Build this as my initial prototype

Copy-paste this component to /components/ui folder:
```tsx
expandable-chat.tsx
"use client";

import React, { useRef, useState } from "react";
import { X, MessageCircle } from "lucide-react";
import { cn } from "@/lib/utils";
import { Button } from "@/components/ui/button";

export type ChatPosition = "bottom-right" | "bottom-left";
export type ChatSize = "sm" | "md" | "lg" | "xl" | "full";

const chatConfig = {
  dimensions: {
    sm: "sm:max-w-sm sm:max-h-[500px]",
    md: "sm:max-w-md sm:max-h-[600px]",
    lg: "sm:max-w-lg sm:max-h-[700px]",
    xl: "sm:max-w-xl sm:max-h-[800px]",
    full: "sm:w-full sm:h-full",
  },
  positions: {
    "bottom-right": "bottom-5 right-5",
    "bottom-left": "bottom-5 left-5",
  },
  chatPositions: {
    "bottom-right": "sm:bottom-[calc(100%+10px)] sm:right-0",
    "bottom-left": "sm:bottom-[calc(100%+10px)] sm:left-0",
  },
  states: {
    open: "pointer-events-auto opacity-100 visible scale-100 translate-y-0",
    closed:
      "pointer-events-none opacity-0 invisible scale-100 sm:translate-y-5",
  },
};

interface ExpandableChatProps extends React.HTMLAttributes<HTMLDivElement> {
  position?: ChatPosition;
  size?: ChatSize;
  icon?: React.ReactNode;
}
```

```
const ExpandableChat: React.FC<ExpandableChatProps> = ({
  className,
  position = "bottom-right",
  size = "md",
  icon,
  children,
  ...props
}) => {
  const [isOpen, setIsOpen] = useState(false);
  const chatRef = useRef<HTMLDivElement>(null);

  const toggleChat = () => setIsOpen(!isOpen);

  return (
    <div
      className={cn(`fixed ${chatConfig.positions[position]} z-50`, className)}
      {...props}
    >
      <div
        ref={chatRef}
        className={cn(
          "flex flex-col bg-background border sm:rounded-lg shadow-md overflow-hidden
transition-all duration-250 ease-out sm:absolute sm:w-[90vw] sm:h-[80vh] fixed inset-0 w-full
h-full sm:inset-auto",
          chatConfig.chatPositions[position],
          chatConfig.dimensions[size],
          isOpen ? chatConfig.states.open : chatConfig.states.closed,
          className,
        )}
      >
        {children}
        <Button
          variant="ghost"
          size="icon"
          className="absolute top-2 right-2 sm:hidden"
          onClick={toggleChat}
        >
          <X className="h-4 w-4" />
        </Button>
      </div>
      <ExpandableChatToggle
        icon={icon}
        isOpen={isOpen}
        toggleChat={toggleChat}
```

```
      />
    </div>
  );
};

ExpandableChat.displayName = "ExpandableChat";

const ExpandableChatHeader: React.FC<React.HTMLAttributes<HTMLDivElement>> = ({
  className,
  ...props
}) => (
  <div
    className={cn("flex items-center justify-between p-4 border-b", className)}
    {...props}
  />
);

ExpandableChatHeader.displayName = "ExpandableChatHeader";

const ExpandableChatBody: React.FC<React.HTMLAttributes<HTMLDivElement>> = ({
  className,
  ...props
}) => <div className={cn("flex-grow overflow-y-auto", className)} {...props} />;

ExpandableChatBody.displayName = "ExpandableChatBody";

const ExpandableChatFooter: React.FC<React.HTMLAttributes<HTMLDivElement>> = ({
  className,
  ...props
}) => <div className={cn("border-t p-4", className)} {...props} />;

ExpandableChatFooter.displayName = "ExpandableChatFooter";

interface ExpandableChatToggleProps
  extends React.ButtonHTMLAttributes<HTMLButtonElement> {
  icon?: React.ReactNode;
  isOpen: boolean;
  toggleChat: () => void;
}

const ExpandableChatToggle: React.FC<ExpandableChatToggleProps> = ({
  className,
  icon,
  isOpen,
```

```
  toggleChat,
  ...props
}) => (
  <Button
    variant="default"
    onClick={toggleChat}
    className={cn(
      "w-14 h-14 rounded-full shadow-md flex items-center justify-center hover:shadow-lg
hover:shadow-black/30 transition-all duration-300",
      className,
    )}
    {...props}
  >
    {isOpen ? (
      <X className="h-6 w-6" />
    ) : (
      icon || <MessageCircle className="h-6 w-6" />
    )}
  </Button>
);

ExpandableChatToggle.displayName = "ExpandableChatToggle";

export {
  ExpandableChat,
  ExpandableChatHeader,
  ExpandableChatBody,
  ExpandableChatFooter,
};
```

