Cahier des charge SUPLLY :

🎯 1. OBJECTIF DU PROJET

Développer une plateforme interne de gestion de commandes SMM (Social Media Marketing) permettant de :

- Passer des commandes vers des fournisseurs tiers (panels SMM)
- Automatiser la livraison progressive (drip feed) sur plusieurs jours
- Gérer les réservations de solde et le tracking des commandes
- Fournir un tableau de bord complet pour les administrateurs et employés

🏗️ 2. ARCHITECTURE TECHNIQUE

2.1 Stack Technologique

Composant	Technologie
Frontend	React 18 + Tailwind CSS
State Management	React Query (@tanstack/react-query)
UI Components	shadcn/ui + Radix UI
Animations	Framer Motion
Backend	Deno Deploy (Serverless Functions)
Base de données	Base44 Entities (BaaS)
Authentification	Base44 Auth
Graphiques	Recharts

2.2 Structure des Fichiers

/

├── pages/

│ ├── Dashboard.js # Page principale avec tous les onglets

│ ├── OrderTracking.js # Suivi des commandes

│ └── Reports.js # Rapports et exports

```
│
├── components/
│   ├── smm/
│   │   ├── OrderForm.jsx # Formulaire de commande
│   │   ├── OrdersTable.jsx # Tableau des commandes
│   │   ├── CategoryExplorer.jsx # Explorateur de services
│   │   └── ServiceCard.jsx
│   │
│   ├── drip/
│   │   ├── ShopifyOrdersTable.jsx # Vue business des commandes
│   │   ├── AccountsTable.jsx # Comptes drip feed
│   │   ├── DripRunsTable.jsx # Historique des runs
│   │   └── AutoDripRunner.jsx # Exécution automatique
│   │
│   ├── balance/
│   │   └── BalanceDisplay.jsx # Affichage solde
│   │
│   └── notifications/
│   └── NotificationCenter.jsx
│
├── functions/
│   ├── smmApi.js # API wrapper fournisseurs
│   ├── runDripEngine.js # Moteur drip feed principal
│   ├── fixBlockedRuns.js # Correction runs bloqués
│   ├── createDripAccount.js # Création compte drip
│   ├── calculateAvailableBalance.js
│   └── cancelDripFeed.js
```

```
│
├── entities/
│   ├── Account.json
│   ├── Order.json
│   ├── ShopifyOrder.json
│   ├── DripRun.json
│   ├── BalanceReservation.json
│   ├── ApiConfig.json
│   └── ... (autres entités)
│
└── Layout.js
```

📊 3. MODÈLE DE DONNÉES

3.1 Entité `Account` (Comptes Drip Feed)

```
{
 "name": "Account",
 "properties": {
 "account_id": { "type": "string", "description": "ID unique du compte" },
 "account_key": { "type": "string", "description": "Clé unique: platform:username" },
 "platform": { "type": "string", "enum": ["TikTok", "Instagram", "Twitter", "YouTube"] },
 "username": { "type": "string", "description": "Username normalisé" },
 "link": { "type": "string", "description": "Lien complet du compte" },
 "status": { "type": "string", "enum": ["ACTIVE", "PAUSED", "COMPLETED", "ERROR"] },
 "daily_base": { "type": "number", "default": 90, "description": "Base journalière" },
 "daily_variation_pct": { "type": "number", "default": 30, "description": "% variation (±)" },
 "panel_min_batch": { "type": "number", "default": 50, "description": "Minimum batch" },
 "window_start": { "type": "string", "default": "08:00" },
 "window_end": { "type": "string", "default": "23:00" },
```

```
"remaining_to_deliver": { "type": "number", "description": "⚠️ VÉRITÉ ABSOLUE" },
"delivered_total": { "type": "number" },
"last_run_date": { "type": "string", "format": "date" },
"next_run_at": { "type": "string", "format": "date-time" },
"service_type": { "type": "string" },
"service_id": { "type": "string" },
"service_name": { "type": "string" },
"provider": { "type": "string" },
"employee_email": { "type": "string" }
}
}
```


