アプリケーションエンジニア

チームメンバー：渡邉諒（2442102）・田中誠真（2442054）

MVTの構成要素

Model (モデル):
　　データ構造の定義とデータベース操作（ORM）。

View (ビュー):
　　ユーザーからのリクエスト処理、データの取得・加工、SOS判定などのビジネスロジック。

Template (テンプレート):
　　ユーザーへの画面表示（HTML生成）。

メリット:
　　データの独立性が高く、UIの変更がロジックに影響しにくい（保守性）。

データベース定義（models.py）

データベースのテーブル定義をPythonクラスとして記述。

外部キー (ForeignKey) を用いて、ユーザーやカテゴリーとのリレーションを定義。

コードのハイライト（スクリーンショット等を貼る）:
Python class Report(models.Model): user = models.ForeignKey(User, ...) # ユーザー紐付け category = models.ForeignKey(Category, ...) # カテゴリー紐付け condition = models.IntegerField(...) # SOS判定用の数値 (1~5) # ...

ポイント:
condition カラムに 1(SOS) ～ 5(絶好調) の数値を格納し、定量的な分析を可能に設計。

ロジックの実装とクエリ最適化（views.py）

クラスベースビュー (CBV) を採用し、コードの再利用性を向上。

データベースへの負荷を考慮したクエリ構築。

コードのハイライト（スクリーンショット等を貼る）:
N+1問題の対策（結合して取得）
def get_queryset(self): return
Report.objects.select_related('user', 'category')
¥ .prefetch_related('tags')

ロジックの実装とクエリ最適化（views.py）

実装した主なロジック:

検索機能: Qオブジェクト を使用したタイトル・本文の複合検索。
SOS判定: コンディション値に基づくアラートフラグの付与。

プロジェクト構造と環境分離

ディレクトリ構成:

機能ごとにアプリケーション (reports, accounts) を分割し、見通しの良い構成を維持。

```
├── config/ # 全体の設定 (settings.py)
├── reports/ # 日報機能 (models, views)
└── accounts/ #ユーザー認証・管理機能
```

環境設定 (settings.py):

ALLOWED_HOSTS や CSRF_TRUSTED_ORIGINS を設定し、Docker環境および外部公開（ngrok）に対応。

画面構築とレスポンシブ対応

テンプレート継承:

共通部分（ヘッダー、ナビゲーション）を base.html に定義し、全ページで継承。修正箇所を最小限に抑える設計。

スマホ対応 (CSS):

メディアクエリ (@media) を使用し、画面幅に応じたレイアウト変更を実装。一覧画面やフォームの崩れを防ぎ、モバイルからの利用をサポート。