Helwan university
Faculty of Engineering
Computer and systems Engineering Department

DO IT.

Goal Tracker and to do Application

Design & Technical Report

February 2026

Submitted by
Hagar Khaled Helmy

Code | video

Contents

1. Project Overview 2
1.1 Core Components 2
2. Color Palette 2
3. Typography 3
4. UI Component Breakdown 3
4.1 State Variables 3
4.2 Handler Function 3
4.3 FlatList (Scrollable Goal List) 3
5. Screenshots & UI States 4
5.1 Default / Initial State 4
5.2 Typing in the Goal Input 5
5.3 After Clicking Add Button 6
5.4 Marking a Goal as Complete 7
5.5 Empty State 8
5.6 List with Custom Title 9
6. Design Decisions 10
7. Technical Summary 10

1. Project Overview

DO IT is a React-based goal tracker application designed with a dark, brutalist-minimal aesthetic. The app allows users to create named goal lists, add individual goals, mark them as complete, and track overall progress. The interface draws inspiration from editorial design and raw industrial typography.

1.1 Core Components

- Text input for the list title
- TextInput (textarea) for entering individual goals
- Button to trigger goal addition
- FlatList (scrollable div) rendering all goals
- Two state variables: inputText and goals
- One handler function: handleAddGoal

2. Color Palette

The app uses a dark monochrome base with high-contrast lime green (#C8F135) as the primary accent, creating a bold, modern feel against near-black surfaces. An orange accent (#FF6B35) is reserved for decorative typographic punctuation.

Swatch	Name	Hex Code	Usage
	Lime Accent	`#C8F135`	Primary CTA, highlights, progress bar
	Blue Accent	`#2274A5`	Decorative punctuation, list title
	Background	`#0D0D0D`	App background
	Surface	`#161616`	Cards, input fields
	Surface 2	`#1F1F1F`	Secondary surfaces, badge bg
	Border	`#2A2A2A`	Dividers, outlines
	Text	`#F0EDE6`	Primary readable text
	Muted	`#6B6B6B`	Placeholder, labels, secondary text
	Done BG	`#1A1F0D`	Background for completed goal items

3. Typography

Two Google Fonts are used to create a clear typographic hierarchy between display/structural elements and body content.

Font Family	Weights Used	Application
Syne	400, 700, 800	App title, section labels, goal numbers, list names, buttons — all structural display text
DM Sans	300, 400, 500 (+ italic)	Goal item text, input fields, subtitle, body copy, labels — readable conversational text

4. UI Component Breakdown

4.1 State Variables

The app uses two React state variables as required:

- inputText (string) — tracks the current value of the goal textarea, updated via onChange handler
- goals (array) — stores the list of goal objects, each with id, text, and done fields

4.2 Handler Function

The single handleAddGoal function fires on button click. It trims the input, creates a new goal object with a timestamp ID, prepends it to the goals array, and resets the input field. It also accepts Enter key presses in the textarea as a shortcut.

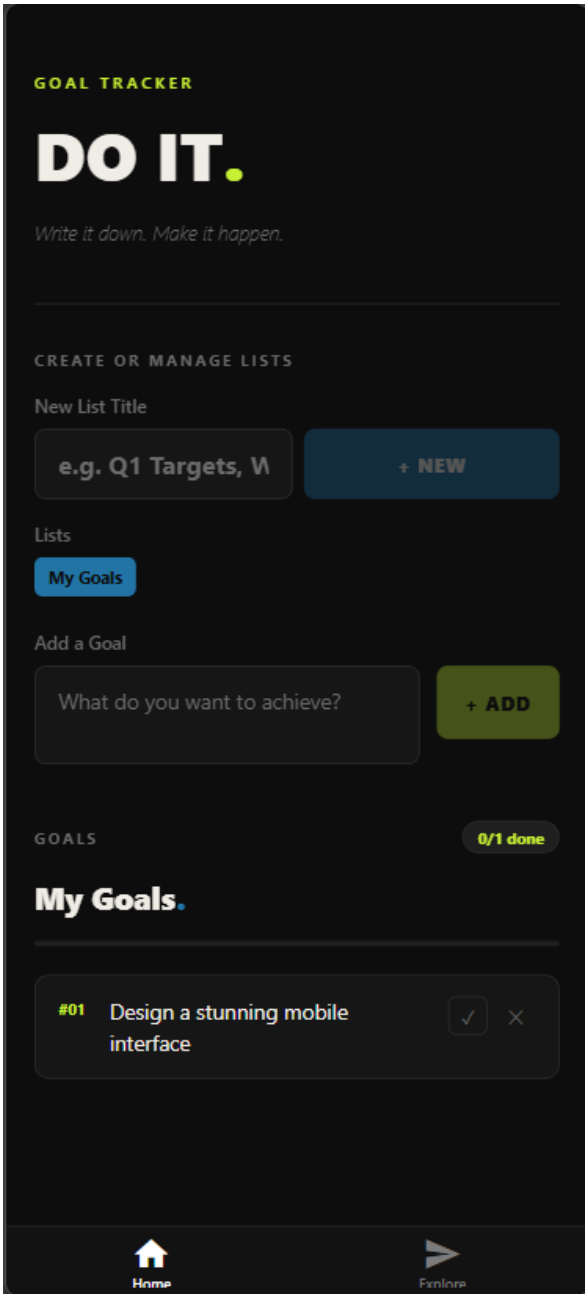
4.3 FlatList (Scrollable Goal List)

