Edge Case and Error Handling Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Case | Description | Input Columns | Expected Behavior | Assertion / Check |
| TC\_E1 | Missing Columns | Checks if any required column is missing from the input | timestamp, position\_x, position\_y | Raises error or logs missing columns | 'position\_y' in df.columns |
| TC\_E2 | Malformed Data Types | Validates that columns have correct data types | timestamp, position\_x, position\_y | Raises ValueError if types can't be converted | pd.to\_numeric(df['timestamp']) with pytest.raises(ValueError) |
| TC\_E3 | Null / NaN Values | Verifies if any value in the dataset is null or NaN | timestamp, position\_x, position\_y | Nulls are detected and handled (e.g., dropped or imputed) | df.isnull().values.any() |
| TC\_E4 | Empty Input | Tests if the DataFrame is completely empty | None; DataFrame empty | Should be identified as empty and handled gracefully | assert df.empty |
| TC\_E5 | Out-of-Bounds Values | Checks if position values are within expected range (e.g., [0, 1]) | position\_x, position\_y | Detects values outside allowed range | df['position\_x'].between(0, 1).all() == False |
| TC\_E6 | Unexpected Extra Columns | Validates presence of only expected columns | timestamp, position\_x, position\_y | Warns or ignores unexpected columns | set(df.columns).issubset(expected\_columns) |

# Standard API Functional Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output | Actual Output | Result |
| TC01 | Test API root | GET `/` | {"message": "Urbanization Shift Detection API is live 🚀"} | Same | Pass |
| TC02 | Fetch menu items | GET `/menu` | List of 4 menu items | Same | Pass |
| TC03 | Urban trends analysis | GET `/trends` | JSON with feature, description, and list of yearly average\_population, green\_cover\_percentage | Same (if CSV is present) | Pass / Fail (depends on file) |
| TC04 | Top shift trends | GET `/top-shift-trends` | JSON with top 5 places, shift counts, and yearly trend data | Same (if CSV has 'label' column) | Pass / Fail |
| TC05 | Gender diff trends | GET `/gender-diff-trends` | Yearly average male, female population, and their difference | Same | Pass |
| TC06 | Urban shift prediction | POST `/predict` with valid JSON | JSON with status, confidence\_percent, retraining | Same (if model and encoders are present) | Pass / Fail |
| TC07 | Stats summary | GET `/stats` | JSON with shift statistics and model confidence | Same (if model and CSV exist) | Pass / Fail |
| TC08 | Urbanization vs road density | GET `/urbanization-vs-road-density` | Year-wise road density and shift % | Same | Pass |
| TC09 | Nighttime light trends | GET `/nighttime-intensity-trends` | Year-wise average light intensity | Same | Pass |
| TC10 | Summary profile | GET `/summary-profile` | JSON with average values for numerical & most frequent categorical | Same | Pass |
| TC11 | Slum area trends | GET `/slum-area-proportion-trends` | Year & zone-wise slum area proportions | Same | Pass |
| TC12 | Land use trends | GET `/land-use-change-trends` | Year, zone, land use frequency data | Same | Pass |
| TC13 | Policy insights | GET `/policy-insights` | Insight strings based on trends | Same | Pass |

**✅ Urbanization Shift Detected (Example Input)**

{

"population\_density": 3807.947177,

"green\_cover\_percentage": 18.513293,

"road\_density": 2.617057,

"nighttime\_light\_intensity": 171.539264,

"water\_bodies\_nearby": 1,

"Male\_Count": 4434,

"Female\_Count": 3836,

"Year": 2020,

"Place\_Name": "Bangalore",

"Land\_Use\_Type": "Industrial",

"Slum\_Area\_Proportion": 15.70,

"Zoning\_Code": "Metro"

}

**❌ No Significant Urbanization Shift (Example Input)**

{

"population\_density": 7346.740024,

"green\_cover\_percentage": 87.294584,

"road\_density": 9.062546,

"nighttime\_light\_intensity": 63.869314,

"water\_bodies\_nearby": 1,

"Male\_Count": 3114,

"Female\_Count": 3276,

"Year": 2019,

"Place\_Name": "Kanpur",

"Land\_Use\_Type": "Residential",

"Slum\_Area\_Proportion": 54.38,

"Zoning\_Code": "Tier 2"

}

These inputs are directly mapped from the CSV file rows corresponding to urban\_shift = 1 and 0 respectively.

Would you like me to include these as test cases in the same DOCX file as before?