# High-Level Design & Implementation Document

## Objective

We need to generate a **report of uptime and downtime** for each store. The report schema is:

store\_id, uptime\_last\_hour (minutes), uptime\_last\_day (hours), uptime\_last\_week (hours),  
 downtime\_last\_hour (minutes), downtime\_last\_day (hours), downtime\_last\_week (hours)

### Key Requirements:

1. **Uptime/Downtime limited to business hours**.
2. **Extrapolate observations** to fill the entire interval:
   * Data comes from periodic polls.
   * Example:
     + Business hours: 9 AM – 12 PM
     + Observations: 10:14 AM (active) and 11:15 AM (inactive)
     + We infer uptime from 10:14 – 11:15 and downtime from 11:15 – 12:00. We also backfill status from first observation.

## High-Level Workflow

### Step 1: Fetch Store Metadata

* get\_store\_details fetches:
  + Store timezone.
  + Store business hours (fills defaults for missing days).
* If missing → defaults applied:
  + Timezone: America/Chicago.
  + Business hours: 00:00 – 23:59 daily.

**Output** → (store\_dict, timezone\_str, business\_hours\_dict)

### Step 2: Fetch Observations

* get\_range\_status pulls last **1 week** of observations from StoreUpdates.
* Converts timestamps to UTC.
* Sorts by timestamp\_utc.

**Output** → Pandas DataFrame:

| timestamp\_utc | status |  
|---------------|--------|  
| 2025-08-20T10:14Z | active |  
| 2025-08-20T11:15Z | inactive|

### Step 3: Convert Observations → Status Intervals

* build\_status\_intervals transforms point events into intervals:
  + Example:
    - Input:
      * 10:14 active
      * 11:15 inactive
    - Output:
      * [10:14, 11:15) → active
      * [11:15, now\_utc) → inactive

**Reasoning:** Observations are sparse → treat them as **step functions**.

### Step 4: Convert Business Hours → UTC Intervals

* build\_bh\_intervals\_utc converts business hours to UTC aligned intervals.
* Handles overnight shifts:
  + Example: 22:00 – 02:00 → split across two days.
* Clips intervals within requested [window\_start, now\_utc].

**Output Example:**

[(2025-08-20 14:30 UTC, 2025-08-20 17:30 UTC), ...]

### Step 5: Intersect Status & Business Hours

* compute\_uptime\_downtime:
  1. Build **status intervals** from Step 3.
  2. Build **business hour intervals** from Step 4.
  3. Intersect them to calculate durations:
     + If interval status = active → add to uptime.
     + Else → add to downtime.

**Example:**

Business Hours: [09:00–12:00]  
Status Intervals: [10:14–11:15 active], [11:15–12:00 inactive]  
Uptime = 61 mins, Downtime = 119 mins

### Step 6: Summarize per Window

* Windows: last **hour**, last **day**, last **week**.
* summarize\_results converts seconds → minutes/hours:
  + Hour window → minutes.
  + Day & Week → hours.

**Example Output:**

{"uptime\_last\_hour\_min": 15,  
 "downtime\_last\_hour\_min": 45,  
 "uptime\_last\_day\_hr": 12.5,  
 "downtime\_last\_day\_hr": 11.5}

### Step 7: Final Report

* final\_report loops through all stores:
  + Runs steps 1–6.
  + Collects results.
* Saves as CSV → {report\_id}.csv in result\_data/.

**Example CSV:**

store\_id,uptime\_last\_hour,downtime\_last\_hour,uptime\_last\_day,downtime\_last\_day,uptime\_last\_week,downtime\_last\_week  
1234,15,45,12.5,11.5,78.0,90.0

## Visual Overview

### 🔹 Observations → Status Intervals

Obs Data: ●----active----●----inactive----●  
 10:14 11:15 now  
  
Intervals: [10:14,11:15)=active  
 [11:15,now)=inactive

### 🔹 Business Hours → UTC

Local: Mon 09:00–12:00 CST  
UTC: Mon 15:00–18:00 UTC

### 🔹 Intersect Status with Business Hours

Business Hours: [09:00–12:00]  
Status Intervals: [10:14–11:15 active], [11:15–12:00 inactive]  
Result: Active=61 mins, Inactive=119 mins

## Summary

Implemented **end-to-end report generation**: - Robust DB queries with error handling. - Defaults for missing metadata. - Step-function extrapolation of sparse observations. - Conversion of business hours into UTC. - Precise interval intersection. - Window-based aggregation (hour/day/week). - CSV output for user consumption.

This ensures **accurate uptime/downtime reporting** that respects store-specific business hours and extrapolates reasonably from sparse observations.