# Plan de Développement : Application de Résolution Mathématique Offline (Flutter)

## 📄 Vue d'ensemble du Projet

Développement d'une application mobile native (iOS/Android) avec **Flutter** capable de résoudre des problèmes mathématiques **hors ligne**. L'application met l'accent sur une UX pédagogique de haute qualité, utilisant la reconnaissance optique (OCR) et un moteur de calcul symbolique embarqué pour fournir des étapes de résolution détaillées (Step-by-step).

### 🎯 Objectifs Clés

1. **100% Offline :** Aucune dépendance API cloud pour la résolution ou l'OCR.
2. **Architecture Clean :** Code modulaire, testable et maintenable (Feature-First Clean Architecture).
3. **Standards IDO :** Respect strict du format *Input-Description-Output* pour chaque étape de résolution.
4. **UI/UX Premium :** Design épuré, animations fluides, rendu LaTeX natif.

## 🏗 1. Architecture Technique & Structure

Nous utiliserons une **Clean Architecture** orientée fonctionnalités (**Feature-First**), couplée au pattern **BLoC** pour la gestion d'état.

### 1.1 Stack Technologique

* **Framework :** Flutter (Dernière version stable).
* **Langage :** Dart.
* **State Management :** flutter\_bloc (Séparation stricte UI / Logique).
* **Dependency Injection :** get\_it + injectable.
* **Navigation :** go\_router.
* **Local Database :** hive (Performance NoSQL pour l'historique).
* **OCR Engine :** google\_mlkit\_text\_recognition (Base) + Modèle TFLite Custom (Complexe via tflite\_flutter).
* **Math Engine (Offline) :** mathsteps (JavaScript) exécuté via flutter\_js (QuickJS engine).
* **Rendering Math :** flutter\_tex (Rendu MathJax local offline).

### 1.2 Structure du Dossier (lib/)

Cette structure doit être strictement respectée pour garantir la maintenabilité.

lib/  
├── app/ # Configuration globale (Thèmes, Routes, Locales)  
├── core/ # Code partagé (Utils, Constants, Errors)  
│ ├── constants/ # Codes couleurs, timeouts  
│ ├── usecases/ # Interface générique UseCase  
│ └── utils/ # LatexParser, ImageUtils  
├── features/ # Modules fonctionnels (Feature-First)  
│ ├── scanner/ # Feature: Caméra & OCR  
│ │ ├── data/ # Sources de données (Camera, TFLite)  
│ │ ├── domain/ # Entités (MathProblem), UseCases (ScanImage)  
│ │ └── presentation/ # BLoC, Pages (CameraScreen), Widgets (Overlay)  
│ ├── solver/ # Feature: Moteur de résolution  
│ │ ├── data/ # MathEngineService (Flutter\_js wrapper)  
│ │ ├── domain/ # Entités (SolutionStep, Explanation)  
│ │ └── presentation/ # SolutionPage, StepsWidget  
│ └── keyboard/ # Feature: Clavier Mathématique Custom  
├── main.dart # Point d'entrée  
└── injector.dart # Injection de dépendances (DI)

## 🎨 2. Design System & Standards Visuels (Conformité IDO)

Basé sur les documents de référence fournis.

### 2.1 Palette de Couleurs (Strict)

* **Primary (Blue) :** #0DA2CC (Variables, éléments actifs)
* **Secondary (Green) :** #6EB819 (Résultats, validations)
* **Tertiary (Orange) :** #FD602E (Opérations, focus, curseurs)
* **Neutral (Gray) :** #7F7F7F (Texte explicatif)
* **Background :** Blanc ou Dark Mode profond (pas de gris sale).

### 2.2 Règles d'Affichage des Étapes (Format IDO)

Chaque étape de résolution ("Solving Step") doit être un Widget composé de 3 blocs verticaux :

1. **Input (Bloc LaTeX) :** L'état de l'équation *avant* la transformation.
   * *Règle :* Doit être identique à l'Output de l'étape précédente.
2. **Description (Texte) :** Une phrase complète expliquant l'action (ex: "Soustraire 4 des deux côtés").
   * *Style :* Police sans-serif, couleur #7F7F7F. Pas de "Je" ou "Nous".
3. **Output (Bloc LaTeX) :** Le résultat *après* transformation.
   * *Coloration :* Les termes qui ont changé doivent être colorés (ex: en Orange #FD602E).

## 🧠 3. Spécifications du Moteur de Résolution (Engine)

C'est le cœur de l'application. Pour obtenir des étapes détaillées hors ligne, nous ne pouvons pas utiliser de simples bibliothèques Dart mathématiques.

### 3.1 Moteur Hybride flutter\_js + mathsteps

Nous allons intégrer la bibliothèque JavaScript mathsteps (utilisée par Socratic/Google) qui est spécialisée dans la décomposition pédagogique.

