詳細設計

1. システム概要

麻雀の多面待ち（7 枚形）問題を出題し、解答時間を記録する。解答の正誤に応じて結果をデータベースに保存し、最終的に CSV 形式でエクスポートすることができます。

2. 機能一覧

機能名	機能詳細
テーブル作成	プレイヤー情報を保存するための SQLite データベーステーブルを作成します。
プレイヤー情報保存	プレイヤーの名前、解答時間、日時をデータベースに保存します。
CSV 出力	データベース内の全プレイヤー情報を CSV ファイルとしてエクスポートします。
ゲームプレイ	麻雀の多面待ち問題を出題し、プレイヤーの解答時間を計測します。
解答の正誤判定	プレイヤーの解答が正しいかどうかを判定します。
結果保存	正解数が 3 問の場合のみ、解答時間をデータベースに保存します。
メインメニュー	ゲームを開始するか、CSV 出力を選択するインターフェースを提供します。

3. データベース設計

カラム名	型	説明
id	INTEGER	主キー、自動増分
name	TEXT	プレイヤー名
answer_time	REAL	解答時間(秒)
date	TEXT	ゲームをプレイした日付 (YYYY-MM-DD)

4. エラーハンドリング

データベース接続や SQL 操作でエラーが発生した場合は、エラーメッセージを表示し、エラーハンドリングを行います。

ファイル操作（CSV 出力）でもエラーが発生した場合は、エラーメッセージを表示。

5. ユーザーインターフェース設計

コマンドラインでの入力を受け付けます。

・メインメニューで CSV 出力かゲームプレイを選択。

・ゲームをプレイする場合、名前の入力を求め、問題を表示。

・正誤判定の後、結果を表示し、全て正解した場合のみ解答時間を保存。

6. 主要クラスと関数設計

データベース接続

DB_PATH: データベースのパス

conn: SQLite 接続オブジェクト

作成関数

create_table_if_not_exists()

目的: プレイヤー情報用のテーブルが存在しない場合に作成する。

処理内容: SQL でテーブルを作成。エラーハンドリングあり。

結果保存

save_result(name, answer_time)

目的: プレイヤー名、解答時間、日付をデータベースに保存。

入力: プレイヤーの名前、解答時間。

処理内容: データベースに INSERT 文で保存。解答時間は小数点以下 2 桁に丸める。

CSV 出力

export_to_csv()

目的: データベース内の全てのプレイヤー情報を CSV 形式で保存。

処理内容: SELECT 文でデータを取得し、CSV ファイルに書き込む。

ゲーム処理

play_game()

目的: 麻雀の待ち牌問題を出題し、プレイヤーの解答を評価。

処理内容: 3 問をランダムに選択し、プレイヤーに待ち牌を入力させる。正誤判定を行い、タイムを計測。

メインメニュー

main()

目的: ユーザーにゲームプレイか CSV 出力を選ばせるインターフェース。

入力: ユーザーからの選択（1: CSV 出力、2: ゲームプレイ）。

処理内容: 選択に基づいてゲームか CSV 出力を呼び出し。

7. 処理フロー

① **アプリ起動時** create_table_if_not_exists() を呼び出してデータベースのテーブルを確認・作成。

main() でメニュー表示（CSV 出力 or ゲームプレイ選択）。

② **ゲームプレイ選択** play_game() が呼び出され、問題出題開始。

ランダムに 3 問を選び、手牌を表示。→ プレイヤーに解答を促す。

解答が正しいか判定、結果を出力。→ 3 問すべて正解の場合、解答時間を保存。

③ **CSV 出力選択** export_to_csv() が呼び出され、データベースの内容を CSV ファイルとしてエクスポート。

④ **終了** ゲーム終了後、データベース接続をクローズ。

8. テスト計画

単体テスト 各関数（データベースの保存、CSV 出力、解答判定など）を個別にテスト。

結合テスト ゲームプレイ全体の流れをテストし、データベースへの保存、CSV 出力の整合性を確認。

ユーザビリティテスト ユーザーがスムーズに操作できるか、エラーメッセージが分かりやすいかを確認。