3.2 Entité Order (Commandes Panel)


```
{
"name": "Order",
"properties": {
"order_id": { "type": "string", "description": "ID retourné par l'API panel" },
"shopify_order_number": { "type": "string" },
"service_id": { "type": "string" },
"service_name": { "type": "string" },
"provider": { "type": "string", "enum": ["provider1", "provider2"] },
"employee_email": { "type": "string" },
"link": { "type": "string" },
"quantity": { "type": "number" },
"charge": { "type": "number", "description": "Coût de la commande" },
"status": { "type": "string", "enum": ["Pending", "Awaiting Launch", "In progress", "Processing", "Completed", "Partial", "Canceled", "Refunded", "Error"] },
"start_count": { "type": "number" },
"remains": { "type": "number" },
```

```
"runs": { "type": "number" },
"interval": { "type": "number" },
"auto_refill": { "type": "boolean" },
"last_refill": { "type": "string", "format": "date-time" }
}
}
```


3.3 Entité DripRun (Logs d'exécution)


```
{
"name": "DripRun",
"properties": {
"run_id": { "type": "string" },
"account_id": { "type": "string" },
"date": { "type": "string", "format": "date" },
"planned_qty": { "type": "number" },
"sent_qty": { "type": "number" },
"panel_order_id": { "type": "string" },
"status_run": { "type": "string", "enum": ["Planned", "Sent", "Failed", "Skipped"] },
"idempotency_key": { "type": "string", "description": "Clé anti-doublon" },
"error_message": { "type": "string" },
"sent_at": { "type": "string", "format": "date-time" }
}
}
```


3.4 Entité ShopifyOrder (Vue Business)


```
{
"name": "ShopifyOrder",
"properties": {
"shopify_order_number": { "type": "string" },
```

```
 "account_id": { "type": "string" },
 "service_type": { "type": "string" },
 "service_name": { "type": "string" },
 "qty_ordered": { "type": "number" },
 "status_business": { "type": "string", "enum": ["Pending", "InProgress", "Completed", "Error"] },
 "employee_email": { "type": "string" },
 "link": { "type": "string" }
 }
}
```

3.5 Entité **BalanceReservation**

```
{
 "name": "BalanceReservation",
 "properties": {
 "reservation_id": { "type": "string" },
 "provider": { "type": "string" },
 "account_id": { "type": "string" },
 "shopify_order_number": { "type": "string" },
 "reserved_amount": { "type": "number" },
 "remaining_amount": { "type": "number" },
 "consumed_amount": { "type": "number", "default": 0 },
 "type": { "type": "string", "enum": ["drip_feed", "compensation", "standard_order"] },
 "status": { "type": "string", "enum": ["active", "consumed", "cancelled"] },
 "employee_email": { "type": "string" },
 "notes": { "type": "string" }
 }
}
```

3.6 Entité **ApiConfig**

```
{
 "name": "ApiConfig",
 "properties": {
 "provider_name": { "type": "string" },
 "api_url": { "type": "string" },
 "api_key": { "type": "string" },
 "is_active": { "type": "boolean" },
 "last_balance": { "type": "number" },
 "last_check": { "type": "string", "format": "date-time" }
 }
}
```


⚙️ 4. FONCTIONS BACKEND

4.1 `smmApi` — API Wrapper Fournisseurs

Actions supportées:

- balance — Récupérer le solde
- services — Lister les services disponibles
- add — Créer une commande
- status — Vérifier le statut d'une commande
- refill — Relancer une commande
- cancel — Annuler une commande

Comportement:

1. Authentification utilisateur obligatoire
2. Récupération de la config API selon le provider
3. Construction des paramètres URLSearchParams
4. Appel HTTP POST vers l'API du panel
5. Parsing de la réponse JSON
6. Création des logs dans OrderLog
7. Synchronisation automatique du statut après création

4.2 `runDripEngine` — Moteur Principal (V3 Log-Driven)

PRINCIPES FONDAMENTAUX:

1. Source unique de vérité: les logs internes (Account, DripRun, Order)

2. Aucune API externe pour vérification de statut

3. Logique déterministe: LOGS > STATUTS > DATES > ACTIONS

4. Aucune exécution sans runs_log préalable

FLUX D'EXÉCUTION:

ÉTAPE 1 — SÉLECTION

├── Filtrer comptes status = 'ACTIVE'

├── Filtrer next_run_at <= now

└── Retourner liste comptes éligibles

ÉTAPE 2 — VÉRIFICATION COMMANDE (pour chaque compte)

