

## 1. Введение

К нашей компании обратился основатель Microsoft Билл Гейтс с проблемой калькулятора в OS Windows 11: “Our calculator is extremely inconvenient to use, no one understands how to multiply or divide on it; users are moving to Linux or Mac OS. Help us urgently! The budget is unlimited!”.

Проведя совещание, длительностью шесть секунд, мы согласились ему помочь, а учитывая все обстоятельства – убедили из всех арифметических операций оставить только сложение. Было составлено и утверждено следующее техническое задание.

## 2. Техническое задание «Калькулятор»

Нужно сделать калькулятор, в котором поддерживается только одна математическая операция – сложение. Результаты вычислений и состояние приложения сохраняется между сессиями (сеансами) приложения – нужно хранить и отображать историю вычислений, состояние ввода. Требования к внешнему виду можно найти в приложении А.

## 3. Основные требования

Основные требования, определяемые общепринятыми в компании технологиями и инструментами, применяемыми в создании приложений, согласно которым требуется разработать приложение «Калькулятор»:

1. Использовать Clean Architecture + MVP (Model-View-Presenter);
2. Поддерживаемые арифметические операции - сложение;
3. Состояние приложения сохраняется между сеансами.

### 3.1. Использование Clean Architecture + MVP

Реализовать слои Presentation/Data/Domain, каждый из которых находится в своей assembly. Presentation слой реализовать с использованием шаблона Model-View-Presenter.

### 3.2. Поддерживаемые арифметические операции - сложение

В случае, если пользователь вводит что-то кроме чисел и знака «+», в результат выводить сообщение «Error». Примеры **правильных** выражений:  $54+21$ ,  $45+00$ . Примеры **неправильных** выражений:  $45+-88$ ,  $98.12+48.1$ . Сделать нужно так, как получается проще всего. Главное получить результат

сложения введенных пользователем данных или вывести сообщение «Error», если результат получить невозможно.

### **3.3. Состояние приложения сохраняется между сеансами**

Когда пользователь закрывает приложение, нужно сохранять его состояние, в данном случае - введенное пользователем выражение. После открытия приложения, сохраненное состояние нужно восстановить. Пример: пользователь ввел  $34+47$ , закрыл приложение, открывает его, на экране введено  $34+47$ .

## **4. Сценарий использования**

Пользователю доступно поле ввода и кнопка «Result», по нажатию на которую в поле ввода выводится результат выражения. Если пользователь вводит что-то кроме чисел и знака «+», то в результат выводится сообщение «Error» (Основные требования п.2). При открытии приложения, пользователь продолжает работу с того же места, где он и остановился (Основные требования п.3). Примеры использования смотреть в приложении А (рисунки 1-5).

## **5. Результат**

Результат выполнения технического задания предоставить в виде исходного кода (zip или unitypackage архив, ссылка на проект github или bitbucket, в зависимости от предпочтений исполнителя). К результату приложить пояснительную записку, в которой будет описано решение данного тестового задания.

**Перед отправкой результата выполнения тестового задания убедиться в том, что оно выполнено верно, соблюдены все требования, поведение приложения соответствует представленному в приложении А сценарию.**

Данное техническое задание защищено авторским правом и честным словом. Всем приступающим к его выполнению желаем успехов и приятного времяпровождения.

