LittleLives Sr. Fullstack Engineer - Interview Answer Document

Nguyen Nam Long - 1997 - j2teamnnl@gmail.com

Part 1: Technical Problem Solving

1.1 Immediate Diagnosis

Root Cause Hypothesis

Based on the timeline and recent changes, I suspect **the new attendance tracking feature deployed yesterday** is the main cause:

- Monitor only alerted this morning, not over the past week → timing matches yesterday's deployment
- Database connections nearly exhausted (95/100) → possible N+1 query problem
- Response time 15s → inefficient queries or blocking operations

Additionally, real-time notifications (last week) combined with 40% database growth (Indonesia pilot) also contributed to the problem.

First 5 Logs/Metrics to Check

#	Log/Metric	Reason
1	**Slow query logs**	Identify which queries are slow
2	**APM traces** (Laravel Telescope/Clockwork)	See time breakdown and query count per request
3	**Git history** of attendance feature	Identify specific changes and responsible party
4	**Database connection pool metrics**	Confirm connection exhaustion
5	**Error logs** of attendance service	See specific errors occurring

First Team Member to Call

I would call **the developer who deployed the attendance feature yesterday** (based on git history):

- They understand the recent changes best
- Can pair program to fix quickly
- Have context to rollback if needed

1.2 Code Analysis

3 Identified Performance Problems

Problem 1: N+1 Queries

- Code queries sequentially: Child → School → Attendance → Parents → COUNT → Update
- Each request has 6+ queries, increasing exponentially with more students

Problem 2: Synchronous Notifications

- Notifications run synchronously in the request cycle
- User must wait for all notifications to be sent before receiving response
- This is not the core feature (attendance), no need for blocking

Problem 3: Expensive COUNT Query

- Every attendance check runs COUNT for the entire school
- This query does a full table scan, very expensive with large data
- Actually only need to increment a counter

Improved Code

Version 2 - Hotfix (Deploy immediately):

```
public function markAttendance($childId, $status)
{
  $child = $this->Child->findById($childId);
  $school = $this->School->findById($child['Child']['school_id']);
  $parents = $this->Parent->findByChildId($childId);

  $this->Attendance->save([
    'child_id' => $childId,
    'school_id' => $school['School']['id'],
    'status' => $status,
    'timestamp' => date('Y-m-d H:i:s'),
    'marked_by' => $this->Auth->user('id')
  ]);

  // Push notifications to job (async)
  $this->dispatch(new SendAttendanceNotificationsJob([
    'child' => $child,
```

```
    'status' => $status,
    'parents' => $parents
  ]));

  // Use increment instead of SELECT + UPDATE
  $this->School->increment($school['School']['id'], 'today_attendance_count');

  return $this->redirect('/attendance/success');
}
```

Version 3 - Eager Loading:

```
public function markAttendance($childId, $status)
{
  // 1 query with eager loading
  $child = $this->Child->find('first', [
    'conditions' => ['Child.id' => $childId],
    'contain' => ['School', 'Parent']
  ]);

  if (!$child) {
    throw new NotFoundException('Child not found');
  }

  $this->Attendance->save([
    'child_id' => $childId,
    'school_id' => $child['School']['id'],
    'status' => $status,
    'timestamp' => date('Y-m-d H:i:s'),
    'marked_by' => $this->Auth->user('id')
  ]);

  $this->dispatch(new SendAttendanceNotificationsJob([
    'child' => $child,
    'status' => $status,
    'parents' => $child['Parent']
  ]));

  $this->School->increment($child['School']['id'], 'today_attendance_count');

  return $this->redirect('/attendance/success');
}
```

Version 4 - Bulk Attendance (Mechanism change):

```
public function markBulkAttendance($schoolId, array $childIds, $status)
{
  // Query school + all children + parents in 1 query
  $school = $this->School->find('first', [
    'conditions' => ['School.id' => $schoolId],
    'contain' => [
      'Child' => [
        'conditions' => ['Child.id' => $childIds],
        'Parent'
      ]
    ]
  ]);

  if (!$school || empty($school['Child'])) {
    throw new NotFoundException('School or children not found');
  }

  $timestamp = date('Y-m-d H:i:s');
  $markedBy = $this->Auth->user('id');

  // Bulk insert attendance records
  $attendanceRecords = array_map(fn($child) => [
    'child_id' => $child['id'],
    'school_id' => $schoolId,
    'status' => $status,
    'timestamp' => $timestamp,
    'marked_by' => $markedBy
  ], $school['Child']);

  $this->Attendance->saveMany($attendanceRecords);

  // 1 job for all notifications
  $this->dispatch(new SendBulkAttendanceNotificationsJob([
    'school' => $school,
    'children' => $school['Child'],
    'status' => $status
  ]));

  // Increment once by count
  $this->School->incrementBy($schoolId, 'today_attendance_count', count($childIds));

  return $this->redirect('/attendance/success');
}
```

