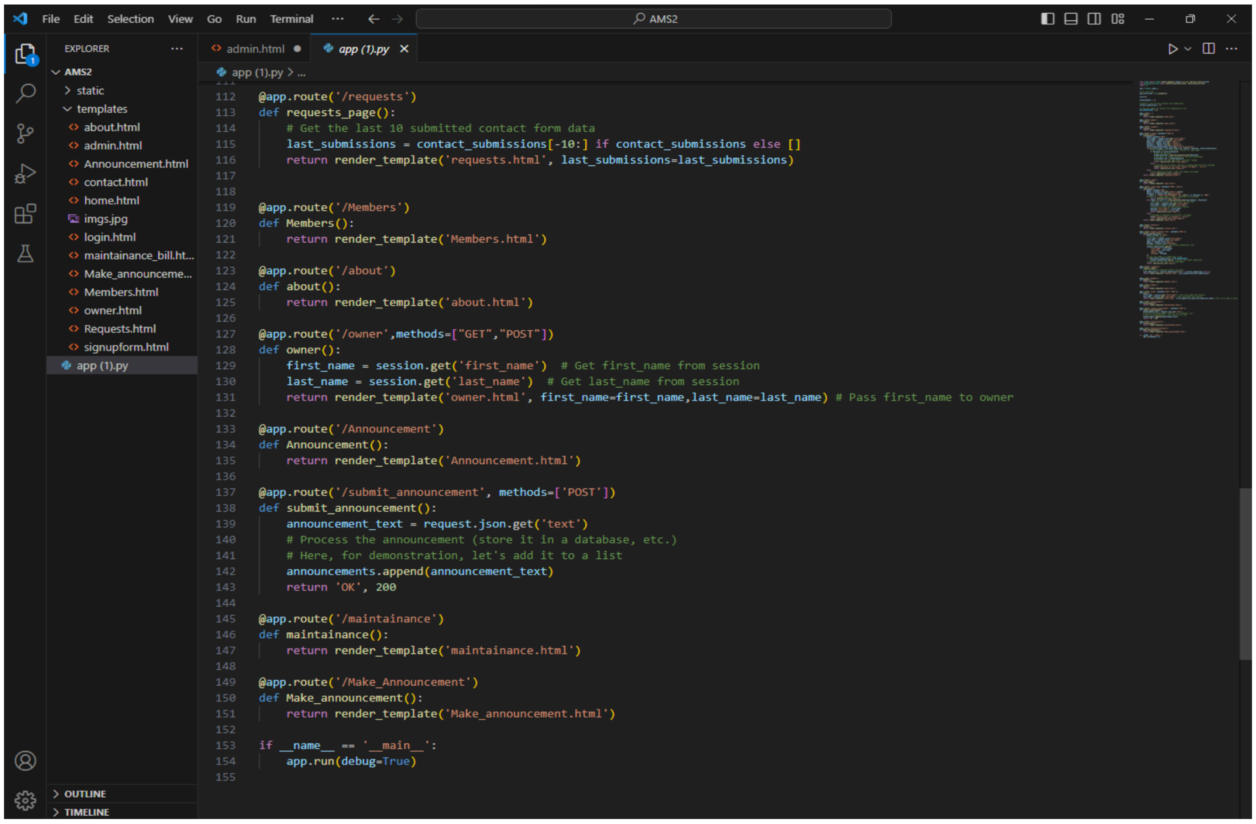
```python
from flask import Flask,render_template,request,url_for,redirect,flash,session
from werkzeug.security import generate_password_hash, check_password_hash
import os

app = Flask(__name__)

# Set secret key
app.secret_key = os.urandom(24)

users={}

announcements = []

# Define a list to store contact form submissions
contact_submissions = []

# Limit the number of contact form submissions to 10
MAX_SUBMISSIONS = 10

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/admin')
def admin():
    return render_template('admin.html')

@app.route('/signup')
def signup():
    return render_template('signupform.html')

@app.route('/signup',methods=["POST"])
def signup_post():
    if request.method=='POST':
        First_Name = request.form.get('first_name')
        Last_Name = request.form.get('last_name')
        Email_id= request.form.get ('email')
        Contact = request.form.get ('contact')
        Password = request.form.get ('password')
        Confirm_Password=request.form.get('confirm_password')
        if all([First_Name, Last_Name, Email_id, Contact, Password, Confirm_Password]):
            # Form submission is successful, redirect to login page
            if Password == Confirm_Password:
                # Hash the password
                hashed_password = generate_password_hash(Password)
                # Add email and hashed password to the users dictionary
```



```python
                # Add email and hashed password to the users dictionary
                users[Email_id] = hashed_password
                # Redirect to login page after successful signup
                return redirect(url_for('login_page'))
            else:
                # Passwords do not match, redirect to signup page with error message
                flash('Passwords do not match. Please try again.', 'error')
                return redirect(url_for('signup'))
        else:
            # Form submission failed, render the signup form again
            return redirect(url_for('signup'))
    return render_template('signupform.html')


@app.route('/login')
def login_page():
    return render_template('login.html')

@app.route('/login_page',methods=["POST","Get"])
def login_post():
    if request.method=='POST':
        Email = request.form.get('email').lower()
        Password = request.form.get('password')
        if Email == 'mabdullahuit19b@gmail.com'.lower() and Password == '0987':
        # Redirect to admin page if admin credentials are provided
            return redirect(url_for('admin'))
        elif Email in users and check_password_hash(users[Email], Password):
            # Retrieve first_name and last_name from the form
            first_name = request.form.get('first_name')
            last_name = request.form.get('last_name')
            # Store first_name and last_name in session
            session['first_name'] = first_name
            session['last_name'] = last_name
            return redirect(url_for('owner'))
        else:
            # Username or Password is incorrect, try again
            flash('Invalid email or password', 'error')
            return redirect(url_for('login_page'))
    return render_template('login.html')


@app.route('/contact')
def contact():
    return render_template('contact.html')

@app.route('/handle_contact_form', methods=['POST'])
```

