Arun Srinivasan

81645

Business Logic

Core Business Rules

1. Release Window Constraint

- Each content item has a defined release window (e.g., Day 1–3).
- Content can only be scheduled within this window.
- If violated → reject the item or skip it gracefully.

2. Max Runs Constraint

- Each content has a weekly cap on how many times it can be featured.
- Once `max_runs` is exhausted → item becomes ineligible.

3. Slot Limit Per Day

- Each day has exactly 10 homepage slots.
- Scheduler must fill all 10 slots per day.

Adjustment Rules (Boosts & Penalties)

4. Kids Time Bonus

- If content genre is "Kids" and scheduled between 9:00–15:00 → +10 engagement.
- Engagement is clamped to a max of 100.

5. PPV Prime-Time Bonus

- If monetization is PPV and scheduled between 19:00–22:00 → revenue × 1.25.

6. Weekend Multiplier

- If day is Saturday or Sunday:
    - Engagement +8 for all content.
    - If monetization is PPV → revenue × 1.15 (can stack with prime-time bonus).

Catalog Diversity Rule

7. Genre Cap Per Day

- No more than 4 items of the same genre per day.
- Two approaches:
    - **Hard rule**: block scheduling beyond 4.
    - **Soft rule**: apply engagement −8 penalty if exceeded.
- Your sample report uses the soft penalty approach.

Scoring Strategies

8. Greedy by Revenue

- Sort content by highest adjusted revenue.

9. Greedy by Engagement

- Sort content by highest adjusted engagement.

10. Balanced Strategy

- Weighted score = 0.6 × Revenue + 0.4 × Engagement.
- Used to balance monetization and viewer interest.

Rule Application Order

1. Filter by release window and max runs.
2. Apply time-based bonuses (Kids, Prime-time).
3. Apply weekend multipliers.
4. Apply genre diversity penalty if needed.
5. Compute final score based on selected strategy.

Reporting Requirements

- Per day:
    - Scheduled content IDs
    - Total revenue and engagement
    - Notes on boosts/penalties applied
- Weekly totals
- Missed content (due to max runs or window violations)

Code:

Schduler.py

```python
from abc import ABC, abstractmethod
from model.content import Content
from context.rule_context import RuleContext
from scoring.scoring import Scoring

class Scheduler(ABC):
    @abstractmethod
    def run(self) -> None:
        pass

class GreedyScheduler(Scheduler):
    def __init__(self, catalog: list[Content], key_fn_factory):
        self.catalog = catalog
        self.key_fn_factory = key_fn_factory
        self.weekend_days = {2, 3}  # Day 2 = Saturday, Day 3 = Sunday
        self.global_runs_left = {c.id: c.max_runs for c in catalog}

    def run(self):
        weekly_revenue = 0
        weekly_engagement = 0
```

```
        missed = set()

        print("=== SmartStream Scheduling Report ===")

        for day in range(1, 6):
            ctx = RuleContext(day, self.weekend_days,
self.global_runs_left.copy())
            scoring = Scoring(ctx)
            key_fn = self.key_fn_factory(scoring, ctx)

            day_slots = []
            day_revenue = 0
            day_engagement = 0
            notes = []

            # Filter valid content
            candidates = [
                c for c in self.catalog
                if c.in_window(day) and self.global_runs_left[c.id] > 0
            ]

            sorted_candidates = sorted(candidates, key=key_fn, reverse=True)

            for slot in range(10):
                for content in sorted_candidates:
                    if not content.in_window(day):
                        continue
                    if self.global_runs_left[content.id] <= 0:
                        continue

                    hour = 20 if content.is_ppv() else 14
                    revenue, engagement = scoring.adjusted(content, hour)

                    # Log bonuses and penalties
                    if content.is_kids() and 9 <= hour <= 15:
                        notes.append(f"{content.id} (Kids daytime bonus
applied)")
                    if content.is_ppv() and hour == 20 and ctx.is_weekend():
                        notes.append(f"{content.id} (PPV prime-time + weekend
boost)")
                    if ctx.genre_over_cap(content.genre):
                        notes.append(f"{content.id} (Genre cap hit → soft
penalty)")
```

```
                    day_slots.append(content.id)
                    day_revenue += revenue
                    day_engagement += engagement

                    self.global_runs_left[content.id] -= 1
                    ctx.increment_genre(content.genre)
                    break  # Move to next slot

            day_type = "Saturday" if day == 2 else "Sunday" if day == 3 else
"Weekday"
            print(f"\nDay {day} ({day_type}{', Prime-Time rules active' if
ctx.is_weekend() else ''}):")
            print(f"Scheduled: {day_slots}")
            print(f"Total Revenue = ₹{day_revenue}")
            print(f"Total Engagement = {day_engagement}")
            print(f"Notes: {', '.join(notes) if notes else 'All within window'}")

            weekly_revenue += day_revenue
            weekly_engagement += day_engagement

        print("\n-----------------------------------")
        print(f"WEEKLY TOTALS")
        print(f"Total Revenue = ₹{weekly_revenue}")
        print(f"Total Engagement = {weekly_engagement}")
        print("Missed Content:")

        for content in self.catalog:
            if self.global_runs_left[content.id] == content.max_runs:
                print(f"- {content.id} (not scheduled at all)")
            elif self.global_runs_left[content.id] > 0:
                print(f"- {content.id} (MaxRuns not fully used)")
```