* **Action :** Créer un bundle JS (math-solver.bundle.js) contenant mathsteps et ses dépendances.
* **Action :** Placer ce fichier dans assets/js/.
* **Implémentation Dart :**
  + Initialiser JavascriptRuntime au démarrage (dans un Isolate pour ne pas bloquer l'UI).
  + Charger le script.
  + Exposer une fonction solve(String latexInput) qui renvoie un JSON structuré.

### 3.2 Structure de Données (Output JSON du Moteur)

Le moteur doit retourner cet objet strict pour alimenter l'UI :

JSON

{  
 "problem\_latex": "2x + 4 = 10",  
 "steps":,  
 "output\_latex": "2x = 6",  
 "changed\_indices": // Pour la coloration syntaxique  
 },  
 {  
 "step\_id": 2,  
 "type": "IDO",  
 "input\_latex": "2x = 6",  
 "description\_key": "divide\_both\_sides",  
 "output\_latex": "x = 3"  
 }  
 ],  
 "final\_answer": "x = 3"  
}

## 📷 4. Module Scanner & OCR (Vision)

### 4.1 Interface Caméra (UX)

* **Widget :** CameraPreview plein écran.
* **Overlay :** Un cadre de redimensionnement ajustable (Crop Box) au centre avec des coins arrondis et une couleur d'accentuation (#0DA2CC).
* **Feedback :** Afficher un indicateur de chargement discret sur le cadre pendant l'analyse.

### 4.2 Pipeline de Reconnaissance

1. **Capture :** L'image est capturée et recadrée selon la Crop Box.
2. **Prétraitement :** Conversion en niveaux de gris + binarisation (noir & blanc) pour nettoyer le bruit (via image\_editor ou opencv).
3. **OCR Mathématique :**
   * *Niveau 1 (Simple) :* Utiliser ML Kit Text Recognition pour les équations linéaires simples.
   * *Niveau 2 (Complexe) :* Si ML Kit échoue ou détecte des symboles complexes (), passer l'image à un modèle **TFLite Custom** (entraîné sur le dataset *IM2LATEX*) via tflite\_flutter.
4. **Conversion :** Le résultat brut est converti en chaîne LaTeX standardisée.

## ⌨️ 5. Clavier Mathématique (Fallback)

Si le scan échoue, l'utilisateur doit pouvoir éditer manuellement.

* **Custom Keyboard :** Ne pas utiliser le clavier système. Créer un Widget MathKeyboard qui s'anime depuis le bas.
* **Layout :** Onglets pour [Nombres], [Fonctions f(x)],, [Lettres].
* **Rendu Temps Réel :** Le champ de saisie (MathField) doit rendre le LaTeX en temps réel (WYSIWYG) en utilisant flutter\_tex.

## 🗓 6. Plan d'Implémentation (Roadmap)

### Phase 1 : Fondations (Semaine 1)

* [ ] Initialiser le projet Flutter flutter create.
* [ ] Configurer l'arborescence Clean Architecture.
* [ ] Configurer le Linter (very\_good\_analysis) pour forcer un code propre.
* [ ] Mettre en place le moteur JS mathsteps et tester la communication Dart <-> JS.

### Phase 2 : Le Cœur Logique (Semaine 2)

* [ ] Créer les UseCases SolveEquation.
* [ ] Implémenter le parsing du JSON de mathsteps vers des entités Dart Step.
* [ ] Créer le système de mapping des descriptions (i18n) pour avoir des phrases en français correct.

### Phase 3 : Interface Utilisateur (Semaine 3)

* [ ] Développer les widgets de rendu LaTeX (flutter\_tex configuré en local).
* [ ] Construire l'écran "Résolution" avec la liste déroulante des étapes (IDO format).
* [ ] Appliquer la charte graphique stricte (Couleurs Photomath).

### Phase 4 : Scanner & OCR (Semaine 4)

* [ ] Intégrer la caméra et le crop widget.
* [ ] Connecter ML Kit pour la reconnaissance texte -> LaTeX.
* [ ] Gérer les erreurs (image floue, pas de maths détectées).

### Phase 5 : Clavier & Finitions (Semaine 5)

* [ ] Développer le clavier mathématique custom.
* [ ] Persistance des données (Historique) avec Hive.
* [ ] Documentation (README) et nettoyage des commentaires.

## 📝 Standards de Code & Maintenance

À inclure dans le README.md du projet :

1. **Commentaires :** Tous les algorithmes complexes (surtout le mapping JS -> Dart) doivent être commentés en anglais technique ou français clair. Utiliser /// pour la documentation des fonctions publiques.
2. **Immutabilité :** Utiliser freezed pour toutes les classes de données (Entities/States) pour garantir l'immutabilité et éviter les bugs d'état.
3. **Tests :**
   * *Unit Tests* obligatoires pour tout le dossier domain/ (Logique mathématique).
   * *Widget Tests* pour vérifier que le rendu LaTeX s'affiche sans erreur.