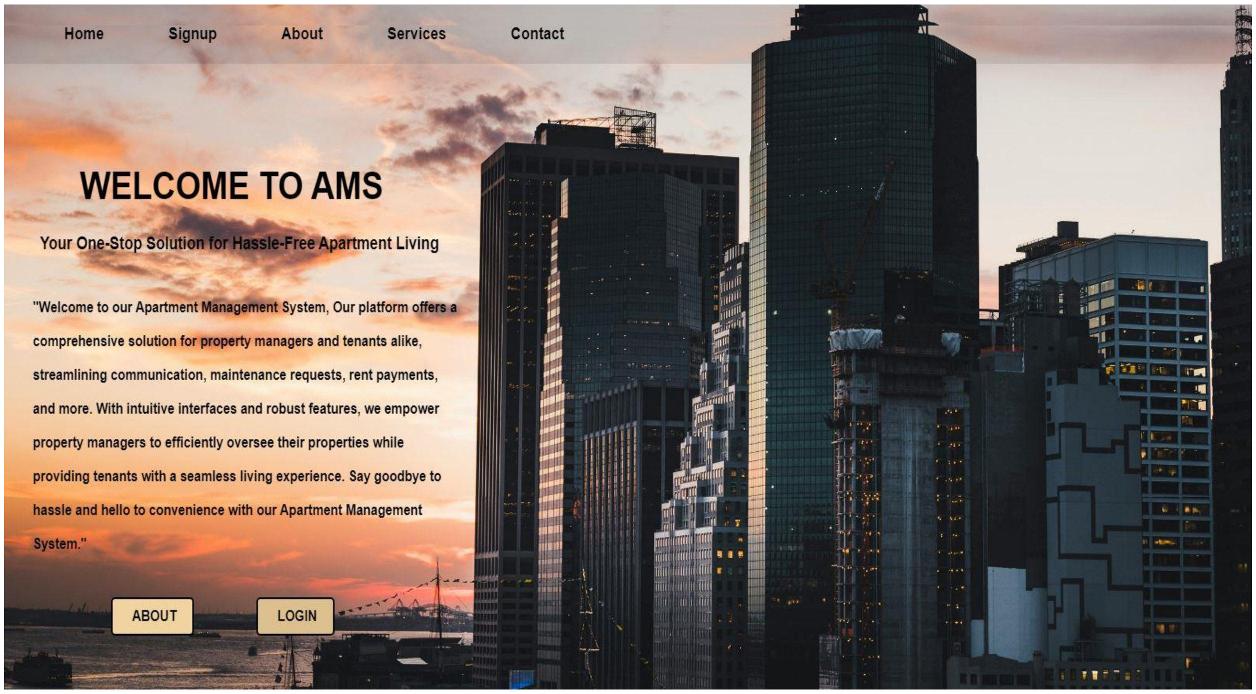


```python


@app.route('/contact')
def contact():
    return render_template('contact.html')

@app.route('/handle_contact_form', methods=['POST'])
def handle_contact_form():
    if request.method == 'POST':
        # Extract form data
        first_name = request.form['first_name']
        last_name = request.form['last_name']
        email = request.form['email']
        message = request.form['message']
        # Store the form data in the contact_submissions list
        contact_submissions.append({
            'first_name': first_name,
            'last_name': last_name,
            'email': email,
            'message': message,
        })
        # Limit the number of submissions to 10
        if len(contact_submissions) > MAX_SUBMISSIONS:
            contact_submissions.pop(0)  # Remove the oldest submission
        # Redirect the user back to the home page
        return redirect(url_for('home'))

@app.route('/requests')
def requests_page():
    # Get the last 10 submitted contact form data
    last_submissions = contact_submissions[-10:] if contact_submissions else []
    return render_template('requests.html', last_submissions=last_submissions)


@app.route('/Members')
def Members():
    return render_template('Members.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/owner',methods=["GET","POST"])
def owner():
    first_name = session.get('first_name')  # Get first_name from session
    last_name = session.get('last_name')  # Get last_name from session
    return render_template('owner.html', first_name=first_name,last_name=last_name) # Pass first_name to owner
```



```python
@app.route('/requests')
def requests_page():
    # Get the last 10 submitted contact form data
    last_submissions = contact_submissions[-10:] if contact_submissions else []
    return render_template('requests.html', last_submissions=last_submissions)


@app.route('/Members')
def Members():
    return render_template('Members.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/owner',methods=["GET","POST"])
def owner():
    first_name = session.get('first_name')  # Get first_name from session
    last_name = session.get('last_name')  # Get last_name from session
    return render_template('owner.html', first_name=first_name,last_name=last_name) # Pass first_name to owner
```

