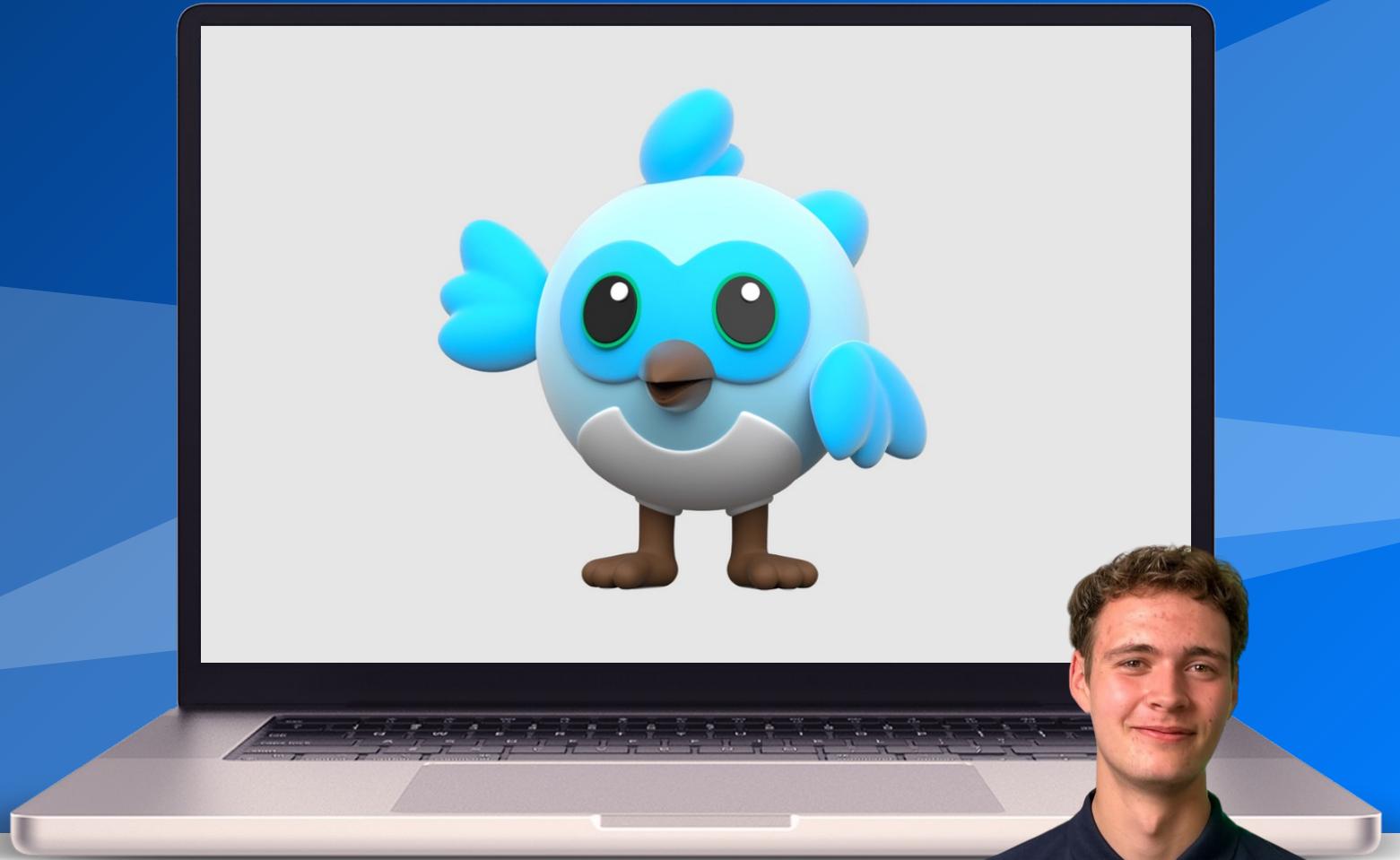




Flutter

12.01.2024



Julian Hartl



That's me



Julian Hartl

- Computer Science student at TUM
- Software Engineer at SelectCode for ~ 2 years
- Currently: Full stack development
- Before: Mainly Flutter

We develop the future

We are an IT agency for startups and medium-sized businesses



2017

Foundation

18

Team members

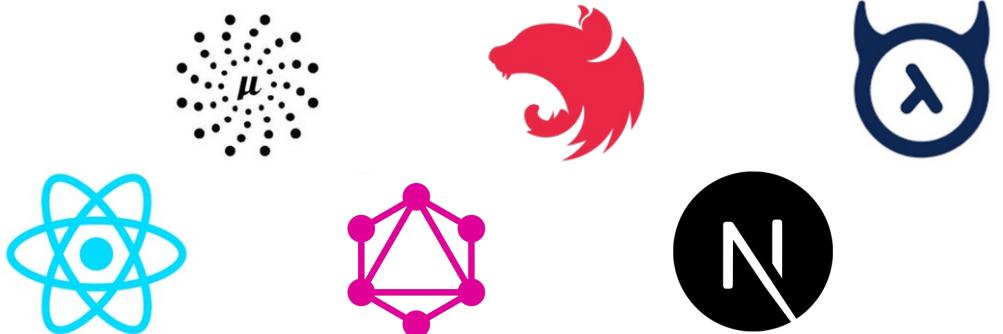
10+

Projects



We are an IT service provider to develop innovative and sustainable SaaS platforms, apps and AI-based tools to advance the technology of the future. With our development team in Munich and our strong ties to TUM, we are at the cutting edge of technology.

what we use besides Flutter:



Our Top 3 why...

