

Here is the folder structure in my root folder

```
32project
++ main.py
++ static/
|  ++ style.css
++ templates/
    ++ admin_users.html
    ++ base.html
    ++ dashboard_admin.html
    ++ dashboard_operator.html
    ++ dashboard_viewer.html
    ++ index.html
    ++ login.html
    ++ register.html
    ++ verify_email.html
```

Here is files code:

```
main.py
``python
from flask import Flask, render_template, redirect, url_for, flash, request, Response, jsonify
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user, current_user
from werkzeug.security import generate_password_hash, check_password_hash
import re
import cv2
import platform
import os
from dotenv import load_dotenv

# ----- Load Environment -----
load_dotenv()

# ----- GPIO Setup -----
IS_PI = platform.system() == "Linux"

if IS_PI:
    import RPi.GPIO as GPIO
    GPIO.setmode(GPIO.BCM)
    print("Running on Raspberry Pi: GPIO enabled")
else:
    print("Not running on Raspberry Pi: GPIO simulated")
    from gpiozero import LED, Device
    from gpiozero.pins.mock import MockFactory
    Device.pin_factory = MockFactory()

# Robot pins
```

```

PIN_FORWARD = 17
PIN_BACKWARD = 27
PIN_LEFT = 22
PIN_RIGHT = 23

if IS_PI:
    GPIO.setup(PIN_FORWARD, GPIO.OUT)
    GPIO.setup(PIN_BACKWARD, GPIO.OUT)
    GPIO.setup(PIN_LEFT, GPIO.OUT)
    GPIO.setup(PIN_RIGHT, GPIO.OUT)

# ----- App Setup -----
app = Flask(__name__)
app.config['SECRET_KEY'] = os.getenv('MY_SECRET_KEY') or os.urandom(32)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'

# ----- Models -----
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(100), unique=True, nullable=False)
    email = db.Column(db.String(150), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    role = db.Column(db.String(50), default='viewer') # viewer, operator, admin

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

# ----- Camera -----
camera = cv2.VideoCapture(0)

def generate_frames():
    while True:
        success, frame = camera.read()
        if not success:
            continue
        _, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n' + b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/video_feed')
@login_required

```

```

def video_feed():
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame'
')

# ----- Robot Movement -----
def move_robot(direction):
    if IS_PI:
        GPIO.output(PIN_FORWARD, GPIO.LOW)
        GPIO.output(PIN_BACKWARD, GPIO.LOW)
        GPIO.output(PIN_LEFT, GPIO.LOW)
        GPIO.output(PIN_RIGHT, GPIO.LOW)
        if direction == "forward": GPIO.output(PIN_FORWARD, GPIO.HIGH)
        elif direction == "backward": GPIO.output(PIN_BACKWARD, GPIO.HIGH)
        elif direction == "left": GPIO.output(PIN_LEFT, GPIO.HIGH)
        elif direction == "right": GPIO.output(PIN_RIGHT, GPIO.HIGH)
    else:
        print(f"Simulated move: {direction}")

@app.route('/move/<direction>', methods=['POST'])
@login_required
def move(direction):
    if current_user.role not in ['operator', 'admin']:
        return jsonify({"error": "Access denied"}), 403
    move_robot(direction)
    return jsonify({"status": f"Moved {direction}"}), 200

# ----- Routes -----
@app.route('/')
def home():
    # If user already logged in, send them to their dashboard directly
    if current_user.is_authenticated:
        if current_user.role == 'admin':
            return redirect(url_for('dashboard_admin'))
        elif current_user.role == 'operator':
            return redirect(url_for('dashboard_operator'))
        else:
            return redirect(url_for('dashboard_viewer'))

    # Otherwise show the landing page with base video
    return render_template('index.html')

# ----- Register -----
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']

```

```

password = request.form['password']

# Validation
if User.query.filter_by(username=username).first():
    flash("Username already exists.", "danger")
    return redirect(url_for('register'))

if User.query.filter_by(email=email).first():
    flash("Email already registered.", "danger")
    return redirect(url_for('register'))

if not re.match(r'^[w\.-]+@[w\.-]+\.\w+$', email):
    flash("Invalid email format.", "danger")
    return redirect(url_for('register'))

if not re.match(r'^(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&]).{8,}$', password):
    flash("Password must be at least 8 chars, include 1 uppercase, 1 number, and 1
special character.", "danger")
    return redirect(url_for('register'))

hashed_password = generate_password_hash(password)
new_user = User(username=username, email=email, password=hashed_password, role='vi
ewer')
db.session.add(new_user)
db.session.commit()

login_user(new_user)
flash("Registration successful! Welcome to Viewer Dashboard.", "success")
return redirect(url_for('dashboard_viewer'))

return render_template('register.html')

# ----- Login -----
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        user = User.query.filter_by(username=username).first()
        if user and check_password_hash(user.password, password):
            login_user(user)
            if user.role == 'admin':
                return redirect(url_for('dashboard_admin'))
            elif user.role == 'operator':
                flash(f"Welcome {user.username}!", "success")
                return redirect(url_for('dashboard_operator'))
            else:

```

```

        flash(f"Welcome {user.username}!", "success")
        return redirect(url_for('dashboard_viewer'))
    else:
        flash("Invalid username or password.", "danger")

    return render_template('login.html')

# ----- Logout -----
@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash("Logged out successfully.", "success")
    return redirect(url_for('login'))

# ----- Dashboards -----
@app.route('/dashboard_viewer')
@login_required
def dashboard_viewer():
    return render_template('dashboard_viewer.html')

@app.route('/dashboard_operator')
@login_required
def dashboard_operator():
    if current_user.role not in ['operator', 'admin']:
        flash("Access denied!", "danger")
        return redirect(url_for('dashboard_viewer'))
    return render_template('dashboard_operator.html')

@app.route('/dashboard_admin')
@login_required
def dashboard_admin():
    if current_user.role != 'admin':
        flash("Access denied!", "danger")
        return redirect(url_for('dashboard_viewer'))
    users = User.query.all()
    existing_operator = User.query.filter_by(role='operator').first()
    return render_template('dashboard_admin.html', users=users, existing_operator=existing_operator)

# ----- Admin Actions -----
@app.route('/approve_operator/<int:user_id>')
@login_required
def approve_operator(user_id):
    if current_user.role != 'admin':
        flash("Access denied!", "danger")
        return redirect(url_for('dashboard_viewer'))

```

```

existing_operator = User.query.filter_by(role='operator').first()
if existing_operator:
    flash(f"Cannot promote. Operator '{existing_operator.username}' already exists.", "warning")
else:
    user = User.query.get_or_404(user_id)
    if user.role == 'viewer':
        user.role = 'operator'
        db.session.commit()
        flash(f"{user.username} promoted to operator.", "success")
    else:
        flash(f"{user.username} cannot be promoted.", "warning")
return redirect(url_for('dashboard_admin'))

@app.route('/demote_operator/<int:user_id>')
@login_required
def demote_operator(user_id):
    if current_user.role != 'admin':
        flash("Access denied!", "danger")
        return redirect(url_for('dashboard_viewer'))

    user = User.query.get_or_404(user_id)
    if user.role == 'operator':
        user.role = 'viewer'
        db.session.commit()
        flash(f"{user.username} demoted to viewer.", "success")
    else:
        flash(f"{user.username} cannot be demoted.", "warning")
    return redirect(url_for('dashboard_admin'))

@app.route('/remove_user/<int:user_id>')
@login_required
def remove_user(user_id):
    if current_user.role != 'admin':
        flash("Access denied!", "danger")
        return redirect(url_for('dashboard_viewer'))

    user = User.query.get_or_404(user_id)
    if user.role != 'admin':
        db.session.delete(user)
        db.session.commit()
        flash(f"{user.username} has been removed.", "success")
    else:
        flash("Cannot remove admin!", "danger")
    return redirect(url_for('dashboard_admin'))

# ----- Auto-create Admin -----
def create_default_admin():

```

```

# Check for existing admin by email or username
admin = User.query.filter(
    (User.username == 'Admin') |
    (User.email == 'admin@gmail.com')
).first()

if not admin:
    hashed_password = generate_password_hash("Thisisadmin01!")
    admin = User(
        username='Admin',
        email='admin@gmail.com',
        password=hashed_password,
        role='admin'
    )
    db.session.add(admin)
    db.session.commit()
    print("Default admin created!")
else:
    print("Admin already exists. Skipping creation.")

```

```

# ----- Run App -----
if __name__ == "__main__":
    with app.app_context():
        db.create_all()
        create_default_admin()
    app.run(host="0.0.0.0", port=5000, debug=True)

```

```
""
```

```

static\style.css
``css
body {
    font-family: 'Montserrat', sans-serif;
    margin: 0;
    padding: 20px;
    background-color: #f5f5f5;
}