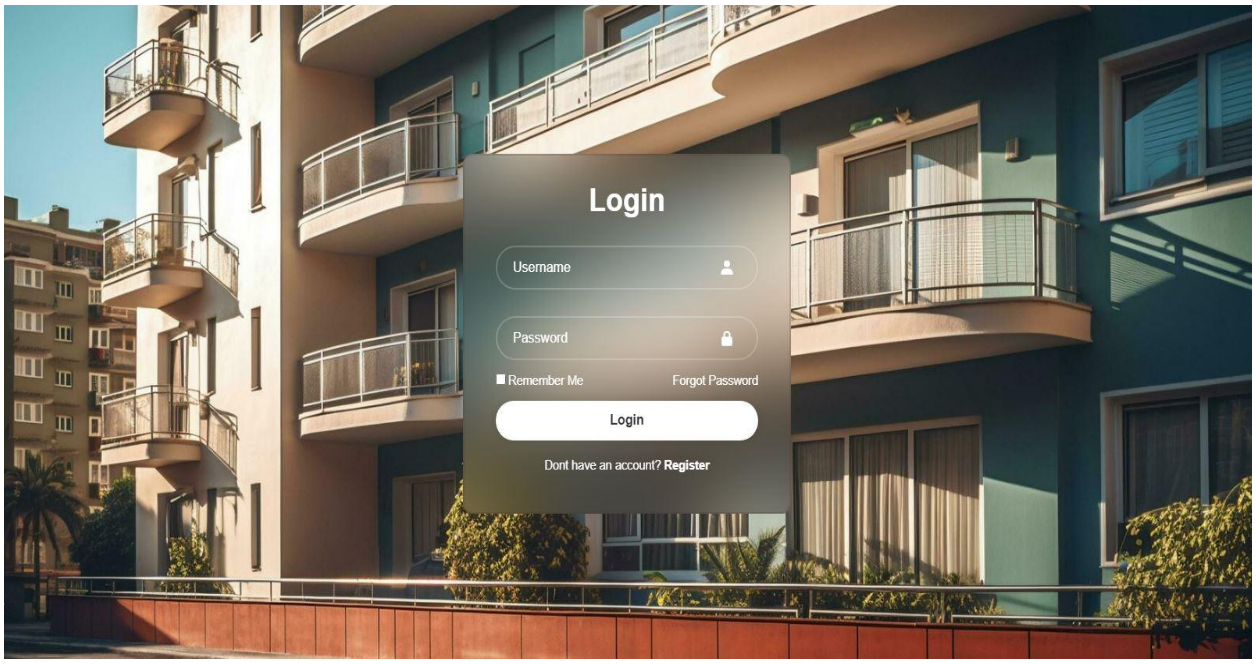
```python

@app.route('/Announcement')
def Announcement():
    return render_template('Announcement.html')

@app.route('/submit_announcement', methods=['POST'])
def submit_announcement():
    announcement_text = request.json.get('text')
    # Process the announcement (store it in a database, etc.)
    # Here, for demonstration, let's add it to a list
    announcements.append(announcement_text)
    return 'OK', 200

@app.route('/maintainance')
def maintainance():
    return render_template('maintainance.html')

@app.route('/Make_Announcement')
def Make_announcement():
    return render_template('Make_announcement.html')

if __name__ == '__main__':
    app.run(debug=True)