...you should join the SelectCode



Working student team

currently 16 working
students



Events & Workshops

eg. SelectCode Weekend



Good salary

18 - 27 €

Still free on 19.01.?
Then take part in our Coding Escape Room competition!

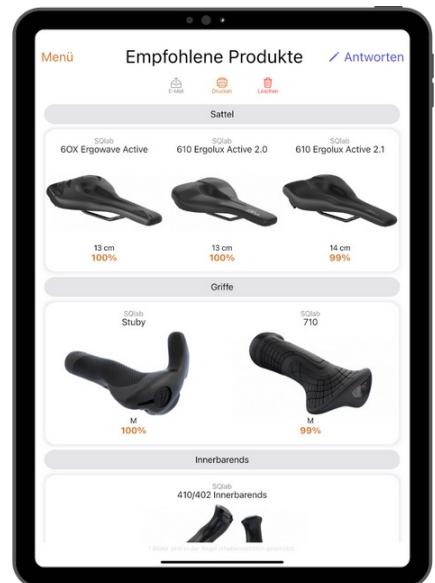


SQlab Profiler

Case Study



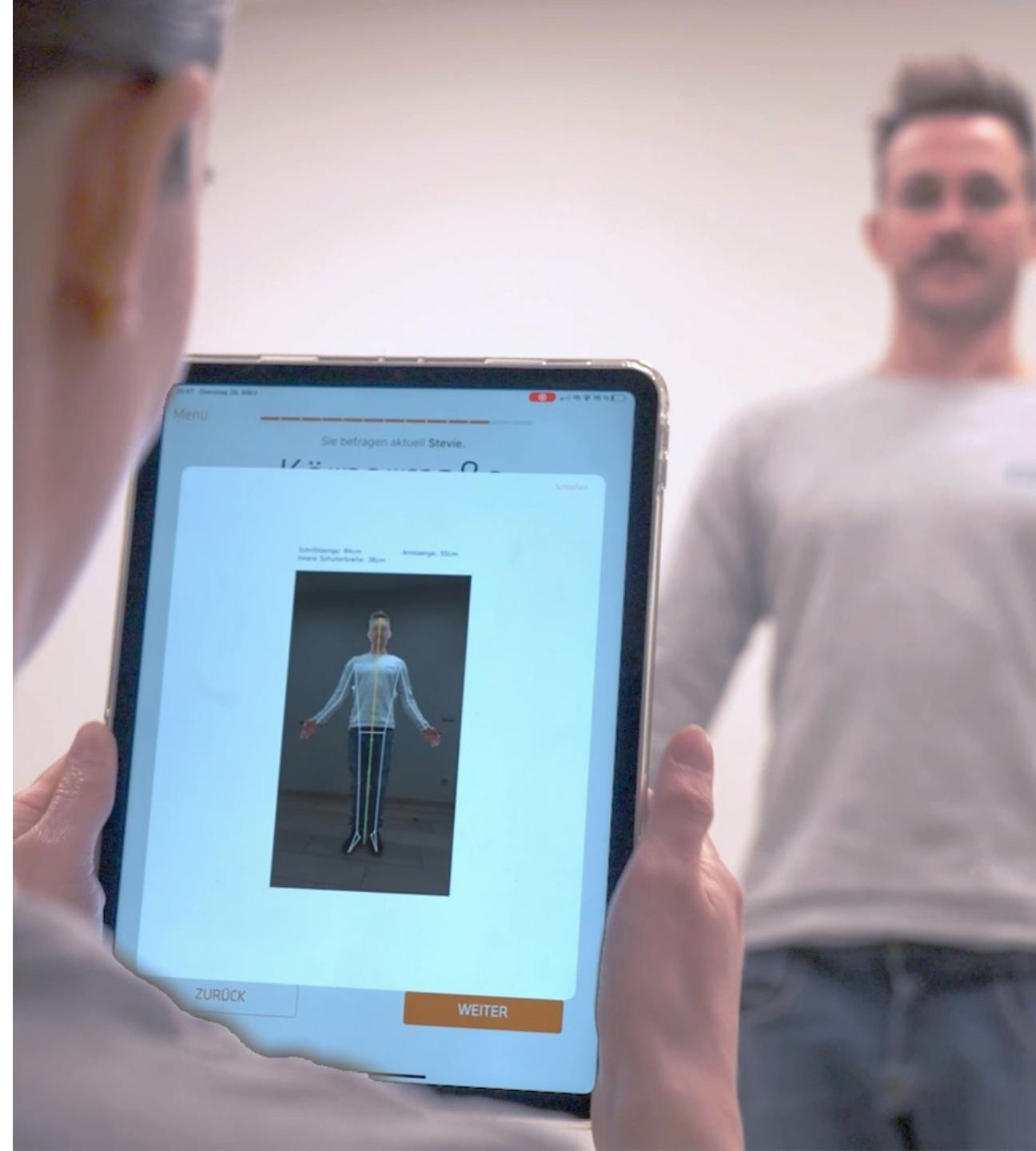
iPad app for 3D measurement and determination of suitable saddle, grip, etc.



→ Over 20.000 measurements worldwide

→ MVP after 2 months

A product for



Introduction to Flutter

Julian Hartl



Overview

Why Flutter?

Flutter Basics

Building a Chat-GPT Clone



Why Flutter?

What even is Flutter?

What even is Flutter?

“Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.”

What even is Flutter?

“Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.”

- Open source UI software development toolkit
- Supports many platforms
- Single codebase



Your experience with mobile development

Did you enjoy it? Why/Why not?