demo.tsx

```
"use client"

import { useState, FormEvent } from "react"
import { Send, Bot, Paperclip, Mic, CornerDownLeft } from "lucide-react"
import { Button } from "@/components/ui/button"
import {
  ChatBubble,
  ChatBubbleAvatar,
  ChatBubbleMessage,
} from "@/components/ui/chat-bubble"
import { ChatInput } from "@/components/ui/chat-input"
import {
```

```
  ExpandableChat,
  ExpandableChatHeader,
  ExpandableChatBody,
  ExpandableChatFooter,
} from "@/components/ui/expandable-chat"
import { ChatMessageList } from "@/components/ui/chat-message-list"

export function ExpandableChatDemo() {
  const [messages, setMessages] = useState([
    {
      id: 1,
      content: "Hello! How can I help you today?",
      sender: "ai",
    },
    {
      id: 2,
      content: "I have a question about the component library.",
      sender: "user",
    },
    {
      id: 3,
      content: "Sure! I'd be happy to help. What would you like to know?",
      sender: "ai",
    },
  ])

  const [input, setInput] = useState("")
  const [isLoading, setIsLoading] = useState(false)

  const handleSubmit = (e: FormEvent) => {
    e.preventDefault()
    if (!input.trim()) return

    setMessages((prev) => [
      ...prev,
      {
        id: prev.length + 1,
        content: input,
        sender: "user",
      },
    ])
    setInput("")
    setIsLoading(true)
```

```
  setTimeout(() => {
    setMessages((prev) => [
      ...prev,
      {
        id: prev.length + 1,
        content: "This is an AI response to your message.",
        sender: "ai",
      },
    ])
    setIsLoading(false)
  }, 1000)
}

const handleAttachFile = () => {
  //
}

const handleMicrophoneClick = () => {
  //
}

return (
  <div className="h-[600px] relative">
    <ExpandableChat
      size="lg"
      position="bottom-right"
      icon={<Bot className="h-6 w-6" />}
    >
      <ExpandableChatHeader className="flex-col text-center justify-center">
        <h1 className="text-xl font-semibold">Chat with AI ✨</h1>
        <p className="text-sm text-muted-foreground">
          Ask me anything about the components
        </p>
      </ExpandableChatHeader>

      <ExpandableChatBody>
        <ChatMessageList>
          {messages.map((message) => (
            <ChatBubble
              key={message.id}
              variant={message.sender === "user" ? "sent" : "received"}
            >
              <ChatBubbleAvatar
                className="h-8 w-8 shrink-0"
```

```
              src={
                message.sender === "user"
                  ?
"https://images.unsplash.com/photo-1534528741775-53994a69daeb?w=64&h=64&q=80&crop=
faces&fit=crop"
                  :
"https://images.unsplash.com/photo-1494790108377-be9c29b29330?w=64&h=64&q=80&crop=
faces&fit=crop"
              }
              fallback={message.sender === "user" ? "US" : "AI"}
            />
            <ChatBubbleMessage
              variant={message.sender === "user" ? "sent" : "received"}
            >
              {message.content}
            </ChatBubbleMessage>
          </ChatBubble>
        ))}

        {isLoading && (
          <ChatBubble variant="received">
            <ChatBubbleAvatar
              className="h-8 w-8 shrink-0"

src="https://images.unsplash.com/photo-1494790108377-be9c29b29330?w=64&h=64&q=80&cr
op=faces&fit=crop"
              fallback="AI"
            />
            <ChatBubbleMessage isLoading />
          </ChatBubble>
        )}
      </ChatMessageList>
    </ExpandableChatBody>

    <ExpandableChatFooter>
      <form
        onSubmit={handleSubmit}
        className="relative rounded-lg border bg-background focus-within:ring-1
focus-within:ring-ring p-1"
      >
        <ChatInput
          value={input}
          onChange={(e) => setInput(e.target.value)}
          placeholder="Type your message..."
```

```
            className="min-h-12 resize-none rounded-lg bg-background border-0 p-3
shadow-none focus-visible:ring-0"
          />
          <div className="flex items-center p-3 pt-0 justify-between">
            <div className="flex">
              <Button
                variant="ghost"
                size="icon"
                type="button"
                onClick={handleAttachFile}
              >
                <Paperclip className="size-4" />
              </Button>

              <Button
                variant="ghost"
                size="icon"
                type="button"
                onClick={handleMicrophoneClick}
              >
                <Mic className="size-4" />
              </Button>
            </div>
            <Button type="submit" size="sm" className="ml-auto gap-1.5">
              Send Message
              <CornerDownLeft className="size-3.5" />
            </Button>
          </div>
        </form>
      </ExpandableChatFooter>
    </ExpandableChat>
  </div>
 )
}