```

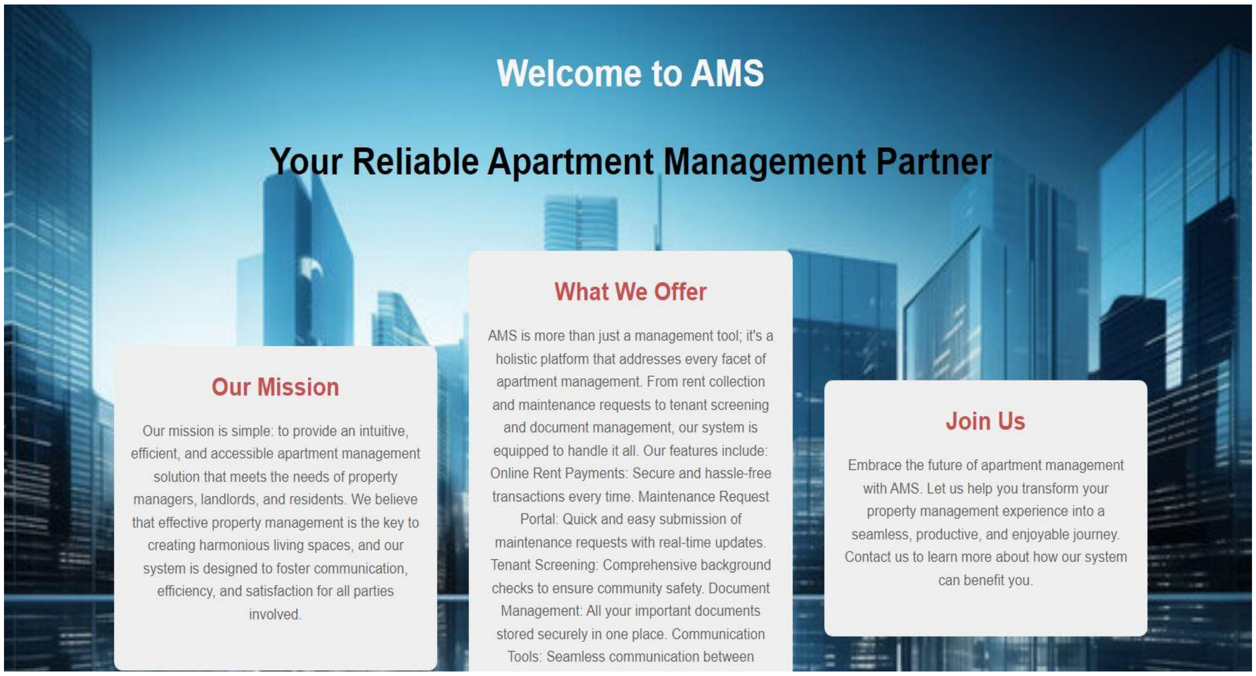


WELCOME TO AMS

Your One-Stop Solution for Hassle-Free Apartment Living

"Welcome to our Apartment Management System, Our platform offers a comprehensive solution for property managers and tenants alike, streamlining communication, maintenance requests, rent payments, and more. With intuitive interfaces and robust features, we empower property managers to efficiently oversee their properties while providing tenants with a seamless living experience. Say goodbye to hassle and hello to convenience with our Apartment Management System."





Welcome to AMS

Your Reliable Apartment Management Partner

What We Offer