```

```
.
form-container {
    max-width: 400px;
    margin: 0 auto 50px;
    padding: 20px;
    border: 1px solid #e0e0e0;
    border-radius: 8px;
    background-color: #fff;
}
```

```
.form-container h2 {  
    text-align: center;  
    color: #333;  
    font-weight: 600;  
    margin-bottom: 20px;  
}  
  
.form-container input {  
    padding: 10px;  
    font-size: 16px;  
    border: 1px solid #e0e0e0;  
    border-radius: 4px;  
    outline: none;  
}  
  
.button-primary {  
    padding: 10px;  
    background-color: #0a8f0a;  
    color: white;  
    border: none;  
    border-radius: 4px;  
    cursor: pointer;  
    font-size: 16px;  
    font-weight: 500;  
}  
  
.button-primary:hover {  
    background-color: #087f08;  
}  
  
.flash-messages {  
    max-width: 400px;  
    margin: 20px auto;  
    text-align: center;  
}  
  
.flash-messages p {  
    padding: 10px;  
    border-radius: 4px;  
    margin: 5px 0;  
}  
  
.flash-success {  
    background-color: #e8f5e9;  
    color: #2e7d32;  
}  
  
.flash-error {
```

```
background-color: #ffebee;
color: #d32f2f;
}

"""

templates\admin_users.html
``html
{% extends "base.html" %}
{% block content %}


## Admin: Approve Users


{% for user in users %}
<form method="POST" style="margin-bottom:10px;">
{{ user.username }} {{ user.email }}
<input type="hidden" name="user_id" value="{{ user.id }}">
<button type="submit">Approve as Operator</button>
</form>
{% else %}
<p>No users to approve.</p>
{% endfor %}
{% endblock %}
```

```
"""

templates\base.html
``html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>% block title %}WebRover% endblock %</title>
<link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;500;700&display=swap" rel="stylesheet">
<style>
* {
    box-sizing: border-box;
}

html, body {
    margin: 0;
    padding: 0;
    height: 100%;
}

body {
    font-family: 'Montserrat', sans-serif;
    background-color: #f8f9fa;
    color: #222;
```

```
    overflow-x: hidden;
    position: relative;
}

/* Shared background video style (pages opt-in via block) */
.bg-video {
    position: fixed;
    top: 50%;
    left: 50%;
    min-width: 100%;
    min-height: 100%;
    width: auto;
    height: auto;
    transform: translate(-50%, -50%);
    object-fit: cover;
    z-index: -1;
    filter: brightness(0.55);
}

/* Main content wrapper */
.container {
    max-width: 900px;
    margin: 60px auto;
    padding: 0 24px;
    position: relative;
    z-index: 1;
}

h1 {
    color: #0a8f0a;
    margin-bottom: 8px;
}

p {
    color: #444;
    margin-bottom: 24px;
}

.btn-primary,
.btn-secondary {
    display: inline-block;
    padding: 12px 26px;
    border-radius: 8px;
    border: none;
    cursor: pointer;
    font-weight: 600;
    font-size: 15px;
    text-decoration: none;
```

```
    transition: all 0.25s ease;
}

.btn-primary {
    background-color: #0a8f0a;
    color: #fff;
}

.btn-primary:hover {
    background-color: #087607;
    transform: translateY(-1px);
    box-shadow: 0 6px 14px rgba(0,0,0,0.22);
}

.btn-secondary {
    background-color: transparent;
    color: #0a8f0a;
    border: 2px solid #0a8f0a;
    margin-left: 10px;
}

.btn-secondary:hover {
    background-color: #0a8f0a;
    color: #fff;
    transform: translateY(-1px);
    box-shadow: 0 6px 14px rgba(0,0,0,0.22);
}

.input-field {
    padding: 12px 15px;
    font-size: 16px;
    border: 1px solid #ddd;
    border-radius: 6px;
    outline: none;
    width: 100%;
    margin-bottom: 15px;
}

.input-field:focus {
    border-color: #0a8f0a;
    box-shadow: 0 0 0 2px rgba(10,143,10,0.1);
}

.flash {
    margin: 6px 0 0;
    padding: 8px 12px;
    border-radius: 4px;
    font-size: 13px;
```

```
display: inline-block;
background: rgba(255,255,255,0.9);
}

.flash-success { color: #0a8f0a; border-left: 4px solid #0a8f0a; }
.flash-danger { color: #c0392b; border-left: 4px solid #c0392b; }
.flash-warning { color: #e67e22; border-left: 4px solid #e67e22; }
.flash-info { color: #2980b9; border-left: 4px solid #2980b9; }

table {
width: 100%;
border-collapse: collapse;
margin-bottom: 30px;
}

table, th, td { border: 1px solid #ddd; }
th, td { padding: 12px; text-align: center; }
th { background-color: #f5f5f5; }

a {
text-decoration: none;
color: #0a8f0a;
font-weight: 500;
}

a:hover {
text-decoration: underline;
}

.camera-section {
background: #fff;
padding: 30px;
border-radius: 12px;
box-shadow: 0 10px 25px rgba(0,0,0,0.1);
margin-bottom: 30px;
}

.camera-buttons button { margin: 10px; }

/* Helper layout classes pages can use */
.center-page {
min-height: calc(100vh - 120px);
display: flex;
flex-direction: column;
justify-content: center;
}

</style>
{% block extra_head %}{% endblock %}
```

```

</head>
<body>
    {% block background %}{% endblock %}
    <div class="container">
        {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
                {% for category, message in messages %}
                    <div class="flash flash-{{ category }}>{{ message }}</div>
                {% endfor %}
            {% endif %}
        {% endwith %}

        {% block content %}{% endblock %}
    </div>
</body>
</html>

```

""

```

templates\dashboard_admin.html
"""
html
{% extends "base.html" %}

{% block content %}


<!-- Header -->
    <div class="dashboard-header">
        <h1 class="dashboard-title">Admin Dashboard</h1>
        <a href="{{ url_for('logout') }}" class="btn-logout">Logout</a>
    </div>

    <!-- Flash messages -->
    {% with messages = get_flashed_messages(with_categories=true) %}
        {% if messages %}
            {% set category, message = messages[0] %}
            <div class="flash {{ category }}>{{ message }}</div>
        {% endif %}
    {% endwith %}

    <!-- User Management Table -->
    <div class="user-management">
        <h2>User Management</h2>
        <table>
            <tr>
                <th>Username</th>
                <th>Email</th>
                <th>Role</th>