## Приложение А

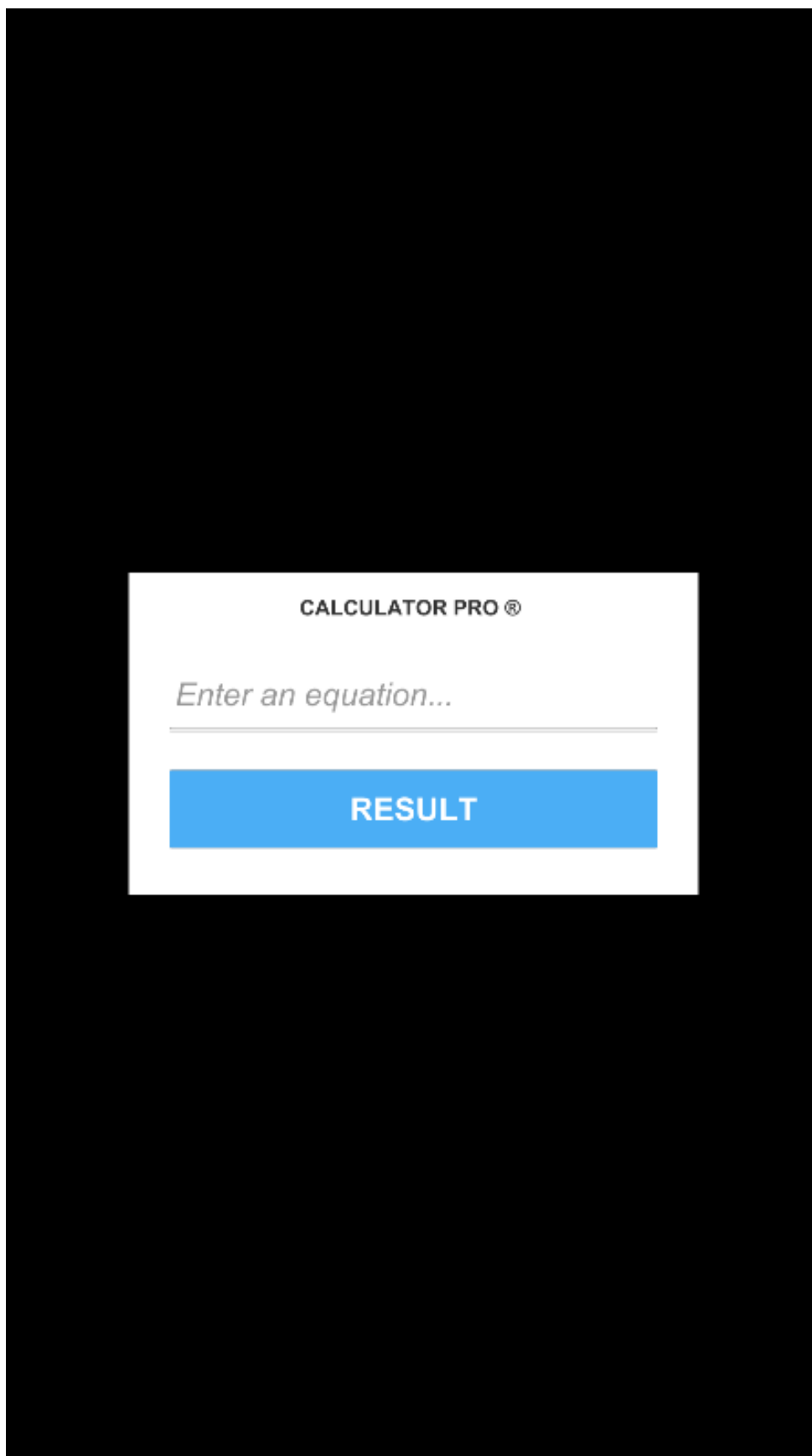


Рисунок 1 – Приложение в начальном состоянии

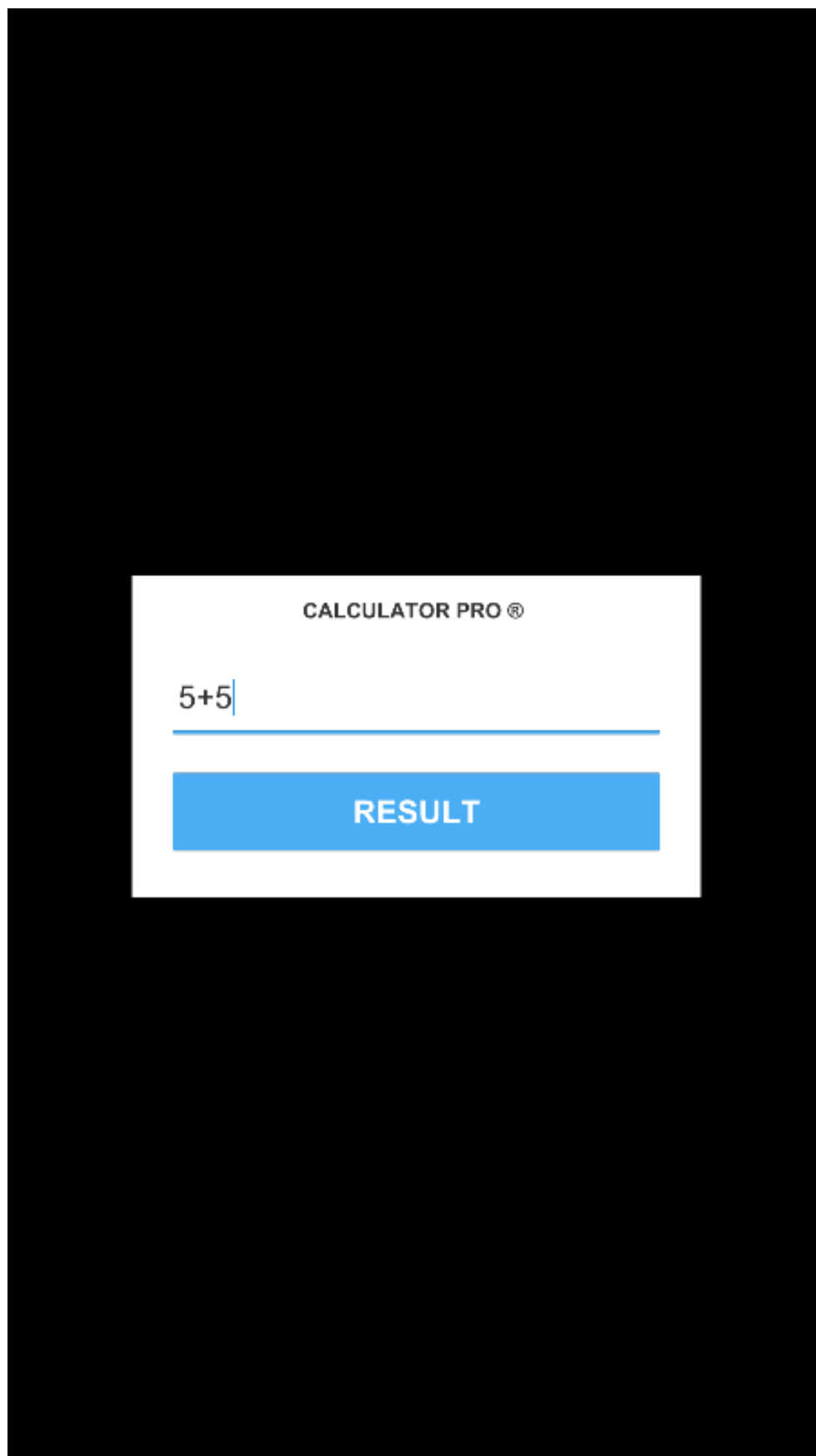


Рисунок 2 – Приложение в состоянии ввода