The goals list is rendered as a scrollable container using a CSS-capped max-height div with overflow-y: auto. Each goal item is mapped from the goals state array, displaying a number, goal text, a checkmark toggle, and a delete button. New items animate in from the top using a CSS keyframe.

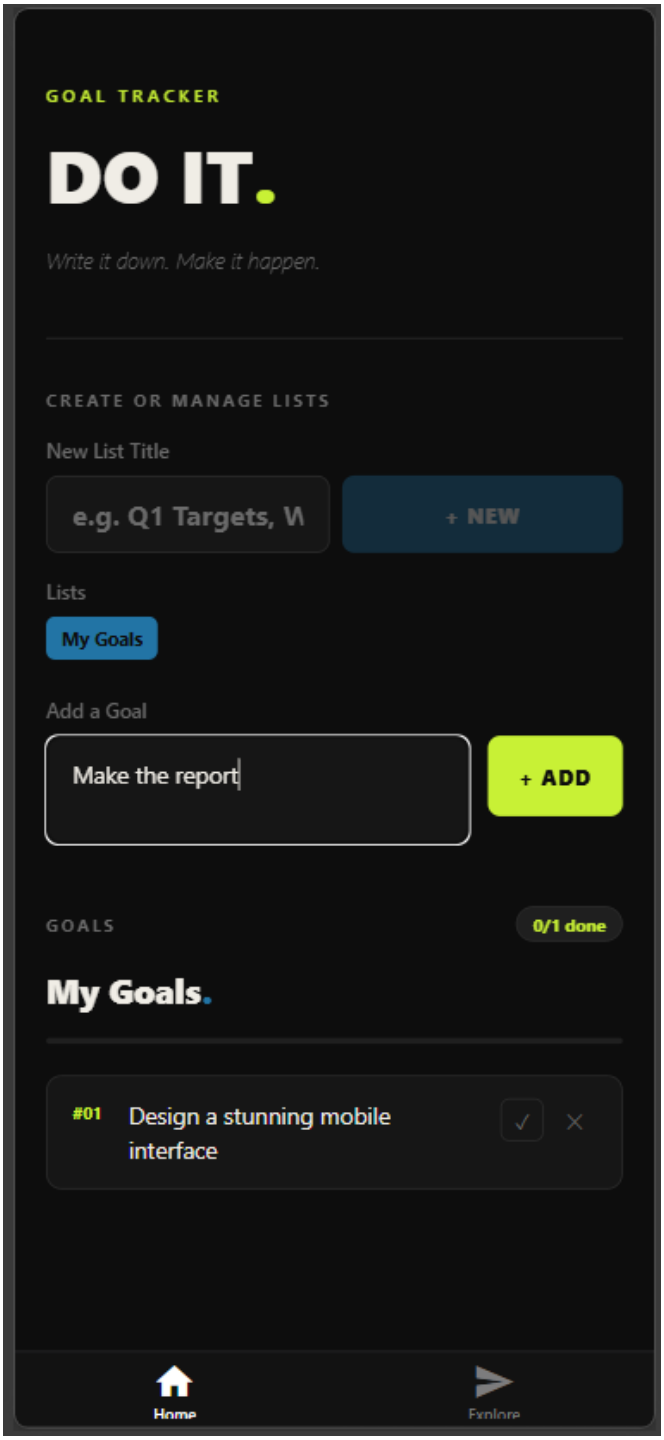
5. Screenshots & UI States

The following panels illustrate the main interface states of the application.