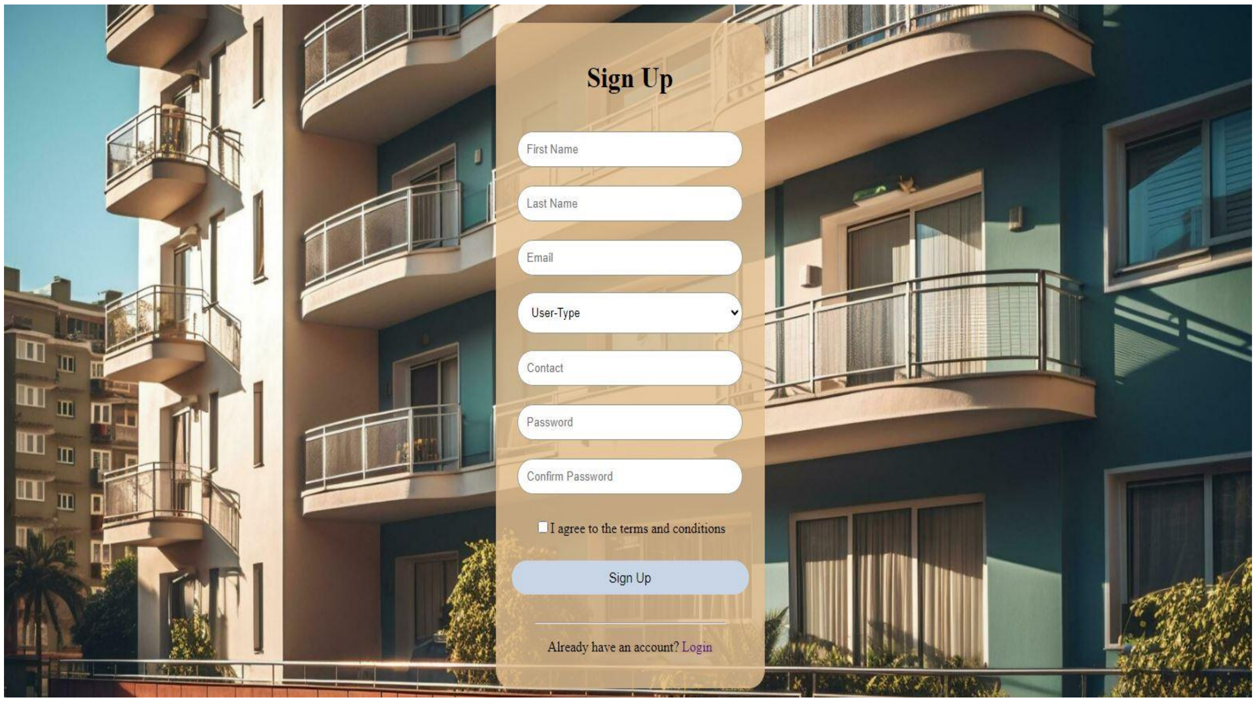
AMS is more than just a management tool; it's a holistic platform that addresses every facet of apartment management. From rent collection and maintenance requests to tenant screening and document management, our system is equipped to handle it all. Our features include: Online Rent Payments: Secure and hassle-free transactions every time. Maintenance Request Portal: Quick and easy submission of maintenance requests with real-time updates. Tenant Screening: Comprehensive background checks to ensure community safety. Document Management: All your important documents stored securely in one place. Communication Tools: Seamless communication between

