TÀI LIỆU PHÂN TÍCH BA

Hệ thống điều phối nhân sự kỹ thuật

Đặc tả kỹ thuật dành cho đội phát triển

Kèm bộ data mẫu: data_mau_dieu_phoi.xlsx
Phiên bản 1.0 · Đối tượng: Dev, Tech Lead, QA

Mục lục

1. Mục đích & phạm vi

Tài liệu dịch bộ rule nghiệp vụ điều phối nhân sự thành đặc tả kỹ thuật để dev hiện thực: data model, thuật toán, API và các edge case. Tài liệu nghiệp vụ gốc (Quy định điều phối) là nguồn chân lý về quy tắc; tài liệu này tập trung cách hiện thực.

Ngoài phạm vi

- UI/UX của hệ thống quản lý công việc (đã có sẵn).
- Module tính lương chi tiết (chỉ định nghĩa điểm tích hợp).
- Hạ tầng triển khai, CI/CD.

2. Mô hình dữ liệu

Các thực thể cốt lõi. Kiểu dữ liệu mang tính gợi ý, đội dev điều chỉnh theo stack thực tế.

Employee (Nhân viên)

Trường	Kiểu	Ghi chú
employee_id	string (PK)	Mã nhân viên
name	string	
areas	array<string>	Danh sách khu vực đăng ký được phép làm
skills	set<skill_id>	Tập kỹ năng (binary — có mặt = có kỹ năng)
performance_score	decimal	Điểm hiệu suất tổng hợp 0–5
status	enum	available / busy / on_leave
monthly_service_pay	decimal	Lương dịch vụ lũy kế tháng (để tính band)

Skill (Kỹ năng)

Trường	Kiểu	Ghi chú
skill_id	string (PK)	
name	string	
is_unique	bool	True nếu ≤1 nhân viên sở hữu → tách flow + cảnh báo HR

Service & Step (Dịch vụ & bước)

Trường	Kiểu	Ghi chú
service_id	string (PK)	
type	enum	regular / branch_opening
urgency	enum	normal / urgent

Trường	Kiểu	Ghi chú
deadline	datetime	Dùng tính tải & cảnh báo trễ
steps	array<Step>	Thứ tự = thứ tự thực hiện (tuần tự)
— step.seq	int	Số thứ tự bước
— step.skill_id	skill_id	Kỹ năng bắt buộc của bước
— step.std_minutes	int	Định biên thời gian
— step.unit_price	decimal	Đơn giá bước
— step.crew_size	int	Số người cần (1 hoặc 2)
— step.split_ratio	array<decimal>	Tỉ lệ chia lương khi crew_size=2, ví dụ [0.6,0.4]

Assignment (Bản gán việc)

Trường	Kiểu	Ghi chú
assignment_id	string (PK)	
service_id / step_seq	ref	Bước được gán
employee_id	ref	Người nhận
assigned_cluster	array<int>	Các step thuộc cùng cụm
flag	enum	auto_in_band / auto_out_of_band / self_picked / urgent
status	enum	assigned / in_progress / done / returned

3. Tham số cấu hình

Tất cả phải nằm trong bảng config, không hardcode. Mỗi tham số có thể override theo nhóm/khu vực/mùa vụ.

Khóa	Mặc định	Ý nghĩa
band_limit	500000	Dải lệch lương ưu tiên (đ)
session_cap_minutes	240	Trần buổi (phút)
urgent_handle_minutes	120	Thời lượng việc khẩn (phút)
w1_performance	0.4	Trọng số điểm hiệu suất
w2_skill_count	0.1	Trọng số số kỹ năng
w3_load	0.3	Trọng số tải (trừ)
w4_pay_fairness	0.5	Trọng số ưu tiên lương thấp

Khóa	Mặc định	Ý nghĩa
spike_threshold	—	Ngưỡng đột biến số lượng để cảnh báo
overdue_threshold_minutes	—	Ngưỡng trễ deadline để điều thêm người

4. Thuật toán điều phối

4.1. Phân loại đầu vào

```
function dispatch(service):
  if service.urgency == 'urgent':
      return dispatchUrgent(service)
  if service.type == 'branch_opening':
      return dispatchBranchProject(service)
  return dispatchRegular(service)   // việc lẻ
```

4.2. Việc lẻ — gom cụm + chấm điểm

Hàm gom cụm cho một ứng viên từ step bắt đầu:

```
function buildCluster(emp, service, startSeq):
  cluster = []; total = 0
  for step in service.steps[startSeq:]:
      if step.skill_id not in emp.skills: break       // cắt do thiếu KN
      if total + step.std_minutes > session_cap: break // cắt do trần buổi
      cluster.add(step); total += step.std_minutes
  return cluster, total
```

Chọn người cho cụm:

```
function dispatchRegular(service):
  startSeq = firstUnassignedStep(service)
  skill = service.steps[startSeq].skill_id
  // B1: lọc cứng
  cands = employees.filter(e =>
      skill in e.skills
      AND service.branch.area in e.areas
      AND e.status == 'available'
      AND remainingSessionTime(e) > 0)

  if cands.isEmpty():
      // B4 van an toàn: nới điều kiện band đã ngầm bỏ ở đây;
      // nếu vẫn rỗng, lấy bất kỳ ai đủ KN+khu vực rảnh
      cands = anyAvailableWithSkill(skill, service.branch.area)
      flag = 'auto_out_of_band'
  else:
      flag = 'auto_in_band'

  // B2-B3: chấm điểm
  minPay = min(e.monthly_service_pay for e in allActive)
  for e in cands:
      cluster, mins = buildCluster(e, service, startSeq)
      payGap = e.monthly_service_pay - minPay
      e.score = w1*e.performance_score
              + w2*len(e.skills)
              - w3*currentLoadHours(e)
              + w4*max(0, (band_limit - payGap))/100000
              + clusterBonus(cluster)   // ưu tiên cụm dài
```

```
  winner = argmax(cands, by=score)
  return createAssignment(winner, cluster, flag)
```

