1. 目的

＜ドキュメントの目的を記載＞

2. 前提・移行要件

2.1 前提

＜移行するスコープを定義する＞

2.2 移行期間

＜いつからいつまでに移行を始めて完了するか記載する。＞

＜リミットがある場合、作業中断と切り戻し開始時間も記載する。＞

2.3 移行期間中の業務運用・既存システムへの影響

＜既存システムへの業務継続性が危ぶまれる場合、ユーザーへのアナウンス方法や時期感を明記する。＞

2.4 移行失敗時の影響・切り戻し方法

＜移行に失敗した場合の影響範囲と切り戻しの方法を定義する。例えば、既存ユーザーデータを残しておけば、プログラムさえ戻せば良いかもしれない。＞

2.5 移行時の体制

＜みんなで頑張るぞーって内容を記載＞

3. 移行の流れ

3.1 移行リハーサル方針

移行スケジュールにおいては、なるべく本番同様の方法で検証ができるようにする。

したがって、プログラム/データ共に本番移行の前にリハーサルができていることが望ましい。

＜サンプル＞

リリース1ヶ月前くらい	ユーザーデータの設計完了、移行スクリプトの実装完了 （できれば）移行後のプログラムがテスト環境で実行できる
リリース2週間前～ 1週間前	データ：ステージング用のテナントにて全ユーザデータをリハーサルとしてインポート実施。 プログラム：ステージング環境にて認証～機能利用が可能なことを検証。 また、トークンの有効期限に関する検証も併せて実施する。 ※1 移行手順は本番と同一の物を用意して実施する。 ※2 移行手順が完了した場合、移行テスト（後述）を実施する。
リリース当日 （多分深夜）	なんとかする。

3.2 プログラム更新の方法

＜リリース時に特別な作業がある場合、その旨を記載する＞

3.3 データ移行の方針

＜データ移行の方法について、手順書レベルで記載する＞

4. 移行テスト

4.1 環境と用途

＜サンプル＞

環境	用途	構築すみ
開発環境 (dev)	ローカル 開発	できた
ステージング環境 (stg)	リハーサル ステージング環境	できた
本番環境 (prod)	リハーサル？ 本番	できた

4.2 検証項目

移行実行後、下記の項目を検証することで、問題がないことを確認する。

＜サンプル[Django用]＞

検証タイミング	検証項目	検証結果
バックエンド開発完了	● password, i = 150000 を移行して認証可能 ● password, i = 120000 を移行して認証可能 ● 文字種類（数字、英語[大文字/小文字]、記号）を移行して認証可能 ● 開発者の本番環境用アカウントを移行してみて認証可能	OK/NG

フロントエンド/アプリ開発完了	● フロントエンドから開発環境の認証が可能 (開発環境用アカウントを使用) ● アプリから開発環境の認証が可能 (開発環境用アカウントを使用) ● フロントエンドから開発環境の認証が可能 (開発者の本番環境アカウントを使用) ● アプリから開発環境の認証が可能 (開発者の本番環境アカウントを使用) ● 認証後、各機能が問題なく動作することをリグレッションテストにて検証	OK/NG
リハーサル/本番	● データ移行時、全件が移行できる ● Try Connectionの機能から認証ができる (社内メンバーやデモ用のアカウントが認証できるかチェックする)	OK/NG
リハーサル	● フロントエンド（ステージング）から認証が可能 (社内メンバー全員が認証できるかチェックする) ● アプリ（ステージング）から認証が可能 (社内メンバー全員が認証できるかチェックする)	OK/NG
本番	● フロントエンド（本番）から認証が可能 (社内メンバー全員が認証できるかチェックする) ● アプリ（本番）から認証が可能 (社内メンバー全員が認証できるかチェックする)	OK/NG

4.3 移行直後の業務・システム運用について

下記のパターンが想定されるため、もし運用時に不具合が発生した場合、特別な運用を実施する。

＜サンプル＞

#	事象	対応方法
1	普段使用しているパスワードで弾かれる	＜リセット案内などの対応方法を記載＞
2	パスワードを変更してください、と出る	＜リセット案内などの対応方法を記載＞
3	パスワードがわからない	＜リセット案内などの対応方法を記載＞
4	ログインページが出ない	＜リセット案内などの対応方法を記載＞
5	ログインしたら知らないデータが表示された	＜すぐログアウトさせ、調査完了するまでログインできないようにブロック。など記載＞
6	**[TODO]** その他ブレストする	

5. 移行設計上の課題一覧

＜機能面、障害設計、監査、運用面あたりの軸で課題がないか確認＞