```

```

<th>Actions</th>
</tr>
{% for user in users %}
<tr>
    <td>{{ user.username }}</td>
    <td>{{ user.email }}</td>
    <td>{{ user.role }}</td>
    <td>
        {% if user.role == 'viewer' %}
        {% if not existing_operator %}
            <a href="{{ url_for('approve_operator', user_id=user.id) }}" class="btn-primary">Promote to Operator</a>
        {% else %}
            <span>Operator exists</span>
        {% endif %}
        {% elif user.role == 'operator' %}
            <a href="{{ url_for('demote_operator', user_id=user.id) }}" class="btn-primary">Demote to Viewer</a>
        {% endif %}
        {% if user.role != 'admin' %}
            <a href="{{ url_for('remove_user', user_id=user.id) }}" class="btn-primary">Remove</a>
        {% endif %}
        {% endif %}
    </td>
</tr>
{% endfor %}
</table>
</div>

<!-- Camera Section --&gt;
&lt;div class="camera-section"&gt;
    &lt;div class="video-wrapper"&gt;
        &lt;video id="video" width="640" height="480" autoplay playsinline&gt;&lt;/video&gt;
        &lt;div id="noCameraMessage"&gt;No Camera Detected&lt;/div&gt;
    &lt;/div&gt;
    &lt;canvas id="snapshot" width="640" height="480" style="display:none;"&gt;&lt;/canvas&gt;

    &lt;div class="camera-buttons"&gt;
        &lt;button id="snapshotBtn" class="btn-primary" disabled&gt;Take Snapshot&lt;/button&gt;
        &lt;button id="startBtn" class="btn-primary" disabled&gt;Start Recording&lt;/button&gt;
        &lt;button id="stopBtn" class="btn-primary" disabled&gt;Stop Recording&lt;/button&gt;
    &lt;/div&gt;
    &lt;div id="timer" class="recording-timer"&gt;00:00&lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;

&lt;script&gt;
let mediaStream = null;
</pre>

```

```

let mediaRecorder;
let recordedChunks = [];
let timerInterval;
let seconds = 0;

// Elements
const video = document.getElementById('video');
const noCam = document.getElementById('noCameraMessage');
const startBtn = document.getElementById('startBtn');
const stopBtn = document.getElementById('stopBtn');
const snapshotBtn = document.getElementById('snapshotBtn');
const timerDisplay = document.getElementById('timer');

// Try to access webcam
navigator.mediaDevices.getUserMedia({ video: true, audio: true })
.then(stream => {
  mediaStream = stream;
  video.srcObject = stream;
  noCam.style.display = "none";
  snapshotBtn.disabled = false;
  startBtn.disabled = false;
})
.catch(() => {
  noCam.style.display = "flex";
  video.style.display = "none";
});

// ----- SNAPSHOT FUNCTION -----
snapshotBtn.addEventListener('click', () => {
  if (!mediaStream) return alert("No camera detected.");
  const canvas = document.getElementById('snapshot');
  const context = canvas.getContext('2d');
  context.drawImage(video, 0, 0, canvas.width, canvas.height);
  const imageURL = canvas.toDataURL("image/png");
  const a = document.createElement('a');
  a.href = imageURL;
  a.download = 'snapshot.png';
  a.click();
  alert("Snapshot saved!");
});

// ----- RECORDING FUNCTIONS -----
startBtn.addEventListener('click', () => {
  if (!mediaStream) return alert("No camera detected.");

  recordedChunks = [];
  mediaRecorder = new MediaRecorder(mediaStream);
  mediaRecorder.ondataavailable = e => {

```

```

        if (e.data.size > 0) recordedChunks.push(e.data);
    };
    mediaRecorder.onstop = saveRecording;

    mediaRecorder.start();
    startBtn.disabled = true;
    stopBtn.disabled = false;
    seconds = 0;
    updateTimer();
    timerInterval = setInterval(updateTimer, 1000);
});

stopBtn.addEventListener('click', () => {
    if (mediaRecorder && mediaRecorder.state !== "inactive") {
        mediaRecorder.stop();
        clearInterval(timerInterval);
        startBtn.disabled = false;
        stopBtn.disabled = true;
        timerDisplay.textContent = "00:00";
    }
});

function saveRecording() {
    const blob = new Blob(recordedChunks, { type: 'video/webm' });
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = 'recording.webm';
    a.click();
    alert("Recording saved!");
}

function updateTimer() {
    seconds++;
    const mins = String(Math.floor(seconds / 60)).padStart(2, '0');
    const secs = String(seconds % 60).padStart(2, '0');
    timerDisplay.textContent = `${mins}:${secs}`;
}
</script>

<style>
.dashboard-header {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 10px 20px;
}

```

```
.dashboard-title {  
    margin: 0;  
    font-size: 28px;  
    font-weight: bold;  
}  
  
.btn-logout {  
    background: #e74c3c;  
    color: white;  
    text-decoration: none;  
    padding: 8px 14px;  
    border-radius: 6px;  
    font-weight: 500;  
    transition: 0.3s;  
}  
.btn-logout:hover {  
    background: #c0392b;  
}  
  
.video-wrapper {  
    position: relative;  
    width: 640px;  
    height: 480px;  
    margin: 20px auto;  
    background: #000;  
    border-radius: 10px;  
    overflow: hidden;  
}  
  
#noCameraMessage {  
    display: none;  
    position: absolute;  
    inset: 0;  
    background: rgba(20, 20, 20, 0.9);  
    color: white;  
    font-size: 28px;  
    display: flex;  
    align-items: center;  
    justify-content: center;  
}  
  
.camera-buttons {  
    text-align: center;  
    margin-top: 10px;  
}  
  
.recording-timer {  
    text-align: center;  
    font-weight: bold;  
    font-size: 20px;  
    margin-top: 10px;
```

```

}

</style>
{% endblock %}

""

templates\dashboard_operator.html
``html
{% extends "base.html" %}

{% block content %}
<div class="dashboard-container full-screen">
    <!-- Top Page Title -->
    <div class="page-title">Web Based Search And Rescue Robot</div>

    <!-- Header with Operator Dashboard and Logout -->
    <div class="dashboard-header">
        <h1 class="dashboard-title-big">Operator Dashboard</h1>
        <a href="{{ url_for('logout') }}" class="btn-logout">Logout</a>
    </div>

    <!-- Single White Box for All Controls -->
    <div class="control-panel">
        <!-- Top Buttons & Timer -->
        <div class="camera-controls">
            <button id="snapshotBtn" class="btn-primary" disabled>Take Snapshot</button>
            <button id="startBtn" class="btn-primary" disabled>Start Recording</button>
            <button id="stopBtn" class="btn-primary" disabled>Stop Recording</button>
            <div id="timer" class="recording-timer">00:00</div>
        </div>

        <!-- Camera Section -->
        <div class="camera-section">
            <div class="video-wrapper">
                <video id="video" autoplay playsinline></video>
                <div id="noCameraMessage">No Camera Detected</div>
            </div>
        </div>

        <!-- Robot Control Section -->
        <div class="robot-controls">
            <h2>Robot Control Panel</h2>
            <div class="control-buttons">
                <div class="row">
                    <button class="btn-primary" onclick="moveRobot('forward')">? Forward</button>
                </div>
                <div class="row">