What does Flutter offer?

What does Flutter offer?

- Cross-Platform development (Android, iOS, Web, Windows, macOS, Linux, Fuchsia)

What does Flutter offer?

- Cross-Platform development (Android, iOS, Web, Windows, macOS, Linux, Fuchsia)



What does Flutter offer?

- Cross-Platform development (Android, iOS, Web, Windows, macOS, Linux, Fuchsia)
- Performant due to Ahead-of-Time (AOT) Compilation of Dart

What does Flutter offer?

- Cross-Platform development (Android, iOS, Web, Windows, macOS, Linux, Fuchsia)
- Performant due to Ahead-of-Time (AOT) Compilation of Dart
- Expressive and flexible UI (large set of prebuilt components)

What does Flutter offer?

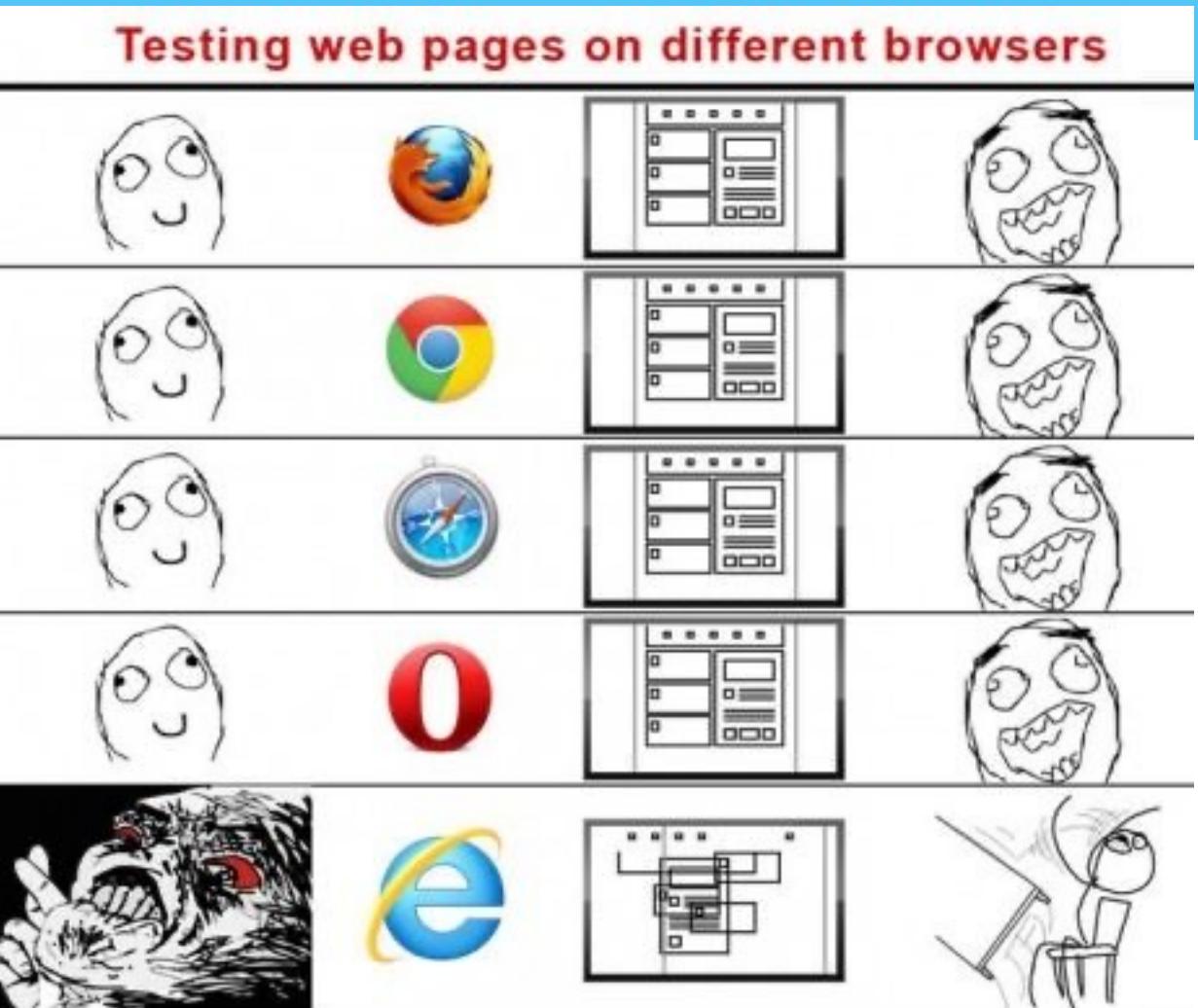
- Cross-Platform development (Android, iOS, Web, Windows, macOS, Linux, Fuchsia)
- Performant due to Ahead-of-Time (AOT) Compilation of Dart
- Expressive and flexible UI (large set of prebuilt components)
- Hot reload (instant UI updates)

What does Flutter offer?

- Cross-Platform development (Android, iOS, Web, Windows, macOS, Linux, Fuchsia)
- Performant due to Ahead-of-Time (AOT) Compilation of Dart
- Expressive and flexible UI (large set of prebuilt components)
- Hot reload (instant UI updates)
- Consistent behavior across all platforms

What does Flutter offer?

- Cross-Platform development (Android, iOS, macOS, Linux, Fuchsia)
- Performant due to Ahead-of-Time compilation
- Expressive and flexible UI (large ecosystem)
- Hot reload (instant UI updates)
- Consistent behavior across all platforms



What does Flutter offer?