Our Mission

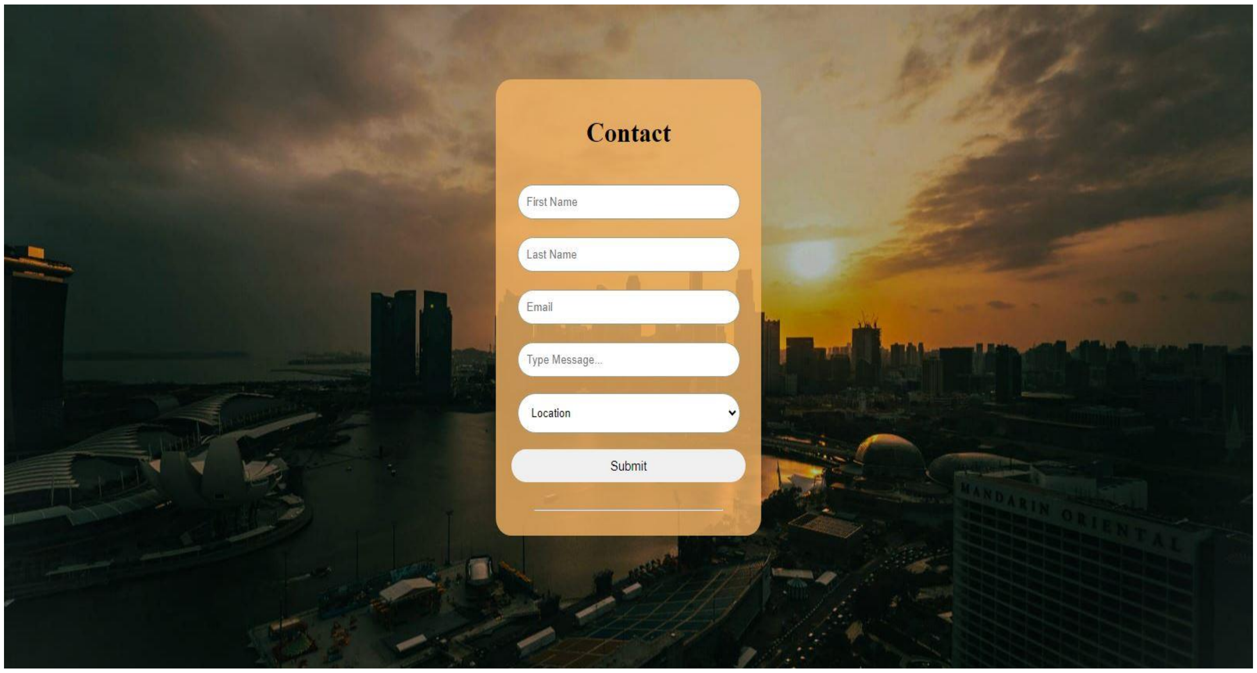
Our mission is simple: to provide an intuitive, efficient, and accessible apartment management solution that meets the needs of property managers, landlords, and residents. We believe that effective property management is the key to creating harmonious living spaces, and our system is designed to foster communication, efficiency, and satisfaction for all parties involved.

