Algorithm 1 Detect UI Tarpit

```
function DetectUITarpit(xml_1, xml_2, threshold)
    similarity ← CompareXML(xml_1, xml_2)
    if similarity > 90 then
        sim_count ← sim_count + 1
        if sim_count ≥ threshold then
            sim_count ← 0
            return True
        end if
    end if
    return False
end function
```

Algorithm 2 Compare XML

```
function CompareXML(xml_1, xml_2)
    tree_1, tree_2 ← Simplify xml_1, xml_2 and construct trees
    score, total ← CompareTree(tree_1, tree_2)
    return 100.0 if total = 0 else (score/total) × 100
end function
```

Algorithm 3 Main Exploration Loop

```
function START(input_manager)
    count ← 0
    while count < max_event_count do
        Update UI state and snapshots
        Start the APP if essential
        if LLM mode is active then
            event ← GENERATELLMEVENT
        else if DETECTUITARPIT(last_state, current_state) then
            Activate LLM Mode
            event ← GENERATELLMEVENT
        else
            event ← GENERATERANDOMEVENT
        end if
        EXECUTE(event)
        count ← count + 1
    end while
    Clean up and exit
end function
```

Algorithm 4 Generate LLM Event

```
function GENERATELLMEVENT
    if Continuing LLM Sequence then
        Build Next Action Prompt
        response ← CALLLLM
        response ← VALIDITEBYLLM
        act ← PARSEACTION(response)
    else
        Build Meaning Prompt
        r_1 ← CALLLLM
        Build Task Prompt
        r_2 ← CALLLLM
        Build First Action Prompt
        r_3 ← CALLLLM
        response ← VALIDITEBYLLM
        act ← PARSEACTION(response)
    end if
    Set LLM Mode to act.hasNext
    return WRAPASU2EVENT(act)
end function
```

Algorithm 5 Frequency-Aware Random Exploration Strategy

```
1: function GenerateEvent
2:     s ← current state
3:     if s ∉ input_table then
4:         possible_events ← GetPossibleInputs(s)
5:         Initialize input_table[s] with an empty events list
6:         for all event ∈ possible_events do
7:             Add event to input_table[s].events
8:             if event ∉ event_table then
9:                 event_table[event] ← 0
10:            end if
11:        end for
12:    end if
13:    counts ← ∅
14:    for all event ∈ input_table[s].events do
15:        counts[event] ← event_table[event].tried
16:    end for
17:    weights ← GetWeights(input_table[s].events, counts)
18:    selected_event ← randomly select one event from the list using weights
19:    Increment event_table[selected_event].tried
20:    return selected_event
21: end function
```