- Cross-Platform development (Android, iOS, Web, Windows, macOS, Linux, Fuchsia)
- Performant due to Ahead-of-Time (AOT) Compilation of Dart
- Expressive and flexible UI (large set of prebuilt components)
- Hot reload (instant UI updates)
- Consistent behavior across all platforms
- Strong community and ecosystem (wide adoption in the industry)

Applications in the wild





“Saved about 60-70% of their
engineer’s time”



“Saved about 60-70% of their
engineer’s time”



“Time saving & code-sharing”



“Saved about 60-70% of their engineer’s time”



“Time saving & code-sharing”



“No more troubleshooting cross-platform functionality”



“Saved about 60-70% of their engineer’s time”



“No more troubleshooting cross-platform functionality”



“Time saving & code-sharing”



“Build & design once”

Summary

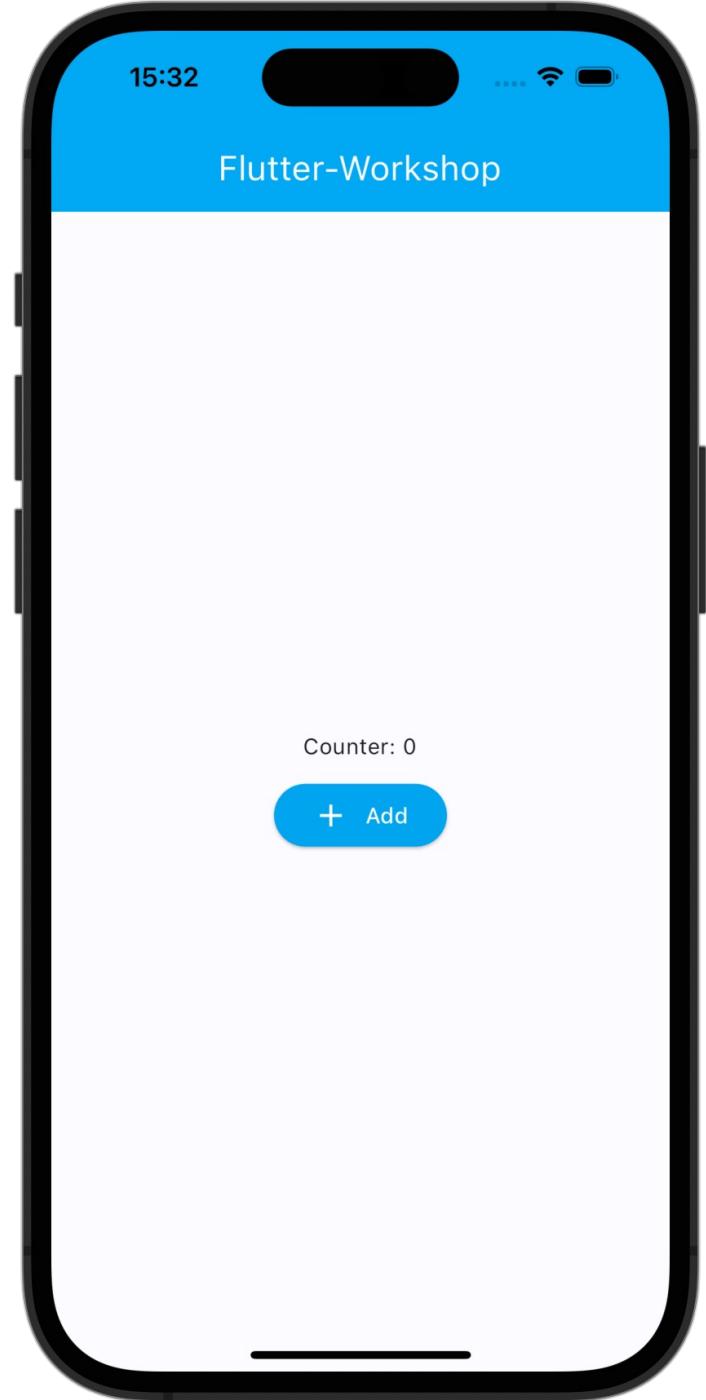


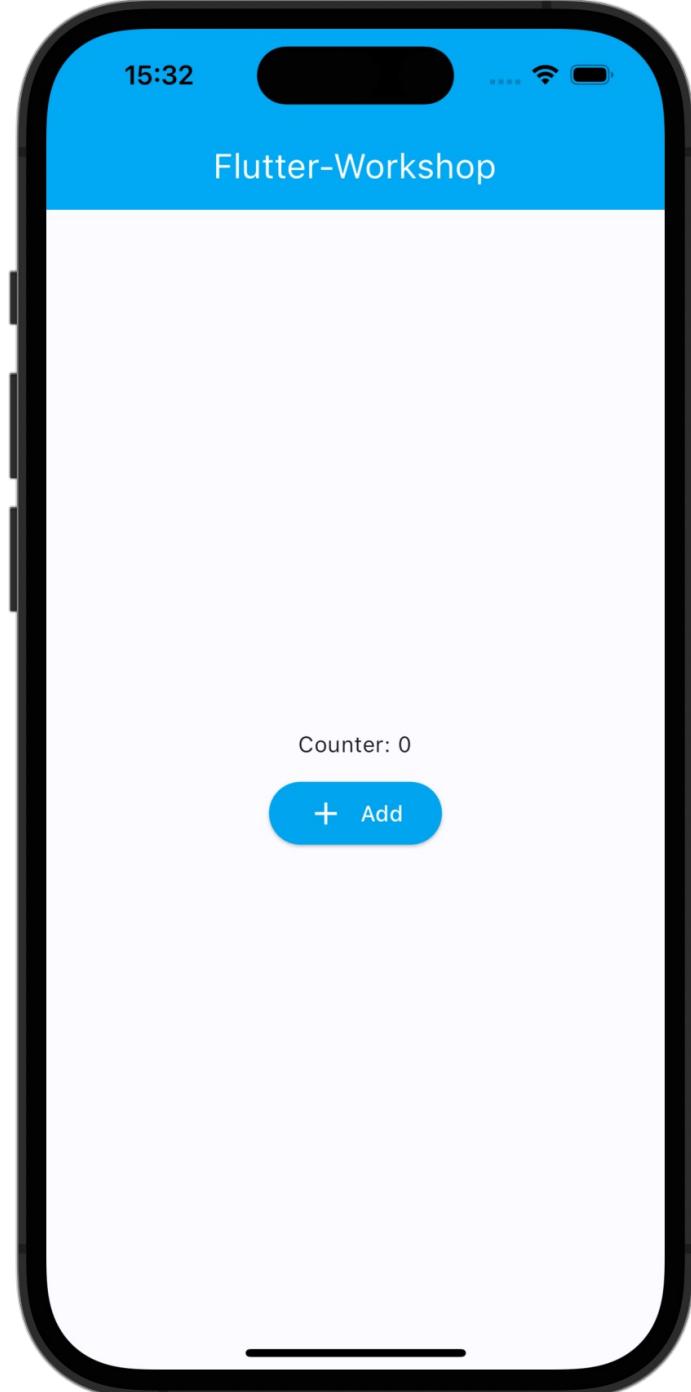
- Build once for all platforms
- Native performance
- Rich ecosystem & wide adoption

