# Project Kickoff Document

## 1. Purpose Statement

An application that lets user build and manage a personal Yu-Gi-Oh! Card collection that sorts cards by type and alphabetically and allows them to search for a card.

## 2. MVP Scope

**Add a card** (enter a name → validate against official pool → add to collection)

**View Collection**

**Remove a card**

**Persist Data**

## 3. User Stories

**Add a card**

* As a user, I want to enter the name of a card, so that it is added to my collection as long as it exists in the official card pool

**View Collection**

* I want to be able to see cards grouped by type (Monster/Spell/Trap) and sorted alphabetically to be able to find cards easily

**Remove a card**

* I want to be able to delete any card from my collection, so that way I can keep it up to date

**Persist Data**

* I want my collection to be saved locally, so that way I don’t lose my cards when I refresh or close the app.

## 4. Task Breakdown

**Add a Card**

* Create the “Add Card” form (input field + submit button)
* Implement client-side logic to capture user input
* Load or fetch the official card pool for validation
* Validate entered name against the card pool
* Add valid cards to the in-memory collection state
* Display an error message for invalid names
* Commit changes with meaningful messages

**View Collection**

* Build the UI component to display the card list
* Group cards by type (Monster/Spell/Trap) in code
* Sort each group alphabetically by card name
* Render grouped and sorted cards in the UI
* Write unit tests for grouping and sorting logic

**Remove a Card**

* Add a “Remove” button for each card in the UI
* Implement the delete function to update state
* Optionally add a confirmation prompt
* Commit changes after removal functionality

**Persist Data**

* Choose storage mechanism (e.g., localStorage)
* Implement “save” on collection updates
* Implement “load” on application startup
* Test persistence across page reloads

## 5. Architecture & Tech Stack

* **Application Type**: Standalone desktop application (not browser-based).
* **Language & Framework**: C# with .NET (e.g., .NET 6 or later).
* **UI Framework**: WinForms (or optionally WPF) for the desktop GUI.
* **Design Pattern**: MVP (for WinForms) or MVVM (if using WPF) to separate UI from business logic.
* **State Management**: In-memory C# collections representing the user’s card collection.
* **Data Storage**: SQLite database accessed via Entity Framework Core for local persistence.
* **Version Control**: Git with a GitHub repository for code and issue tracking.
* **Testing**: NUnit (or xUnit) for unit tests covering grouping, sorting, validation, and data access logic.
* **Build & Distribution**: .NET CLI (dotnet build, dotnet publish) with self-contained deployment or installer packaging.

## 6. Milestones

* **Release 1 (MVP)**: Add, view, remove cards with local persistence and basic validation.
* **Release 2**: Enhance UX with live search-as-you-type and improved error handling.
* **Release 3**: Integrate card images into the collection view.
* **Release 4 (Future)**: Connect to an external API for live card data, add user accounts and cloud sync.