```

Copy-paste these files for dependencies:
```tsx
shadcn/button
import * as React from "react"
import { Slot } from "@radix-ui/react-slot"
import { cva, type VariantProps } from "class-variance-authority"
```

```
import { cn } from "@/lib/utils"

const buttonVariants = cva(
  "inline-flex items-center justify-center whitespace-nowrap rounded-md text-sm font-medium
ring-offset-background transition-colors focus-visible:outline-none focus-visible:ring-2
focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none
disabled:opacity-50",
  {
    variants: {
      variant: {
        default: "bg-primary text-primary-foreground hover:bg-primary/90",
        destructive:
          "bg-destructive text-destructive-foreground hover:bg-destructive/90",
        outline:
          "border border-input bg-background hover:bg-accent hover:text-accent-foreground",
        secondary:
          "bg-secondary text-secondary-foreground hover:bg-secondary/80",
        ghost: "hover:bg-accent hover:text-accent-foreground",
        link: "text-primary underline-offset-4 hover:underline",
      },
      size: {
        default: "h-10 px-4 py-2",
        sm: "h-9 rounded-md px-3",
        lg: "h-11 rounded-md px-8",
        icon: "h-10 w-10",
      },
    },
    defaultVariants: {
      variant: "default",
      size: "default",
    },
  },
)

export interface ButtonProps
  extends React.ButtonHTMLAttributes<HTMLButtonElement>,
    VariantProps<typeof buttonVariants> {
  asChild?: boolean
}

const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(
  ({ className, variant, size, asChild = false, ...props }, ref) => {
    const Comp = asChild ? Slot : "button"
    return (
```

```
      <Comp
        className={cn(buttonVariants({ variant, size, className }))}
        ref={ref}
        {...props}
      />
    )
  },
)
Button.displayName = "Button"

export { Button, buttonVariants }