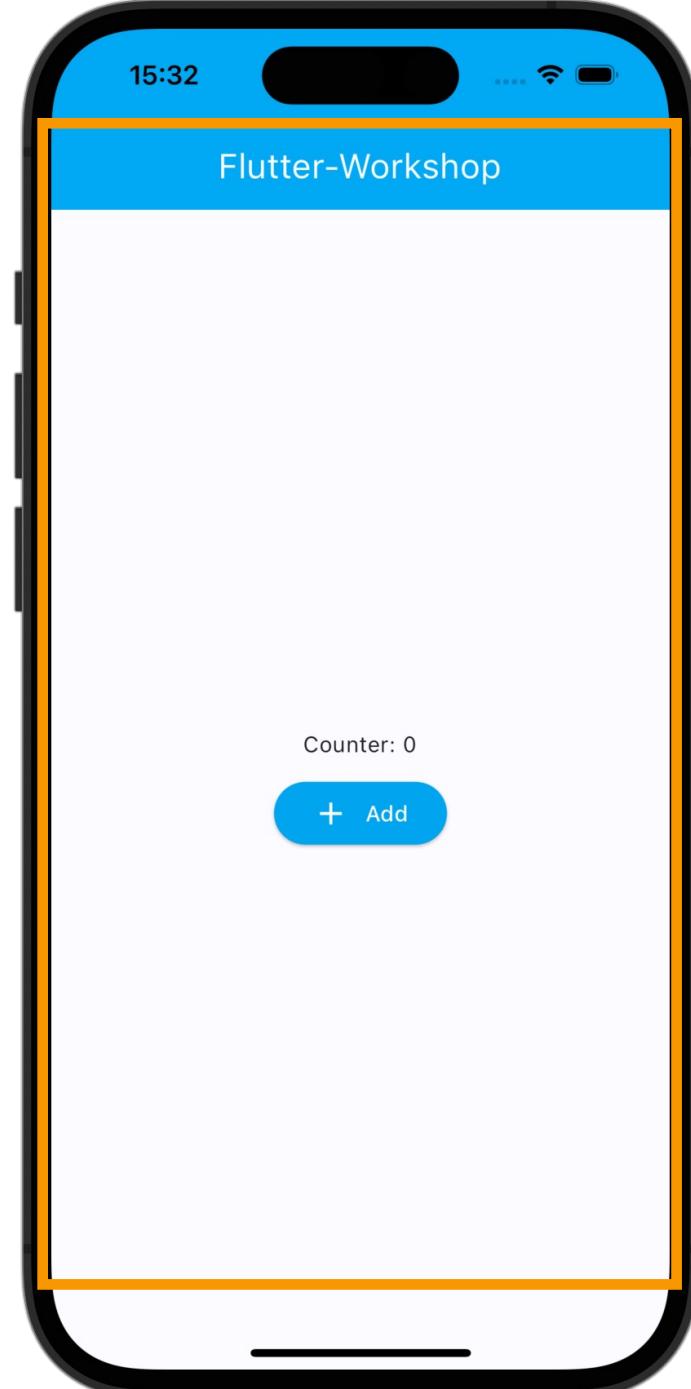
Flutter Basics