```

```

        <button class="btn-primary" onclick="moveRobot('left')">? Left</button>
    >
        <button class="btn-primary" onclick="moveRobot('stop')">? Stop</button>
    >
        <button class="btn-primary" onclick="moveRobot('right')">? Right</button>
    on>
        </div>
        <div class="row">
            <button class="btn-primary" onclick="moveRobot('backward')">? Backward
        </button>
        </div>
        <p class="hint">(Use arrow keys to control robot ? Space to stop)</p>
        </div>
        </div>
    </div>
</div>

<script>
// ----- CAMERA SETUP -----
let mediaStream = null;
let mediaRecorder;
let recordedChunks = [];
let timerInterval;
let seconds = 0;

const video = document.getElementById('video');
const noCam = document.getElementById('noCameraMessage');
const startBtn = document.getElementById('startBtn');
const stopBtn = document.getElementById('stopBtn');
const snapshotBtn = document.getElementById('snapshotBtn');
const timerDisplay = document.getElementById('timer');

navigator.mediaDevices.getUserMedia({ video: true, audio: true })
.then(stream => {
    mediaStream = stream;
    video.srcObject = stream;
    noCam.style.display = "none";
    snapshotBtn.disabled = false;
    startBtn.disabled = false;
})
.catch(() => {
    noCam.style.display = "flex";
    video.style.display = "none";
});

// ----- SNAPSHOT -----
snapshotBtn.addEventListener('click', () => {
    if (!mediaStream) return alert("No camera detected.");

```

```

const canvas = document.createElement('canvas');
canvas.width = video.videoWidth;
canvas.height = video.videoHeight;
const context = canvas.getContext('2d');
context.drawImage(video, 0, 0, canvas.width, canvas.height);
const imageURL = canvas.toDataURL("image/png");
const a = document.createElement('a');
a.href = imageURL;
a.download = 'snapshot.png';
a.click();
});

// ----- RECORDING -----
startBtn.addEventListener('click', () => {
  if (!mediaStream) return alert("No camera detected.");
  recordedChunks = [];
  mediaRecorder = new MediaRecorder(mediaStream);
  mediaRecorder.ondataavailable = e => { if (e.data.size > 0) recordedChunks.push(e.data) };
  mediaRecorder.onstop = saveRecording;
  mediaRecorder.start();
  startBtn.disabled = true;
  stopBtn.disabled = false;
  seconds = 0;
  updateTimer();
  timerInterval = setInterval(updateTimer, 1000);
});

stopBtn.addEventListener('click', () => {
  if (mediaRecorder && mediaRecorder.state !== "inactive") {
    mediaRecorder.stop();
    clearInterval(timerInterval);
    startBtn.disabled = false;
    stopBtn.disabled = true;
    timerDisplay.textContent = "00:00";
  }
});

function saveRecording() {
  const blob = new Blob(recordedChunks, { type: 'video/webm' });
  const url = URL.createObjectURL(blob);
  const a = document.createElement('a');
  a.href = url;
  a.download = 'recording.webm';
  a.click();
}

function updateTimer() {

```

```

seconds++;
const mins = String(Math.floor(seconds / 60)).padStart(2, '0');
const secs = String(seconds % 60).padStart(2, '0');
timerDisplay.textContent = '${mins}:${secs}';

}

// Robot control
function moveRobot(direction) {
  fetch(`/move/${direction}`, { method: 'POST' })
    .then(r => r.text())
    .then(console.log)
    .catch(err => console.error("Robot control error:", err));
}

// Keyboard controls
document.addEventListener('keydown', e => {
  switch (e.key) {
    case 'ArrowUp': moveRobot('forward'); break;
    case 'ArrowDown': moveRobot('backward'); break;
    case 'ArrowLeft': moveRobot('left'); break;
    case 'ArrowRight': moveRobot('right'); break;
    case ' ': moveRobot('stop'); break;
  }
});
</script>

<style>
html, body { margin:0; padding:0; height:100%; width:100%; }

.page-title {
  text-align: center;
  font-size: 28px;
  font-weight: 700;
  color: #0a8f0a;
  margin: 10px 0;
}

.dashboard-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 0 20px 10px 20px;
}
.dashboard-title-big { font-size: 28px; font-weight: bold; margin: 0; }

.btn-logout {
  background: #e74c3c;
  color: white;
}