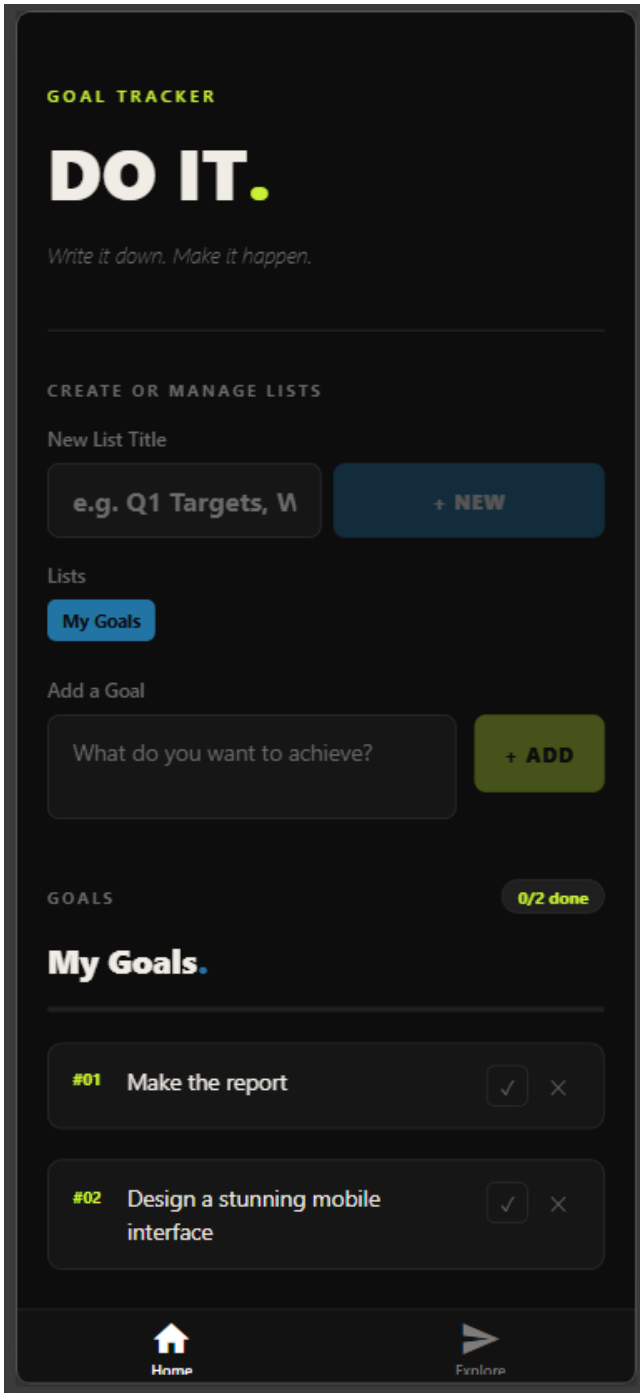
5.1 Default / Initial State



5.2 Typing in the Goal Input

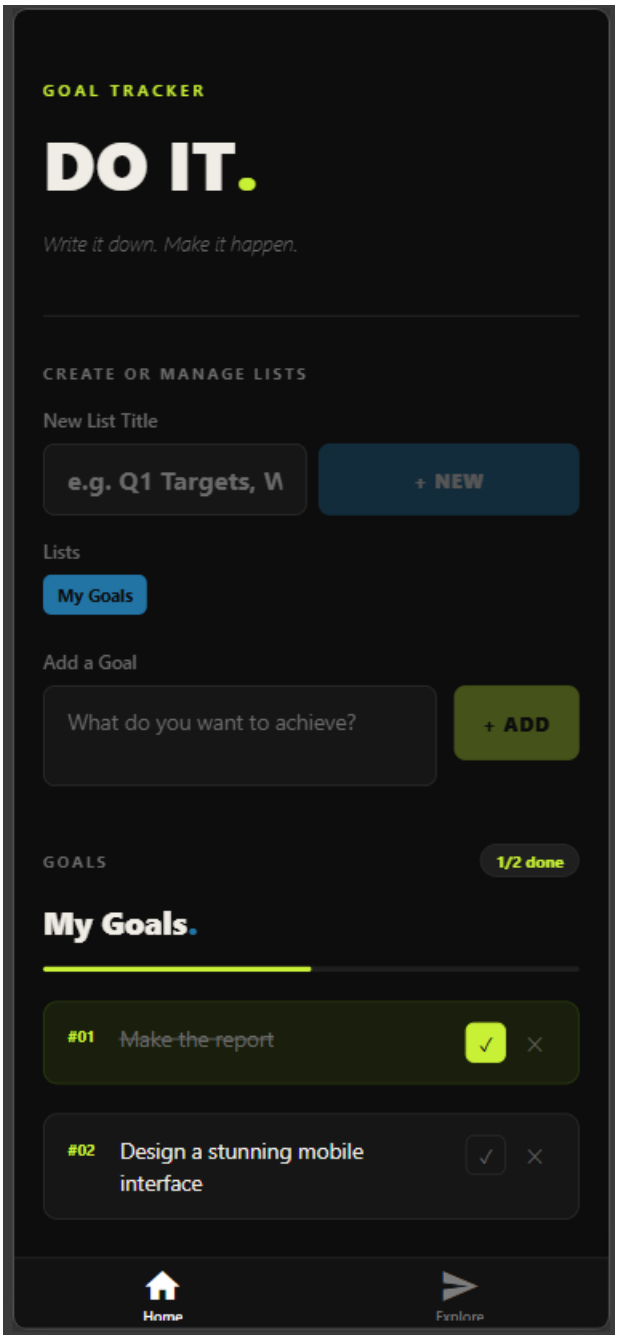


5.3 After Clicking Add Button



