issue_tracker.py

```python
import streamlit as st
import pandas as pd

# --- Initialize session state ---
if "issues" not in st.session_state:
    st.session_state.issues = pd.DataFrame(
        columns=["ID", "Category", "Location", "Priority", "Status", "Assigned To", "Comments"]
    )

# --- Sidebar Menu ---
st.sidebar.title("Admin Portal")
menu = st.sidebar.radio("Navigation", ["Dashboard", "Add Issue", "Manage Issues", "Import/Export"])

# --- Dashboard ---
if menu == "Dashboard":
    st.title("📊 Issue Dashboard")
    if st.session_state.issues.empty:
        st.info("No issues logged yet.")
    else:
        st.dataframe(st.session_state.issues)

        # Summary counts
        st.subheader("Summary")
        col1, col2, col3 = st.columns(3)
        col1.metric("Total Issues", len(st.session_state.issues))
        col2.metric("Open Issues", (st.session_state.issues["Status"] == "Open").sum())
        col3.metric("Closed Issues", (st.session_state.issues["Status"] == "Closed").sum())

# --- Add Issue ---
elif menu == "Add Issue":
    st.title("➕ Add New Issue")
    with st.form("issue_form"):
        category = st.selectbox("Category", ["Road", "Water", "Electricity", "Waste", "Other"])
        location = st.text_input("Location")
```

```
        priority = st.selectbox("Priority", ["Low", "Medium", "High"])
        submit = st.form_submit_button("Add Issue")

    if submit:
        new_id = len(st.session_state.issues) + 1
        new_issue = {
            "ID": new_id,
            "Category": category,
            "Location": location,
            "Priority": priority,
            "Status": "Open",
            "Assigned To": "",
            "Comments": ""
        }
        st.session_state.issues = pd.concat(
            [st.session_state.issues, pd.DataFrame([new_issue])],
            ignore_index=True
        )
        st.success("✅ Issue added successfully!")

# --- Manage Issues ---
elif menu == "Manage Issues":
    st.title("🛠 Manage Issues")
    if st.session_state.issues.empty:
        st.info("No issues to manage.")
    else:
        # Filtering
        st.subheader("Filters")
        col1, col2, col3 = st.columns(3)
        f_category = col1.selectbox("Category", ["All"] + st.session_state.issues["Category"].unique().tolist())
        f_location = col2.text_input("Location filter")
        f_priority = col3.selectbox("Priority", ["All"] + st.session_state.issues["Priority"].unique().tolist())

        filtered = st.session_state.issues.copy()
        if f_category != "All":
            filtered = filtered[filtered["Category"] == f_category]
        if f_location:
```

```
            filtered = filtered[filtered["Location"].str.contains(f_location, case=False)]
        if f_priority != "All":
            filtered = filtered[filtered["Priority"] == f_priority]

        st.dataframe(filtered)

        # Select Issue
        issue_id = st.number_input("Enter Issue ID to update", min_value=1, step=1)
        issue = st.session_state.issues[st.session_state.issues["ID"] == issue_id]

        if not issue.empty:
            st.subheader("Update Issue")
            assigned = st.text_input("Assign To", issue["Assigned To"].values[0])
            status = st.selectbox("Status", ["Open", "In Progress", "Closed"], index=["Open", "In Progress",
"Closed"].index(issue["Status"].values[0]))
            comments = st.text_area("Comments", issue["Comments"].values[0])
            if st.button("Update Issue"):
                st.session_state.issues.loc[st.session_state.issues["ID"] == issue_id, ["Assigned To", "Status", "Comments"]] =
[assigned, status, comments]
                st.success("✅ Issue updated successfully!")
                st.experimental_rerun()

# --- Import/Export ---
elif menu == "Import/Export":
    st.title("⬆⬇ Import / Export Issues")

    # Export
    st.subheader("Export Issues")
    if not st.session_state.issues.empty:
        csv = st.session_state.issues.to_csv(index=False).encode("utf-8")
        st.download_button("Download CSV", csv, "issues.csv", "text/csv")

    # Import
    st.subheader("Import Issues")
    uploaded = st.file_uploader("Upload CSV", type=["csv"])
    if uploaded:
        try:
```

```
            df = pd.read_csv(uploaded)
            if all(col in df.columns for col in st.session_state.issues.columns):
                st.session_state.issues = df
                st.success("✅ Issues imported successfully!")
                st.experimental_rerun()
            else:
                st.error("CSV format does not match required structure.")
        except Exception as e:
            st.error(f"Import failed: {e}")

# --- Footer ---
st.caption("💡 Tip: Use Streamlit secrets for secure password storage in production.")
```