```

```
text-decoration: none;
padding: 6px 12px;
border-radius: 6px;
transition: 0.3s;
}
.btn-logout:hover { background-color: #c0392b; }

.control-panel {
  width: 100%;
  height: calc(100vh - 80px);
  background: #fff;
  padding: 15px;
  box-sizing: border-box;
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
  align-items: stretch;
  border-radius: 12px;
  box-shadow: 0 10px 25px rgba(0,0,0,0.1);
}

.camera-controls {
  display: flex;
  justify-content: center; /* centered */
  align-items: center;
  gap: 15px;
  margin-bottom: 10px;
}

.recording-timer {
  font-weight:bold;
  font-size:18px;
}

.camera-section {
  flex: 1;
  display: flex;
  justify-content: center;
  align-items: center;
  margin: 10px 0;
}

/* 16:9 Aspect Ratio for Camera */
.video-wrapper {
  width: 100%;
  max-width: 1400px;
  aspect-ratio: 16 / 9;
  background:#000;
  border-radius:12px;
```

```

        overflow:hidden;
        position:relative;
    }
    video { width:100%; height:100%; object-fit:cover; }
    #noCameraMessage {
        display:none; position:absolute; inset:0; background:rgba(30,30,30,0.9);
        color:white; font-size:26px; display:flex; align-items:center; justify-content:center;
    }

    .robot-controls {
        background:#f8f8f8; padding:15px; border-radius:12px; box-shadow:0 8px 20px rgba(0,0,0,0.05);
        text-align:center;
    }
    .robot-controls h2 { margin-bottom:12px; }
    .control-buttons { display:flex; flex-direction:column; gap:10px; }
    .control-buttons .row { display:flex; justify-content:center; gap:10px; }
    .hint { font-size:14px; color:#555; }

    .btn-primary { background-color:#0a8f0a; color:white; padding:10px 16px; border:none; border-radius:6px; cursor:pointer; font-size:16px; }
    .btn-primary:hover { background-color:#087607; }

    @media(max-width:768px){
        .page-title { font-size:24px; }
        .dashboard-title-big { font-size:22px; }
        .btn-primary { font-size:14px; padding:8px 12px; }
    }

```

</style>

{% endblock %}

""

templates\dashboard\_viewer.html

```

<html
{% extends "base.html" %}
{% block content %}
<div class="dashboard-container">
    <div class="dashboard-header">
        <h1 class="dashboard-title">Viewer Dashboard</h1>
        <div class="dashboard-actions">
            <a href="{{ url_for('logout') }}" class="btn-logout">Logout</a>
        </div>
    </div>

    <p class="dashboard-subtitle">View the camera feed and take snapshots or recordings</p>

```

```
<div class="camera-section">
  <video id="video" width="640" height="480" autoplay></video>
  <canvas id="snapshot" width="640" height="480" style="display:none;"></canvas>
  <div class="camera-buttons">
    <button onclick="takeSnapshot()" class="btn-primary">Take Snapshot</button>
    <button onclick="startRecording()" class="btn-primary">Start Recording</button>
  >
    <button onclick="stopRecording()" class="btn-primary">Stop Recording</button>
  </div>
</div>
</div>

<script>
const video = document.getElementById('video');
navigator.mediaDevices.getUserMedia({ video: true, audio: true })
.then(stream => video.srcObject = stream)
.catch(err => console.error("Camera error:", err));

function takeSnapshot() {
  const canvas = document.getElementById('snapshot');
  const context = canvas.getContext('2d');
  context.drawImage(video, 0, 0, canvas.width, canvas.height);
  alert("Snapshot taken!");
}

let mediaRecorder;
let recordedChunks = [];

function startRecording() {
  recordedChunks = [];
  mediaRecorder = new MediaRecorder(video.srcObject);
  mediaRecorder.ondataavailable = e => { if (e.data.size > 0) recordedChunks.push(e.data) };
  mediaRecorder.start();
  alert("Recording started");
}

function stopRecording() {
  mediaRecorder.stop();
  mediaRecorder.onstop = () => {
    const blob = new Blob(recordedChunks, { type: 'video/webm' });
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = 'recording.webm';
    a.click();
    alert("Recording saved");
  };
}
```

```
}

</script>

<style>
.dashboard-container {
    max-width: 900px;
    margin: 50px auto;
    text-align: center;
}

.dashboard-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 15px;
}

.dashboard-title {
    font-size: 32px;
    color: #0a8f0a;
    font-weight: 700;
    margin: 0;
}

.dashboard-subtitle {
    font-size: 16px;
    color: #666;
    margin-bottom: 30px;
}

.btn-primary {
    padding: 8px 15px;
    background-color: #0a8f0a;
    color: white;
    border: none;
    border-radius: 6px;
    cursor: pointer;
}
.btn-primary:hover {
    background-color: #087607;
}

.btn-logout {
    background-color: #d9534f;
    color: white;
    padding: 8px 15px;
    border-radius: 6px;
    text-decoration: none;
}
```

```

    font-weight: 600;
    transition: background 0.3s;
}
.btn-logout:hover {
    background-color: #c9302c;
}

.camera-section {
    background: #fff;
    padding: 30px;
    border-radius: 12px;
    box-shadow: 0 10px 25px rgba(0,0,0,0.1);
}

.camera-buttons button {
    margin: 10px;
}
</style>
{% endblock %}

"""

templates\index.html
``html
{% extends "base.html" %}

{% block title %}WebRover Controller{% endblock %}

{% block background %}
<video class="bg-video" autoplay muted loop playsinline>
    <source src="{{ url_for('static', filename='stock/base.mp4') }}" type="video/mp4">
</video>
{% endblock %}

{% block extra_head %}
<style>
.hero-wrap {
    min-height: calc(100vh - 120px);
    display: flex;
    flex-direction: column;
    justify-content: center;
}
/* Title */
.hero-title {
    font-size: 54px;
    font-weight: 700;
    color: #ffffff;