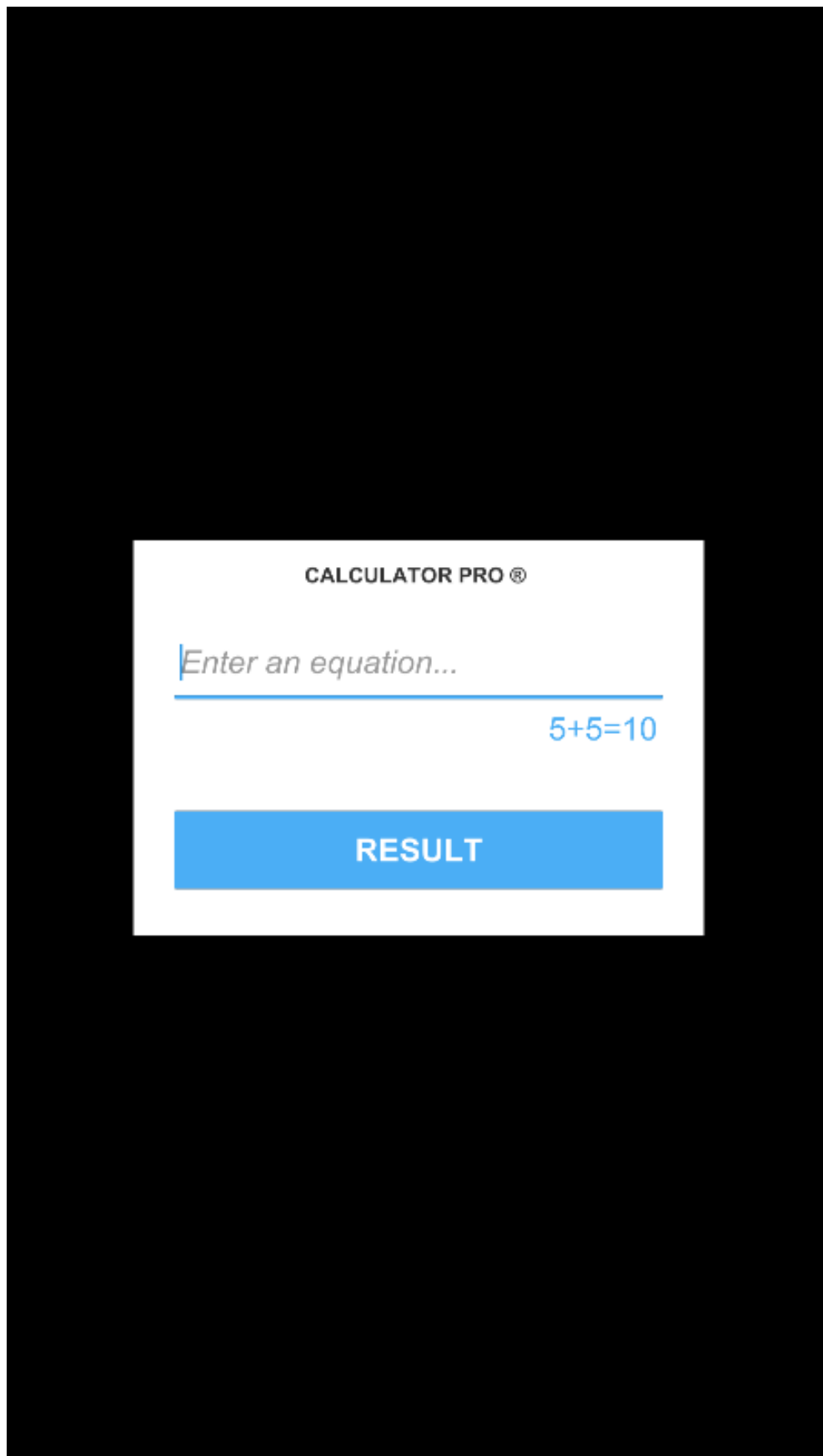


Рисунок 3 – Результат ввода корректной операции



Рисунок 4 – Результат ввода операции с ошибкой



Рисунок 5 – Отображение истории при большом количестве выполненных операций