1.3 Quick Fix Strategy

1. **Deploy Version 2 immediately:**
    - Push notifications to job (non-blocking)
    - Replace COUNT + UPDATE with increment
    - No core logic changes, just moving code → safe
2. **If still unstable:**
    - Stats update → move to end-of-day cron job
3. **Traffic Routing Strategy:**
    - Route Indonesian traffic to a separate server (if available)
    - Rate limiting for attendance endpoint to prevent spam
    - Scale up database connections if needed
4. **Rollback Plan:**
    - T+0: Deploy hotfix
    - T+5: Check metrics
    - T+10: If no improvement → git revert to previous version
    - T+15: Verify rollback successful
    - T+20: Stable for demo and testing

Part 2: Architecture & Leadership Decisions

2.1 Technical Risk Assessment

Risk Ranking

Rank	Risk	Reasoning
1	Cross-region latency (Indonesia)	Directly affects customer experience. Product company → customers are priority #1. Losing customers = losing revenue.
2	Database bottleneck	Database down → everything dies. Affects the entire system.
3	Mobile app crash	Affects UX but can be fixed with rate limiting. Customers can wait 1-2 seconds, but crashes are unacceptable.

Rank	Risk	Reasoning
4	PHP memory limits	Can increase memory (costs money but quick). More customers → more revenue → covers the cost.
5	AWS costs	Important for business but not a technical blocker. Having customers = having revenue.

Mitigation Strategy for #1 (Cross-region Latency)

Short-term (immediate):

- CDN for static assets (CloudFront endpoint Indonesia)
- Reduce payload size in API responses
- Cache frequently accessed data (Redis)

Medium-term (1-2 months):

- Deploy read replica in Indonesia region

Long-term:

- Multi-region deployment
- Data locality compliance

2.2 Team Resource Dilemma

Recommendation

Assuming team has 6 people:

Week	Option C (Monitoring)	Option B (Localization)	Option A (Performance)
1	2 people ✓	4 people (in progress)	-
2	Done → move to A	4 people (demo ready)	2 people
3+	-	Continue	2 people ongoing

Reasoning:

1. **Option C first** (2 people, 1 week): Monitoring/alerting is the foundation. Without it, customers face issues → they might leave without us knowing.
2. **Option B in parallel** (4 people): Indonesian features are a commitment to customers. Must have a demo version in 2 weeks to show progress.

3. **Option A after** (2 people from C): Performance issues always exist in product companies. 2 people can fix critical issues first.

Communicating with CEO

> "I understand the urgency of onboarding 500 Indonesian schools.
>
> **Reality:** We cannot ship all 3 with full quality in 2 weeks with the current team.
>
> **My proposal:**
>
> - Week 1: Monitoring (2 people) + Localization starts (4 people)
> - Week 2: Localization has demo version + Performance fix (2 people)
>
> **Results after 2 weeks:**
>
> - ✅ Monitoring/alerting system complete
> - ✅ Demo version of Indonesian features to show customers
> - ✅ Started fixing performance issues

Communicating with Team Members

> "I know option A (performance) is also very important.
>
> **Reason for deferral:**
>
> - Monitoring needs to be done first to detect issues early
> - Indonesian features are a commitment to waiting customers
>
> **My commitment:**
>
> - After week 1, 2 people will switch to performance
> - Performance issues will be tracked and prioritized
> - This is an ongoing effort, not being ignored
>
> **Note:** More people ≠ faster completion. Putting the entire team on 1 task can cause conflicts and communication overhead."

AI Usage

✅ Used AI For:

- Summarizing and translating the challenge to Vietnamese for better understanding
- Creating an answer flow guide
- Syntax checking and code formatting

❌ Self-Analyzed:

- Identifying root cause from timeline and experience
- Designing fix versions (V2 → V3 → V4)
- Risk assessment based on product company experience
- Priority decisions and communication strategy

Validation:

- Reviewed AI-generated code to ensure correct logic
- Cross-checked with actual experience building school management systems
- Ensured solutions are practical, not over-engineered

Conclusion

This is my first time doing this type of challenge interview and I found it very impressive. The test reflects very well the real scenarios that a Sr. Engineer would face.

What I like about LittleLives:

- Real technical problems, not coding puzzles
- Focus on both leadership and communication, not just code
- Approach that allows honest AI usage

I believe that with my experience working at a product company and having built school management systems, I can contribute effectively to the LittleLives team.