5.4 Marking a Goal as Complete

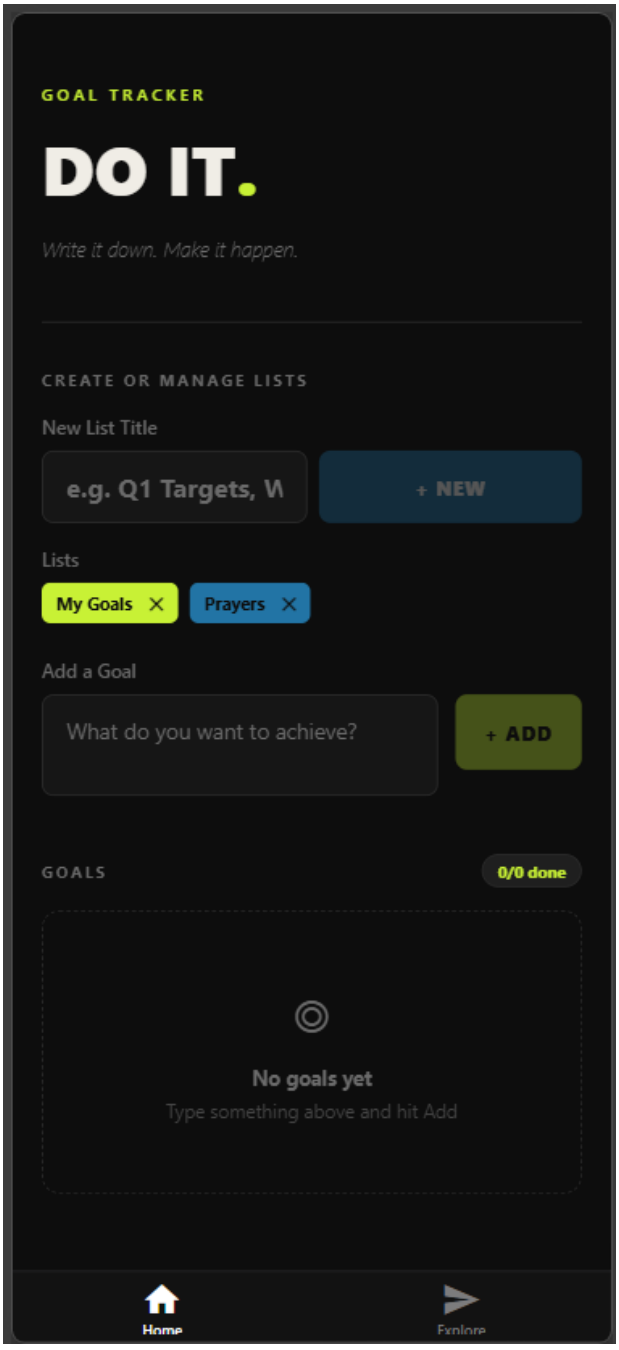
The progress bar percentage and X/Y badge count update immediately to reflect completion.



