# QR Storage App — Design Document

## Overview

Box Storage ("qrstorageapp") is a cross-platform Flutter application to catalogue and manage physical storage boxes. Each box can contain metadata (title, description, location), pictures, and an assigned box number. The app persists data locally using Hive and provides a simple, mobile-friendly UI to add, view, edit, favorite, and delete boxes.

This document gives a brief overview of the project's goals, architecture, data model, main screens, and notes for development and testing.

## Goals

- Provide an easy way to track physical boxes and their contents.
- Support photos for each box and quick search/filtering.
- Persist data locally (no sign-in) using a lightweight storage solution.
- Cross-platform (mobile, desktop, web) using Flutter.

## High-level Architecture

- UI: Flutter + Provider for lightweight state management (ThemeManager used for theming).
- Local storage: Hive (fast, key-value local DB) for storing BoxItem objects.
- File storage: Images associated with boxes are stored as file paths and loaded with Image.file.
- Entrypoint: `lib/main.dart` sets up Hive, routes, and app theme.

## Main Components / Files

- `lib/main.dart` — App entrypoint, theme/provider initialization, Hive setup.
- `lib/home_screen.dart` — Primary UI showing a grid of boxes, search, sort, and navigation to add or detail screens.
- `lib/add_box_screen.dart` — Form to create or edit a box, add photos, description, location, and box number.
- `lib/box_detail.dart` — Detailed view of a box with edit/delete actions and image viewer.
- `lib/full_image.dart` — Full-screen image viewer for box photos.
- `lib/models/box_item.dart` (+ generated

## Data Model (BoxItem)

A single box item typically stores:

- Unique ID / index (managed by Hive)
- boxNumber (human-friendly number like "Box #3")
- displayTitle / title
- description
- location (optional text)
- isFavorite (bool)
- imagePaths (list of strings referencing file paths)
- created/updated timestamps (optional)

This model is persisted in a Hive Box keyed by name `boxes`.

## UI & UX Highlights

- Home / Grid view: Displays box cards with image and title; favorites are visually marked and sorted first.
- Search: Expandable AppBar search to filter boxes by title, description, or location.
- Add / Edit flow: Add new boxes with photos, title, description and optional location.
- Deletion: Confirms deletion with an undo SnackBar.
- Theme: Light/Dark toggle via a ThemeManager, stored in-app.

## Platform & Assets

- Targets: Android, iOS, Web, Linux, macOS, Windows (platform-specific folders present).
- Assets: `assets/images/boxpicture.png` used as a default placeholder image.

## Development & Running

Quick start (from project root):

- Ensure Flutter and required SDKs are installed.
- Run the app during development:
  - Mobile: `flutter run -d <device>`
  - Web: `flutter run -d chrome`
  - Windows/Linux/macOS: `flutter run -d windows` (or platform-specific device)

Notes:

- Hive adapters must be registered before usage (see `main.dart`).
- Generated files (e.g., `box_item.g.dart`) are managed by

## Testing

- Unit/widget tests found under `test/` should be run via:

  flutter test

- For integration tests, use `flutter drive` or `flutter test integration_test` depending on setup.

## Ideas / Future Enhancements

- Add QR code generation & scanning per box to physically label boxes for quick lookup.
- Sync / backup to cloud (e.g., Firebase, Azure) for multi-device access.
- Allow tagging and nested folder/grouping of boxes.
- Add batch import/export (CSV or JSON) for migration.
- Add QOL features