```

```
margin-bottom: 16px;
text-shadow: 0 6px 24px rgba(0,0,0,0.55);
}

/* Description ? crisp, not blurry */
.hero-desc {
    font-size: 20px;
    color: #ffffff;
    max-width: 720px;
    line-height: 1.6;
    margin-bottom: 32px;
    text-shadow: 0 2px 6px rgba(0,0,0,0.4);
}

/* Buttons row */
.hero-buttons {
    display: flex;
    flex-wrap: wrap;
    gap: 18px;
}

/* Both Login & Register: same green style */
.hero-buttons .btn-main {
    display: inline-block;
    padding: 14px 40px;
    border-radius: 18px;
    border: none;
    cursor: pointer;
    font-weight: 700;
    font-size: 22px;
    background-color: #0a8f0a;
    color: #ffffff;
    box-shadow: 0 10px 28px rgba(0,0,0,0.35);
    transition: all 0.2s ease;
}

.hero-buttons .btn-main:hover {
    background-color: #0a7c0a;
    transform: translateY(-2px);
    box-shadow: 0 14px 34px rgba(0,0,0,0.45);
}

</style>
{% endblock %}

{% block content %}
<div class="hero-wrap">
    <h1 class="hero-title">WebRover Controller</h1>
    <p class="hero-desc">
```

Securely control your Raspberry Pi rover from any browser. Assign roles for administrators, operators, and viewers, stream live video, and send movement commands in real time

```
</p>

<div class="hero-buttons">
    <a href="{{ url_for('login') }}" class="btn-main">Login</a>
    <a href="{{ url_for('register') }}" class="btn-main">Register</a>
</div>
</div>
{% endblock %}
```

“

```
templates\login.html
``html
{% extends "base.html" %}

{% block title %}Login • WebRover{% endblock %}

{% block background %}
<video class="bg-video" autoplay muted loop playsinline>
    <source src="{{ url_for('static', filename='stock/access.mp4') }}" type="video/mp4">
</video>
{% endblock %}

{% block extra_head %}
<style>
.auth-wrapper {
    min-height: calc(100vh - 120px);
    display: flex;
    justify-content: center;
    align-items: center;
}
</style>
/* Card: white with soft shadow, green accent */
.auth-card {
    background: rgba(255, 255, 255, 0.97);
    border-radius: 24px;
    padding: 34px 30px 26px;
    width: 100%;
    max-width: 430px;
    text-align: center;
    box-shadow: 0 18px 48px rgba(0,0,0,0.25);
    border: 1px solid rgba(10, 143, 10, 0.18);
    color: #222;
```

```
}

/* Title in green like landing page */
.auth-title {
    font-size: 30px;
    font-weight: 700;
    color: #0a8f0a;
    margin-bottom: 8px;
    text-shadow: 0 2px 6px rgba(0,0,0,0.18);
}

.auth-subtitle {
    font-size: 14px;
    color: #555;
    margin-bottom: 22px;
}

/* Inputs */
.auth-form .input-field {
    background: #ffffff;
    border: 1px solid #ddd;
    color: #222;
}

.auth-form .input-field::placeholder {
    color: #999;
}

.auth-form .input-field:focus {
    border-color: #0a8f0a;
    box-shadow: 0 0 0 2px rgba(10,143,10,0.08);
}

/* Full-width green button */
.auth-form .btn-primary {
    width: 100%;
    margin-top: 8px;
    background-color: #0a8f0a;
    color: #ffffff;
    font-size: 17px;
    border-radius: 14px;
    padding: 12px 0;
    box-shadow: 0 10px 24px rgba(0,0,0,0.25);
}

.auth-form .btn-primary:hover {
    background-color: #0a7c0a;
    transform: translateY(-1px);
```

```
        box-shadow: 0 14px 30px rgba(0,0,0,0.3);  
    }  
  
/* Switch text */  
.switch-text {  
    margin-top: 14px;  
    font-size: 13px;  
    color: #666;  
}  
  
.switch-text a {  
    color: #0a8f0a;  
    font-weight: 600;  
}  
</style>  
{% endblock %}  
  
{% block content %}  
<div class="auth-wrapper">  
    <div class="auth-card">  
        <h1 class="auth-title">WebRover Controller</h1>  
        <p class="auth-subtitle">Login to control or monitor your rover.</p>  
  
        <form method="POST" class="auth-form">  
            <input type="text" name="username" placeholder="Username" required class="input-field">  
            <input type="password" name="password" placeholder="Password" required class="input-field">  
            <button type="submit" class="btn-primary">Login</button>  
        </form>  
  
        <p class="switch-text">  
            Don't have an account?  
            <a href="{{ url_for('register') }}>Register</a>  
        </p>  
    </div>  
</div>  
{% endblock %}
```