Ghi chú: band là tiêu chí ưu tiên (cộng điểm w4), KHÔNG phải bộ lọc. Người vượt band vẫn nằm trong cands; chỉ bị điểm w4 thấp hơn. Van an toàn ở nhánh cands rỗng đảm bảo không bao giờ pending.

4.3. Chủ động nhận thêm

```
function selfPick(emp, service):
  assert emp finished current cluster early
  assert noOtherCandidateAvailable(service)  // không ai khác làm
  // bỏ qua kiểm tra band hoàn toàn
  return createAssignment(emp, ..., flag='self_picked')
```

4.4. Việc khẩn

```
function dispatchUrgent(service):
  onSite = employeesOnSite(service.branch)
  target = pick(onSite, by=nearest/available)
  // KHÔNG cắt cụm đang làm; giao thêm, cho phép vượt trần buổi
  a = createAssignment(target, service, flag='urgent',
                       allowOverCap=true, maxMinutes=urgent_handle)
  // target có quyền trả phần cụm còn lại:
  // returnRemainingCluster(target) -> status='returned' về hàng đợi
  if spikeDetected() OR overdue(service):
      escalateAddPeople(service)
  return a
```

4.5. Dự án mở chi nhánh

```
function dispatchBranchProject(service):
  uniqueSteps = steps where step.skill.is_unique
  if uniqueSteps not empty:
      for s in uniqueSteps: pinAssign(soleOwner(s.skill), s) // ràng buộc cứng
  // phần còn lại: gom đội phủ kỹ năng theo round-robin
  needed = remainingSkills(service)
  crew = []
  while not covers(crew.skills, needed):
      e = roundRobinNext(eligiblePool(needed, service.area))
      crew.add(e)
  return assignCrew(crew, service)  // làm xuyên suốt; handover khi nghỉ
```

Handover: khi một thành viên on_leave, gán backup từ step đang dở; lương chia theo step mỗi người đã hoàn thành. Kỹ năng độc nhất không có backup → tạo alert HR.

5. Gợi ý API / sự kiện

Endpoint / Event	Mô tả
POST /services	Tiếp nhận việc mới → trigger dispatch()

Endpoint / Event	Mô tả
POST /steps/{id}/complete	NV cập nhật step xong → tính lại tải, mở step kế
GET /queue?area=&skill=	Hàng đợi việc pending cho NV xem & tự nhận
POST /assignments/{id}/self-pick	NV chủ động nhận thêm
POST /assignments/{id}/return	Trả phần cụm còn lại vào hàng đợi
EVENT urgent.created	Kích hoạt dispatchUrgent
EVENT skill.unique.detected	Cảnh báo HR đào tạo dự phòng

6. Edge case & quy tắc xử lý

Tình huống	Xử lý mong đợi
Không ai trong band rảnh	Giao cho bất kỳ ai đủ KN rảnh, gắn cờ out_of_band
Không ai có kỹ năng của step	Đẩy cảnh báo điều phối thủ công; không tự gán sai người
NV làm xong sớm, còn việc tồn	Cho self-pick kể cả ngoài band
Khẩn khi NV đang giữa cụm	Giao thêm, không cắt; NV được quyền trả phần còn lại
Khẩn vượt trần buổi	Cho phép (ngoại lệ), đánh dấu urgent
Step cần 2 người	Gán 2 người đủ KN, chia lương theo split_ratio
Kỹ năng độc nhất, người đó nghỉ	Dự án đứng; alert HR; không có backup hợp lệ
Cụm vượt 4h	Cắt tại step làm tổng ≤ trần; phần dư về hàng đợi
NV quên cập nhật step xong	Không mở được step kế; cần nhắc/agree SLA cập nhật ngay

7. Bộ data mẫu kèm theo

File data_mau_dieu_phoi.xlsx gồm 4 sheet để dev đối chiếu khi viết unit test:

- Sheet 1 — Nhân viên × kỹ năng: 7 nhân viên, 8 kỹ năng (gồm 1 kỹ năng độc nhất: thang máy).
- Sheet 2 — Dịch vụ → step: 4 dịch vụ với định biên, đơn giá, kỹ năng từng step.
- Sheet 3 — 10 kịch bản điều phối: mỗi kịch bản nêu tình huống và kết quả rule mong đợi — dùng làm test case.
- Sheet 4 — Demo chấm điểm: bảng tính sống minh họa band ±500k, chỉnh trọng số để thấy thứ tự ưu tiên đổi.

Khuyến nghị QA: chuyển trực tiếp 10 kịch bản ở Sheet 3 thành bộ acceptance test. Mỗi kịch bản đã có input rõ ràng (nhân viên, kỹ năng, lương) và output mong đợi.

Pseudocode mang tính minh họa logic, không phải code production. Dev điều chỉnh theo stack, thêm transaction/lock cho gán đồng thời (last-write-wins hoặc optimistic locking).