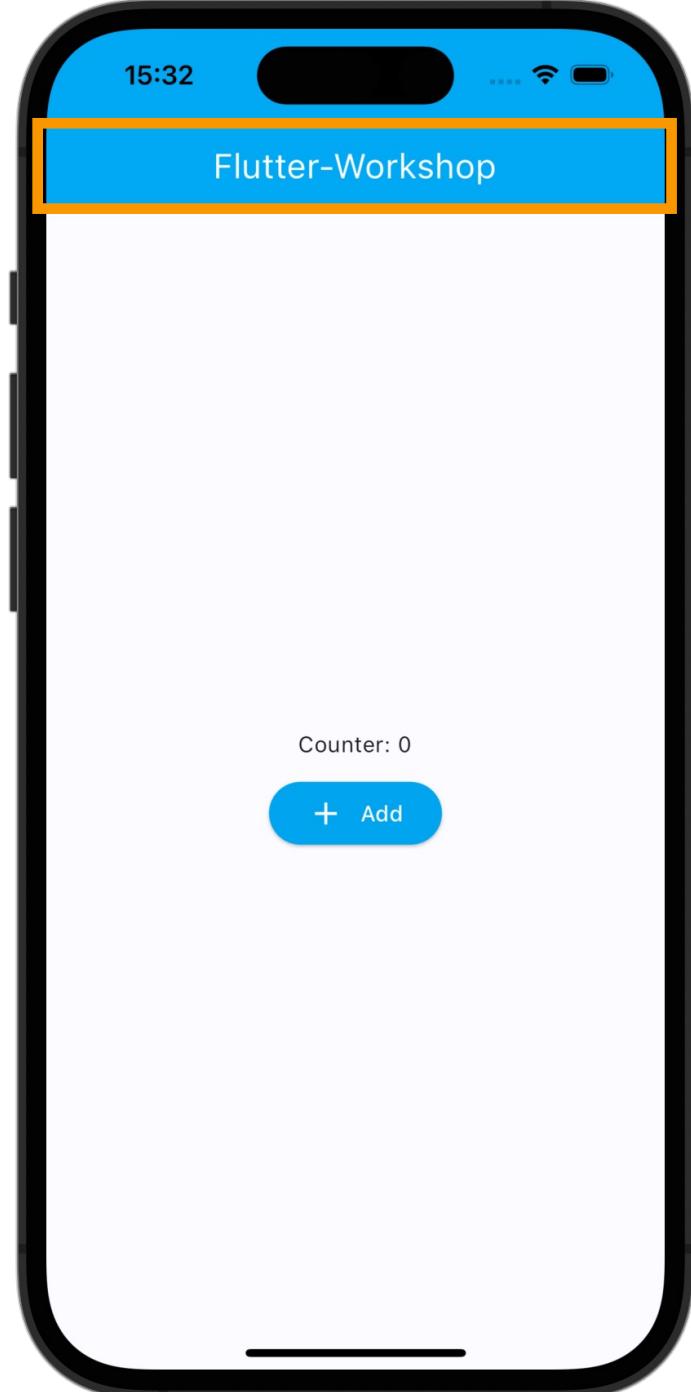
Widgets