66

templates\register.html

““html

```
{% extends "base.html" %}
```

{% block title %}Register • WebRover{% endblock %}

```
{% block background %}  
<video class="bg-video" autoplay muted loop playsinline>  
  <source src="{{ url_for('static', filename='stock/access.mp4') }}" type="video/mp4">  
</video>  
{% endblock %}  
  
{% block extra_head %}  
<style>  
.auth-wrapper {  
  min-height: calc(100vh - 120px);  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}  
  
.auth-card {  
  background: rgba(255, 255, 255, 0.97);  
  border-radius: 24px;  
  padding: 34px 30px 24px;  
  width: 100%;  
  max-width: 460px;  
  text-align: center;  
  box-shadow: 0 18px 48px rgba(0,0,0,0.25);  
  border: 1px solid rgba(10, 143, 10, 0.18);  
  color: #222;  
}  
  
.auth-title {  
  font-size: 30px;  
  font-weight: 700;  
  color: #0a8f0a;  
  margin-bottom: 8px;  
  text-shadow: 0 2px 6px rgba(0,0,0,0.18);  
}  
  
.auth-subtitle {  
  font-size: 14px;  
  color: #555;  
  margin-bottom: 20px;  
}  
  
.auth-form .input-field {  
  background: #ffffff;  
  border: 1px solid #ddd;  
  color: #222;  
}  
  
.auth-form .input-field::placeholder {
```

```
color: #999;
}

.auth-form .input-field:focus {
    border-color: #0a8f0a;
    box-shadow: 0 0 0 2px rgba(10,143,10,0.08);
}

.auth-form .btn-primary {
    width: 100%;
    margin-top: 8px;
    background-color: #0a8f0a;
    color: #ffffff;
    font-size: 17px;
    border-radius: 14px;
    padding: 12px 0;
    box-shadow: 0 10px 24px rgba(0,0,0,0.25);
}

.auth-form .btn-primary:hover {
    background-color: #0a7c0a;
    transform: translateY(-1px);
    box-shadow: 0 14px 30px rgba(0,0,0,0.3);
}

.switch-text {
    margin-top: 10px;
    font-size: 13px;
    color: #666;
}

.switch-text a {
    color: #0a8f0a;
    font-weight: 600;
}
</style>
{% endblock %}

{% block content %}
<div class="auth-wrapper">
    <div class="auth-card">
        <h1 class="auth-title">Create your WebRover account</h1>
        <p class="auth-subtitle">
            Register as a viewer first. An admin can promote you to operator when you're ready.
        </p>
        <form method="POST" class="auth-form">
```

```

<input type="text" name="username" placeholder="Username" required class="input-field">
    <input type="email" name="email" placeholder="Email" required class="input-field">
        <input type="password" name="password" placeholder="Password" required class="input-field">
            <button type="submit" class="btn-primary">Register</button>
        </form>

        <p class="switch-text">
            Already have an account?
            <a href="{{ url_for('login') }}>Login</a>
        </p>
    </div>
</div>
{% endblock %}

```

""

```

templates\verify_email.html
``html
{% extends "base.html" %}
{% block content %}
<div class="login-container">
    <div class="login-card">
        <h1 class="login-title">Verify Email</h1>
        <p class="login-subtitle">Enter the verification code sent to your email</p>
        <form method="POST" class="login-form">
            <input type="text" name="username" placeholder="Username" value="{{ username }}" readonly class="input-field">
            <input type="text" name="code" placeholder="Verification Code" required class="input-field">
            <button type="submit" class="btn-primary">Verify</button>
        </form>
        <p class="register-text">
            Didn't receive the code? <a href="{{ url_for('register') }}" class="btn-register">Resend</a>
        </p>
    </div>
</div>

<style>
body {
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;

```

```
background-color: #f8f9fa;
font-family: 'Montserrat', sans-serif;
margin: 0;
}
.login-container {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 100%;
  padding: 20px;
}
.login-card {
  background-color: #fff;
  padding: 40px 30px;
  border-radius: 12px;
  box-shadow: 0 10px 25px rgba(0,0,0,0.1);
  width: 100%;
  max-width: 400px;
  text-align: center;
}
.login-title {
  font-size: 32px;
  color: #0a8f0a;
  margin-bottom: 8px;
  font-weight: 700;
}
.login-subtitle {
  font-size: 16px;
  color: #666;
  margin-bottom: 30px;
}
.login-form {
  display: flex;
  flex-direction: column;
  gap: 15px;
}
.input-field {
  padding: 12px 15px;
  font-size: 16px;
  border: 1px solid #ddd;
  border-radius: 6px;
  outline: none;
  transition: border-color 0.3s;
}
.input-field:focus {
  border-color: #0a8f0a;
}
.btn-primary {
```

```
padding: 12px 0;
background-color: #0a8f0a;
color: #fff;
border: none;
border-radius: 6px;
font-size: 16px;
cursor: pointer;
font-weight: 500;
transition: background-color 0.3s;
}
.btn-primary:hover {
background-color: #087607;
}
.register-text {
margin-top: 20px;
font-size: 14px;
color: #555;
}
.btn-register {
color: #0a8f0a;
text-decoration: none;
font-weight: 500;
margin-left: 5px;
}
.btn-register:hover {
text-decoration: underline;
}
</style>
{% endblock %}
```

""