```
```tsx
jakobhoeg/chat-bubble
"use client"

import * as React from "react"
import { cn } from "@/lib/utils"
import { Avatar, AvatarFallback, AvatarImage } from "@/components/ui/avatar"
import { Button } from "@/components/ui/button"
import { MessageLoading } from "@/components/ui/message-loading";

interface ChatBubbleProps {
  variant?: "sent" | "received"
  layout?: "default" | "ai"
  className?: string
  children: React.ReactNode
}

export function ChatBubble({
  variant = "received",
  layout = "default",
  className,
  children,
}: ChatBubbleProps) {
  return (
    <div
      className={cn(
        "flex items-start gap-2 mb-4",
        variant === "sent" && "flex-row-reverse",
        className,
      )}
    >
```

```
      {children}
    </div>
  )
}

interface ChatBubbleMessageProps {
  variant?: "sent" | "received"
  isLoading?: boolean
  className?: string
  children?: React.ReactNode
}

export function ChatBubbleMessage({
  variant = "received",
  isLoading,
  className,
  children,
}: ChatBubbleMessageProps) {
  return (
    <div
      className={cn(
        "rounded-lg p-3",
        variant === "sent" ? "bg-primary text-primary-foreground" : "bg-muted",
        className
      )}
    >
      {isLoading ? (
        <div className="flex items-center space-x-2">
          <MessageLoading />
        </div>
      ) : (
        children
      )}
    </div>
  )
}

interface ChatBubbleAvatarProps {
  src?: string
  fallback?: string
  className?: string
}

export function ChatBubbleAvatar({
```

```
  src,
  fallback = "AI",
  className,
}: ChatBubbleAvatarProps) {
  return (
    <Avatar className={cn("h-8 w-8", className)}>
      {src && <AvatarImage src={src} />}
      <AvatarFallback>{fallback}</AvatarFallback>
    </Avatar>
  )
}

interface ChatBubbleActionProps {
  icon?: React.ReactNode
  onClick?: () => void
  className?: string
}

export function ChatBubbleAction({
  icon,
  onClick,
  className,
}: ChatBubbleActionProps) {
  return (
    <Button
      variant="ghost"
      size="icon"
      className={cn("h-6 w-6", className)}
      onClick={onClick}
    >
      {icon}
    </Button>
  )
}

export function ChatBubbleActionWrapper({
  className,
  children,
}: {
  className?: string
  children: React.ReactNode
}) {
  return (
    <div className={cn("flex items-center gap-1 mt-2", className)}>
```

```
    {children}
   </div>
  )
}

```
```tsx
jakobhoeg/message-loading
// @hidden
function MessageLoading() {
  return (
    <svg
      width="24"
      height="24"
      viewBox="0 0 24 24"
      xmlns="http://www.w3.org/2000/svg"
      className="text-foreground"
    >
      <circle cx="4" cy="12" r="2" fill="currentColor">
        <animate
          id="spinner_qFRN"
          begin="0;spinner_OcgL.end+0.25s"
          attributeName="cy"
          calcMode="spline"
          dur="0.6s"
          values="12;6;12"
          keySplines=".33,.66,.66,1;.33,0,.66,.33"
        />
      </circle>
      <circle cx="12" cy="12" r="2" fill="currentColor">
        <animate
          begin="spinner_qFRN.begin+0.1s"
          attributeName="cy"
          calcMode="spline"
          dur="0.6s"
          values="12;6;12"
          keySplines=".33,.66,.66,1;.33,0,.66,.33"
        />
      </circle>
      <circle cx="20" cy="12" r="2" fill="currentColor">
        <animate
          id="spinner_OcgL"
          begin="spinner_qFRN.begin+0.2s"
          attributeName="cy"
```

```
          calcMode="spline"
          dur="0.6s"
          values="12;6;12"
          keySplines=".33,.66,.66,1;.33,0,.66,.33"
        />
      </circle>
    </svg>
  );
}

export { MessageLoading };

```
```tsx
shadcn/avatar
"use client"

import * as React from "react"
import * as AvatarPrimitive from "@radix-ui/react-avatar"

import { cn } from "@/lib/utils"

const Avatar = React.forwardRef<
  React.ElementRef<typeof AvatarPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Root>
>(({ className, ...props }, ref) => (
  <AvatarPrimitive.Root
    ref={ref}
    className={cn(
      "relative flex h-10 w-10 shrink-0 overflow-hidden rounded-full",
      className,
    )}
    {...props}
  />
))
Avatar.displayName = AvatarPrimitive.Root.displayName

const AvatarImage = React.forwardRef<
  React.ElementRef<typeof AvatarPrimitive.Image>,
  React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Image>
>(({ className, ...props }, ref) => (
  <AvatarPrimitive.Image
    ref={ref}
    className={cn("aspect-square h-full w-full", className)}
```

```
    {...props}
  />
))
AvatarImage.displayName = AvatarPrimitive.Image.displayName

const AvatarFallback = React.forwardRef<
  React.ElementRef<typeof AvatarPrimitive.Fallback>,
  React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Fallback>
>(({ className, ...props }, ref) => (
  <AvatarPrimitive.Fallback
    ref={ref}
    className={cn(
      "flex h-full w-full items-center justify-center rounded-full bg-muted",
      className,
    )}
    {...props}
  />
))
AvatarFallback.displayName = AvatarPrimitive.Fallback.displayName

export { Avatar, AvatarImage, AvatarFallback }