├── SI remaining_to_deliver = 0 → COMPLETED

└── SINON → continuer

ÉTAPE 3 — VÉRIFICATION RUNS (lecture logs internes UNIQUEMENT)

├── Récupérer dernière batch avec panel_order_id

├── Lire statut dans Order (table interne)

├── SI batch EN COURS (Pending, In progress, Processing)

│ └── BLOQUER → décaler à demain 12h

└── SI batch TERMINÉE → OK pour nouveau run

ÉTAPE 4 — CRÉATION DU RUN

├── Calculer quantité: base ± variation%

├── Respecter min(panel_min_batch) et max(remaining)

├── CRÉER runs_log avec status = 'Planned' AVANT exécution

└── Générer idempotency_key unique

ÉTAPE 5 — EXÉCUTION

├── Appeler smmApi action='add'

├── SI erreur → marquer run comme 'Failed'

└── SI succès → récupérer order_id

ÉTAPE 6 — FINALISATION

├── Mettre à jour runs_log → status = 'Sent'

├── Mettre à jour Account → delivered_total, remaining_to_deliver

├── SI remaining = 0 → status = 'COMPLETED'

└── SINON → next_run_at = demain 12h

RÈGLES CRITIQUES:

❌ JAMAIS exécuter un run si une batch est déjà en cours

❌ JAMAIS relancer un run sur une commande terminée

✅ Le statut interne (Order) est prioritaire sur les dates

✅ Toute action doit être précédée d'un runs_log

✅ Un run par jour par compte maximum

4.3 fixBlockedRuns — Correction des Runs Bloqués

Objectif: Corriger les comptes avec dates de run dépassées

Logique:

1. Récupérer tous les comptes ACTIVE
2. Pour chaque compte en retard:
   - Vérifier la dernière batch dans la DB interne
   - Si batch TERMINÉE + remaining > 0 → planifier demain 12h
   - Si batch EN COURS → décaler demain 12h (ne pas exécuter)
   - Si remaining = 0 → marquer COMPLETED
3. Retourner rapport détaillé

4.4 createDripAccount — Création Compte Drip

Validations:

- ✅ UNIQUEMENT pour services "followers" TikTok/Instagram
- ❌ Refusé pour vues, likes, commentaires, etc.

Calcul des paramètres selon quantité:

Quantité	daily_base	variation	min_batch	buffer
< 500	90	±30%	50	15%
500-2000	110	±35%	60	15%
2000-5000	130	±40%	80	15%
> 5000	160	±45%	100	20%

Fusion automatique: Si un compte ACTIVE existe déjà pour le même username, ajouter la quantité au drip existant.

4.5 calculateAvailableBalance — Calcul Solde Disponible

FORMULE:

solde_employé = solde_réel - solde_immobilisé

Calcul solde immobilisé:

- Uniquement comptes ACTIVE
- Uniquement services followers (pas vues/likes)
- coût = (remaining_to_deliver / 1000) × taux_service

Limite négative:

limite_négative = -0.5 × solde_immobilisé

Statuts:

- ok — Solde positif
- warning — Solde négatif mais dans la limite
- blocked — Limite dépassée, commandes bloquées

🖥️ 5. INTERFACE UTILISATEUR

5.1 Dashboard Principal

Onglets:

1. **Dashboard** — Stats, graphiques, commandes récentes
2. **Services** — Explorateur de services avec filtres
3. **Orders** — Historique complet des commandes
4. **Tracking** — Recherche par numéro Shopify
5. **IA** — Assistant IA pour analyse
6. **Drip** (Admin) — Gestion drip feed
7. **Reports** (Admin) — Rapports et exports

8. **Team** (Admin) — Gestion des accès
9. **Refill** (Admin) — Auto-refill
10. **Config** (Admin) — Paramètres API

5.2 Composant OrderForm

Modes de livraison:

- **Standard** — Livraison immédiate en une batch
- **Simple** — Drip feed panel (2-3 runs automatiques)
- **Advanced** — Drip feed internalisé (1 batch/jour)

Validation obligatoire:

- Lien valide
- Quantité > 0 et dans les limites min/max
- Numéro Shopify obligatoire

Vérification avant commande:

1. Vérifier solde disponible
2. Vérifier si limite négative OK
3. Warning si coût > $25

5.3 Composant ShopifyOrdersTable

Affichage:

- Numéro Shopify
- Username extrait du lien
- Quantité commandée vs restante
- Prochain run planifié
- Coût mobilisé (basé sur taux service)
- Service utilisé
- Statut avec badge coloré
- Bouton annulation (hold 6s)

Actions:

- Clic sur ligne → Dialog détails avec historique runs
- Hold 6s sur X → Confirmation annulation

5.4 Composant AccountsTable

Colonnes:

- Account ID
- Username
- Plateforme
- Statut
- Restant / Livré
- Base journalière
- Prochain run
- Progress bar
- Actions

5.5 Composant DripRunsTable

Affichage historique:

- Date du run
- Account ID
- Quantité planifiée vs envoyée
- Panel Order ID
- Statut avec icône
- Message d'erreur si échec

🔐 6. SÉCURITÉ & PERMISSIONS

6.1 Rôles

Rôle	Permissions
admin	Accès total, gestion équipe, config API
user	Commandes, tracking, services

6.2 Protections Backend

- Toutes les fonctions vérifient base44.auth.me()
- Fonctions admin vérifient user.role === 'admin'
- Utilisation base44.asServiceRole pour opérations privilégiées

6.3 Validations

- Drip feed limité aux followers TikTok/Instagram
- Blocage commandes si limite négative dépassée
- Idempotency keys pour éviter doublons

📅 7. AUTOMATISATIONS

7.1 AutoDripRunner

- Exécution automatique quotidienne à 12h (Paris)
- Appelle runDripEngine
- Invalide les queries React Query après exécution

7.2 Auto-refresh Commandes

- Commandes en cours: toutes les 10 minutes
- Toutes les commandes: toutes les heures

7.3 Timers Affichés

- Countdown jusqu'à minuit (auto-refill)
- Countdown prochain refresh actif
- Countdown prochain refresh complet

🎨 8. DESIGN SYSTEM

8.1 Palette de Couleurs

```
:root {
 --background: #0a1628; /* Fond principal */
 --card: #0f1f35; /* Cartes */
 --primary: #7FFF00; /* Vert fluo accent */
 --secondary: #152238; /* Fond secondaire */
 --muted: #1a2d45; /* Éléments désactivés */
 --border: #7FFF00; /* Bordures accent */
}
```

8.2 Badges Statut

Statut	Couleur
Pending	Jaune
InProgress	Bleu
Completed	Vert
Error	Rouge
ACTIVE	Vert
PAUSED	Jaune

8.3 Animations

- Framer Motion pour transitions
- Progress bars animées
- Loaders avec spin
- Fade in/out sur les cartes

📋 9. CHECKLIST DE LIVRAISON

9.1 Entités à créer

- Account
- Order
- ShopifyOrder
- DripRun
- BalanceReservation
- ApiConfig
- OrderLog
- Employee
- BalanceHistory
- RefillHistory
- ServiceCategory
- CustomService
- NotificationHistory
- AccessRequest

9.2 Fonctions Backend

- smmApi (wrapper API)
- runDripEngine (moteur drip)
- fixBlockedRuns (correction)
- createDripAccount
- cancelDripFeed
- calculateAvailableBalance
- getServiceDeliveryStats

9.3 Pages

- Dashboard (avec tous les onglets)
- OrderTracking
- Reports

9.4 Composants Principaux

- OrderForm
- OrdersTable
- CategoryExplorer
- ShopifyOrdersTable
- AccountsTable
- DripRunsTable
- AutoDripRunner
- BalanceDisplay
- NotificationCenter

9.5 Tests à effectuer

- Création commande standard
- Création drip feed (mode advanced)
- Fusion drip feed existant
- Exécution cycle drip
- Correction runs bloqués
- Blocage si limite négative
- Annulation drip feed
- Calcul solde disponible

📞 10. CONTACTS & RÉFÉRENCES

10.1 APIs Fournisseurs

Provider 1 (MoreThanPanel):

- Endpoint: Configuré dans ApiConfig
- Actions: balance, services, add, status, refill, cancel

Provider 2 (BulkMedya):

- Endpoint: Configuré dans ApiConfig
- Mêmes actions

10.2 Fuseau Horaire

- Toutes les heures sont en **Europe/Paris**
- Fenêtre d'exécution: 08h-23h
- Exécution automatique: 12h quotidien