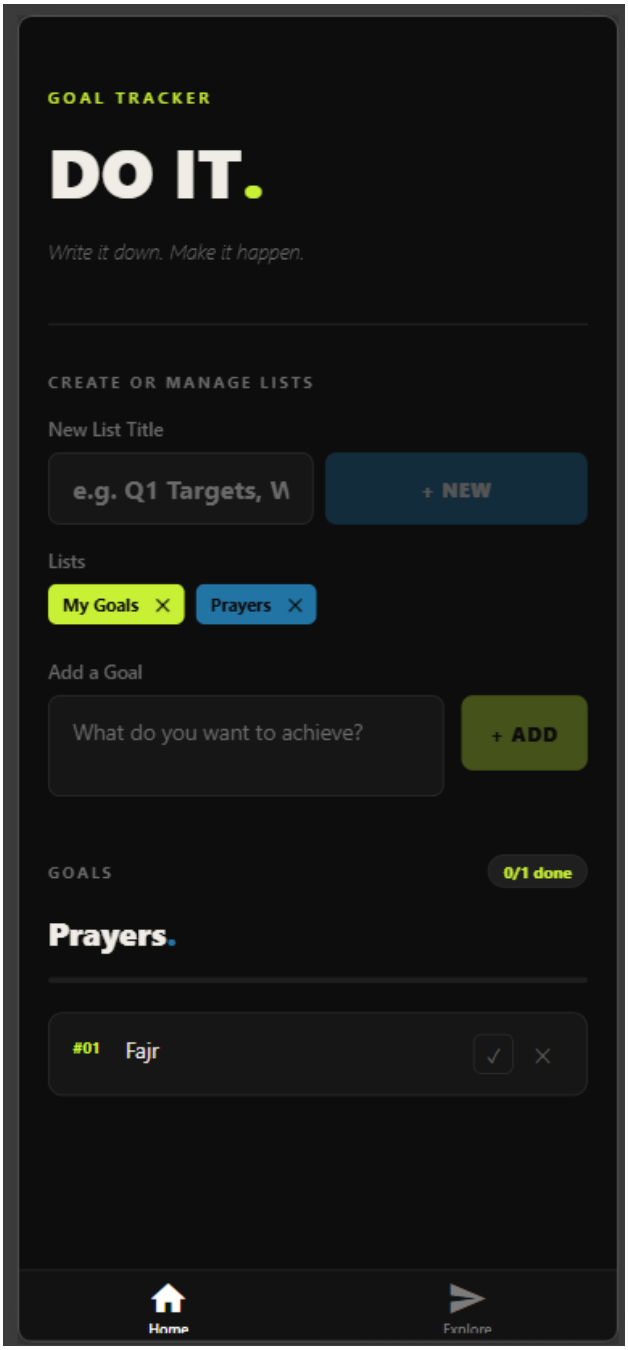
5.5 Empty State

Empty List State

When all goals are deleted, the FlatList renders an empty-state panel with a circular icon, 'No goals yet' heading (Syne font), and instructional subtext. No progress bar is shown.



5.6 List with Custom Title



6. Design Decisions

Dark theme with lime accent: The near-black background (#0D0D0D) with lime green (#C8F135) creates maximum contrast and a distinctive, editorial feel that stands apart from typical blue/purple app palettes.

Syne font for structure: Syne is a geometric, angular typeface that gives UI labels and titles a strong, deliberate presence without being decorative. Its 800 weight creates visual hierarchy instantly.

DM Sans for content: DM Sans is optimized for screen legibility at small sizes. Its light italic variant is used for the app subtitle, creating a calm contrast against the bold Syne headings.

Animated list entries: New goals slide in from above using a CSS keyframe animation, providing clear visual confirmation that the add action succeeded.

Progress bar: A simple 4px bar beneath the list title gives an at-a-glance completion ratio without cluttering the interface with numbers or percentages.

7. Technical Summary

Property	Value
Framework	React (JSX, functional components with hooks)
State management	useState — inputText and goals
Handler function	handleAddGoal — creates and prepends new goal
List component	Scrollable div (FlatList equivalent), mapped from goals state
Fonts	Syne (Google Fonts) + DM Sans (Google Fonts)
Styling	Injected CSS with CSS custom properties (variables)
Animations	CSS @keyframes slideIn on new list items
Extra interactions	Keyboard shortcut (Enter), mark-done, delete goal