```
```tsx
jakobhoeg/chat-input
import * as React from "react";
import { Textarea } from "@/components/ui/textarea";
import { cn } from "@/lib/utils";

interface ChatInputProps extends React.TextareaHTMLAttributes<HTMLTextAreaElement>{}

const ChatInput = React.forwardRef<HTMLTextAreaElement, ChatInputProps>(
  ({ className, ...props }, ref) => (
    <Textarea
      autoComplete="off"
      ref={ref}
      name="message"
      className={cn(
        "max-h-12 px-4 py-3 bg-background text-sm placeholder:text-muted-foreground
focus-visible:outline-none focus-visible:ring-ring disabled:cursor-not-allowed disabled:opacity-50
w-full rounded-md flex items-center h-16 resize-none",
        className,
      )}
      {...props}
```

```
    />
  ),
);
ChatInput.displayName = "ChatInput";

export { ChatInput };
```
```tsx
shadcn/textarea
import * as React from "react"

import { cn } from "@/lib/utils"

export interface TextareaProps
  extends React.TextareaHTMLAttributes<HTMLTextAreaElement> {}

const Textarea = React.forwardRef<HTMLTextAreaElement, TextareaProps>(
  ({ className, ...props }, ref) => {
    return (
      <textarea
        className={cn(
          "flex min-h-[80px] w-full rounded-md border border-input bg-background px-3 py-2
text-sm ring-offset-background placeholder:text-muted-foreground focus-visible:outline-none
focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed
disabled:opacity-50",
          className
        )}
        ref={ref}
        {...props}
      />
    )
  }
)
Textarea.displayName = "Textarea"

export { Textarea }

```
```tsx
jakobhoeg/chat-message-list
import * as React from "react";
import { ArrowDown } from "lucide-react";
import { Button } from "@/components/ui/button";
import { useAutoScroll } from "@/components/hooks/use-auto-scroll";
```

```
interface ChatMessageListProps extends React.HTMLAttributes<HTMLDivElement> {
  smooth?: boolean;
}

const ChatMessageList = React.forwardRef<HTMLDivElement, ChatMessageListProps>(
  ({ className, children, smooth = false, ...props }, _ref) => {
    const {
      scrollRef,
      isAtBottom,
      autoScrollEnabled,
      scrollToBottom,
      disableAutoScroll,
    } = useAutoScroll({
      smooth,
      content: children,
    });

    return (
      <div className="relative w-full h-full">
        <div
          className={`flex flex-col w-full h-full p-4 overflow-y-auto ${className}`}
          ref={scrollRef}
          onWheel={disableAutoScroll}
          onTouchMove={disableAutoScroll}
          {...props}
        >
          <div className="flex flex-col gap-6">{children}</div>
        </div>

        {!isAtBottom && (
          <Button
            onClick={() => {
              scrollToBottom();
            }}
            size="icon"
            variant="outline"
            className="absolute bottom-2 left-1/2 transform -translate-x-1/2 inline-flex rounded-full
shadow-md"
            aria-label="Scroll to bottom"
          >
            <ArrowDown className="h-4 w-4" />
          </Button>
        )}
```

```
      </div>
    );
  }
);

ChatMessageList.displayName = "ChatMessageList";

export { ChatMessageList };

```
```tsx
jakobhoeg/use-auto-scroll
// @hidden
import { useCallback, useEffect, useRef, useState } from "react";

interface ScrollState {
  isAtBottom: boolean;
  autoScrollEnabled: boolean;
}

interface UseAutoScrollOptions {
  offset?: number;
  smooth?: boolean;
  content?: React.ReactNode;
}

export function useAutoScroll(options: UseAutoScrollOptions = {}) {
  const { offset = 20, smooth = false, content } = options;
  const scrollRef = useRef<HTMLDivElement>(null);
  const lastContentHeight = useRef(0);
  const userHasScrolled = useRef(false);

  const [scrollState, setScrollState] = useState<Scr
```