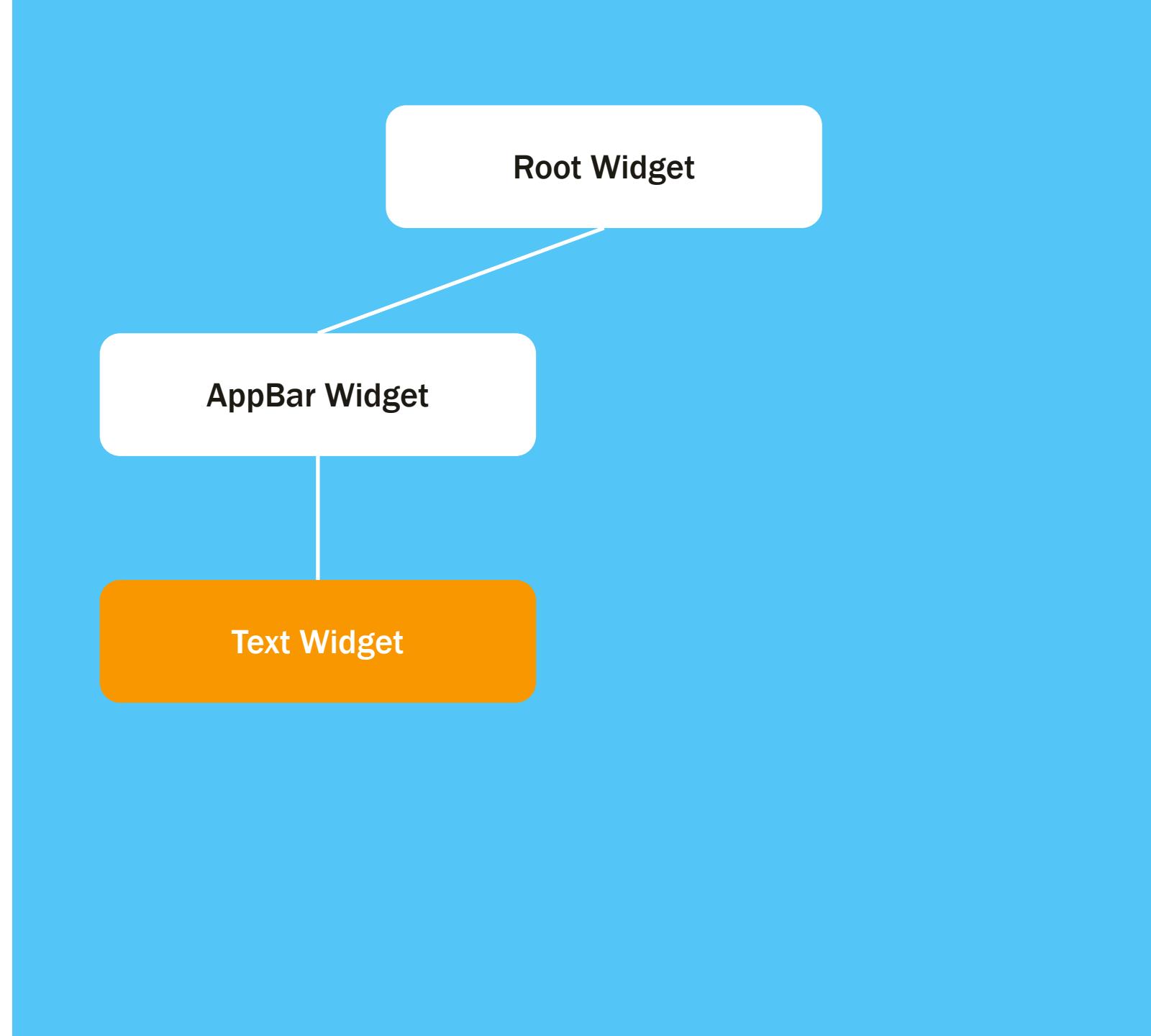
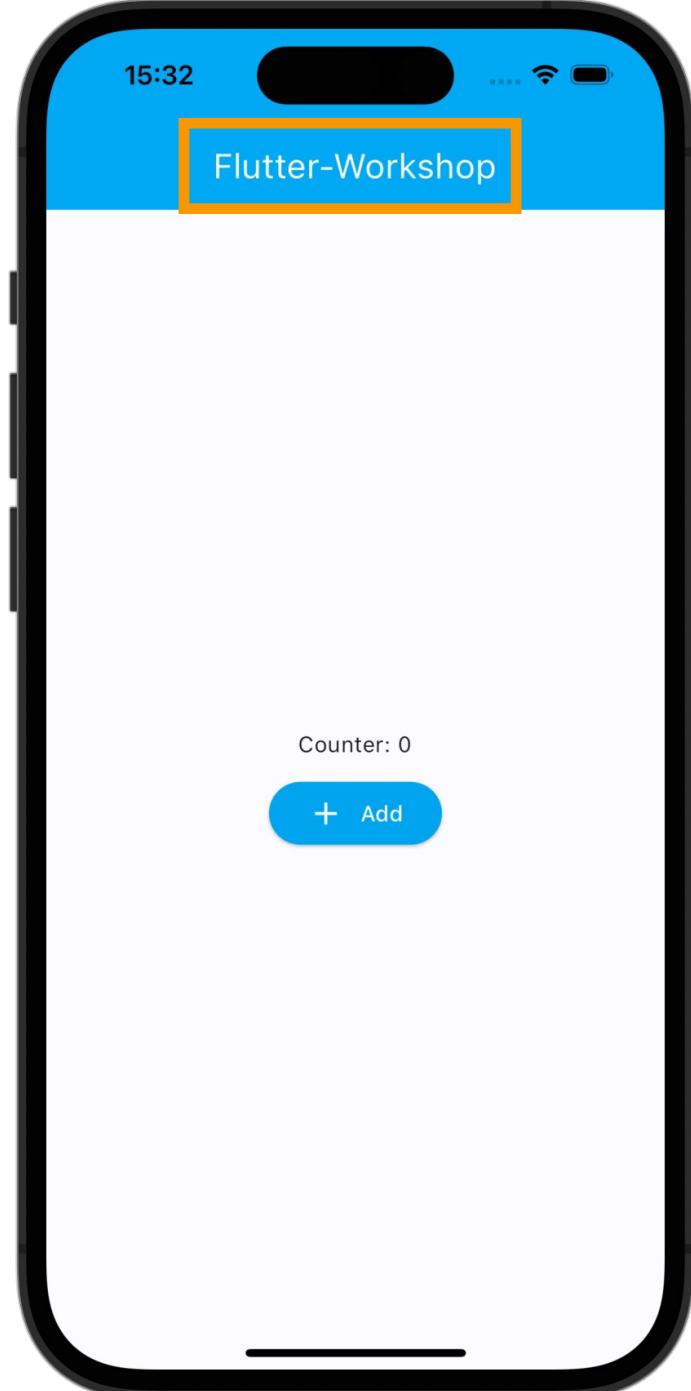