Strategy.py

```
from context.rule_context import RuleContext
from scoring.scoring import Scoring
def by_revenue(scoring: Scoring, ctx: RuleContext):
    return lambda c: scoring.adjusted(c, hour=20)[0]

def by_engagement(scoring: Scoring, ctx: RuleContext):
```

```
    return lambda c: scoring.adjusted(c, hour=14)[1]

def balanced(scoring: Scoring, ctx: RuleContext, wR=0.6, wE=0.4):
    return lambda c: (
        wR * scoring.adjusted(c, hour=20)[0] +
        wE * scoring.adjusted(c, hour=20)[1]
    )
```

Content.py

```
from context.rule_context import RuleContext
from scoring.scoring import Scoring
def by_revenue(scoring: Scoring, ctx: RuleContext):
    return lambda c: scoring.adjusted(c, hour=20)[0]

def by_engagement(scoring: Scoring, ctx: RuleContext):
    return lambda c: scoring.adjusted(c, hour=14)[1]

def balanced(scoring: Scoring, ctx: RuleContext, wR=0.6, wE=0.4):
    return lambda c: (
        wR * scoring.adjusted(c, hour=20)[0] +
        wE * scoring.adjusted(c, hour=20)[1]
    )
```

Consent.py

```
from util.clamp import clamp
from context.rule_context import RuleContext
from model.content import Content


class Scoring:
    def __init__(self, ctx: RuleContext):
        self.ctx = ctx

    def adjusted(self, content: Content, hour: int) -> tuple[int, int]:
        revenue, engagement = content.revenue, content.engagement

        # Kids bonus
        if content.is_kids() and 9 <= hour <= 15:
            engagement = clamp(engagement + 10)
```

```
        # PPV prime-time
        if content.is_ppv() and 19 <= hour <= 22:
            revenue *= 1.25

        # Weekend bonus
        if self.ctx.is_weekend():
            engagement = clamp(engagement + 8)
            if content.is_ppv():
                revenue *= 1.15

        # Diversity penalty (soft)
        if self.ctx.genre_over_cap(content.genre):
            engagement = clamp(engagement - 8)

        return int(revenue), int(engagement)
```

Clamp.py

```python
def clamp(value: int, min_val: int = 0, max_val: int = 100) -> int:
    return max(min_val, min(max_val, value))
```

Output:

```
=== SmartStream Scheduling Report ===

Day 1 (Weekday):
Scheduled: ['C1', 'C1', 'C3', 'C3', 'C6', 'C6', 'C6', 'C10', 'C10', 'C10']
Total Revenue = ₹1150
Total Engagement = 801
Notes: All within window

Day 2 (Weekend):
Scheduled: ['C2', 'C1', 'C1', 'C9', 'C9', 'C3', 'C3', 'C7', 'C7', 'C6']
Total Revenue = ₹1915
Total Engagement = 927
Notes: All within window

Day 3 (Weekend):
Scheduled: ['C1', 'C1', 'C9', 'C9', 'C3', 'C3', 'C7', 'C7', 'C8', 'C8']
Total Revenue = ₹1440
Total Engagement = 926
Notes: All within window

Day 4 (Weekday):
Scheduled: ['C5', 'C9', 'C9', 'C3', 'C3', 'C7', 'C7', 'C8', 'C8', 'C6']
Total Revenue = ₹1495
Total Engagement = 825
Notes: All within window

Day 5 (Weekday):
Scheduled: ['C5', 'C9', 'C9', 'C3', 'C3', 'C7', 'C7', 'C8', 'C8', 'C6']
Total Revenue = ₹1495
Total Engagement = 825
Notes: All within window

----------------------------------
WEEKLY TOTALS
Total Revenue = ₹7495
Total Engagement = 4304
```