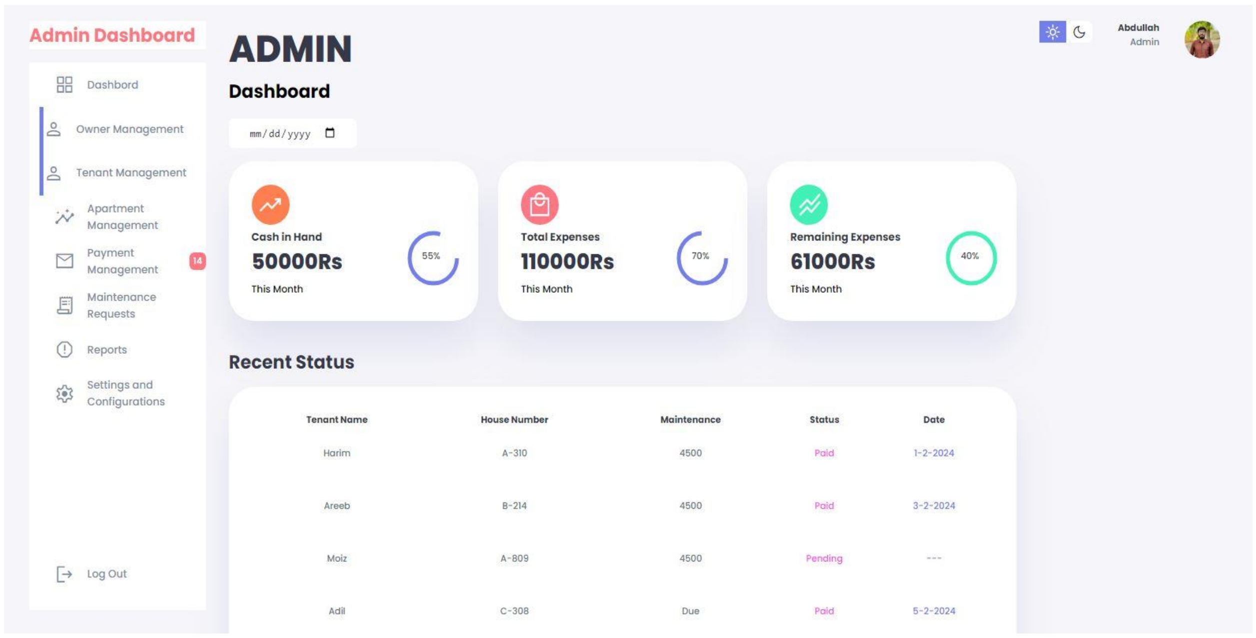
Join Us

Embrace the future of apartment management with AMS. Let us help you transform your property management experience into a seamless, productive, and enjoyable journey. Contact us to learn more about how our system can benefit you.



Sign Up





ADMIN

Dashboard

Cash in Hand	50000Rs	This Month
Total Expenses	110000Rs	This Month
Remaining Expenses	61000Rs	This Month

55% 70% 40%

Recent Status

Tenant Name	House Number	Maintenance	Status	Date
Harim	A-310	4500	Paid	1-2-2024
Areeb	B-214	4500	Paid	3-2-2024
Moiz	A-809	4500	Pending	---
Adil	C-308	Due	Paid	5-2-2024

➢ Links:

1.Github:

https://github.com/KhawajaAbdulMoiz/.AMS

2.Video:

https://www.facebook.com/share/v/koLkwi4Vv4XqMPot/?mibextid=w8EBqM