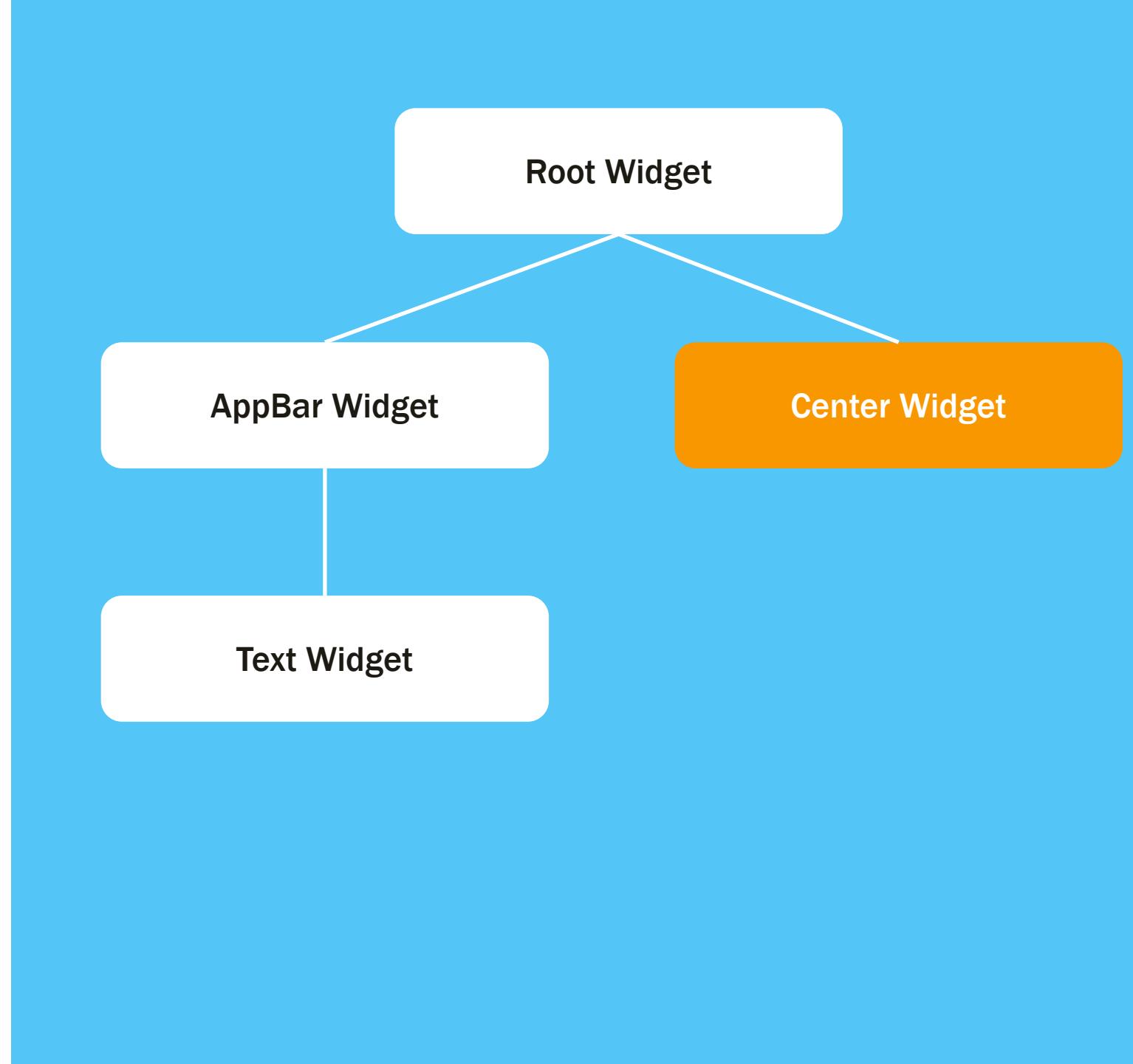
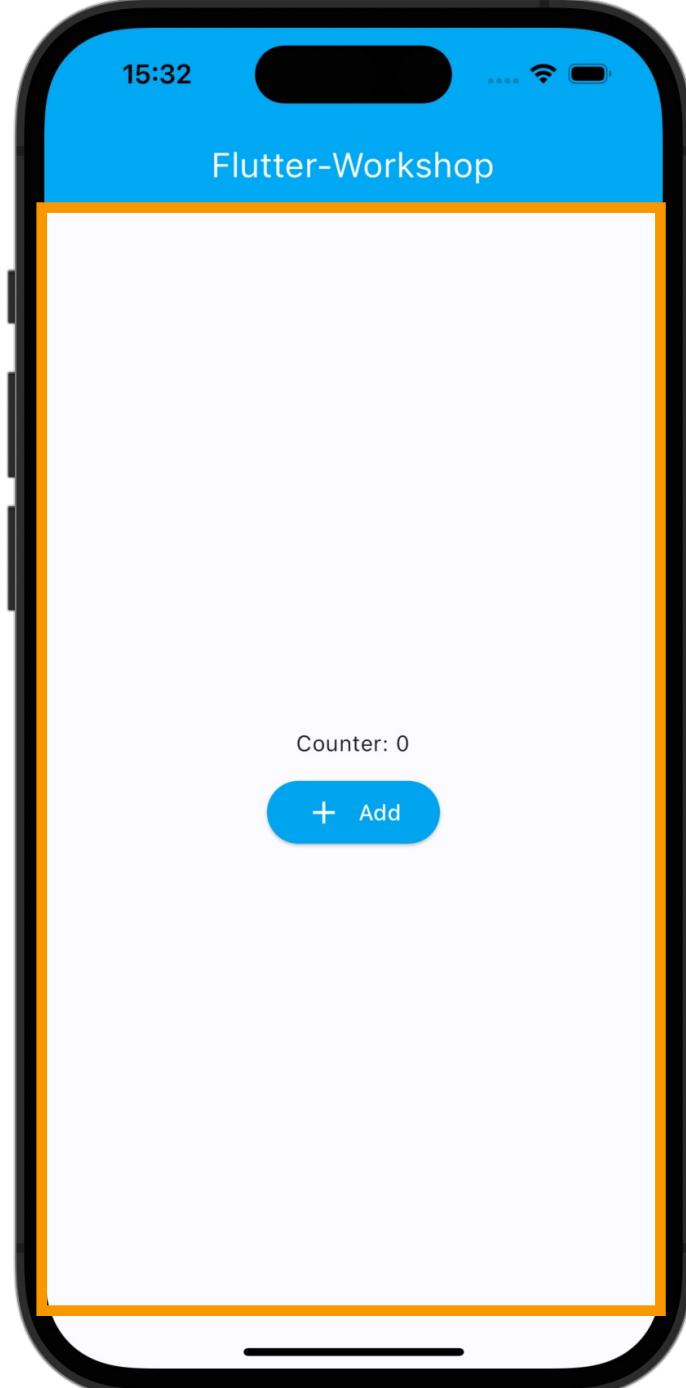


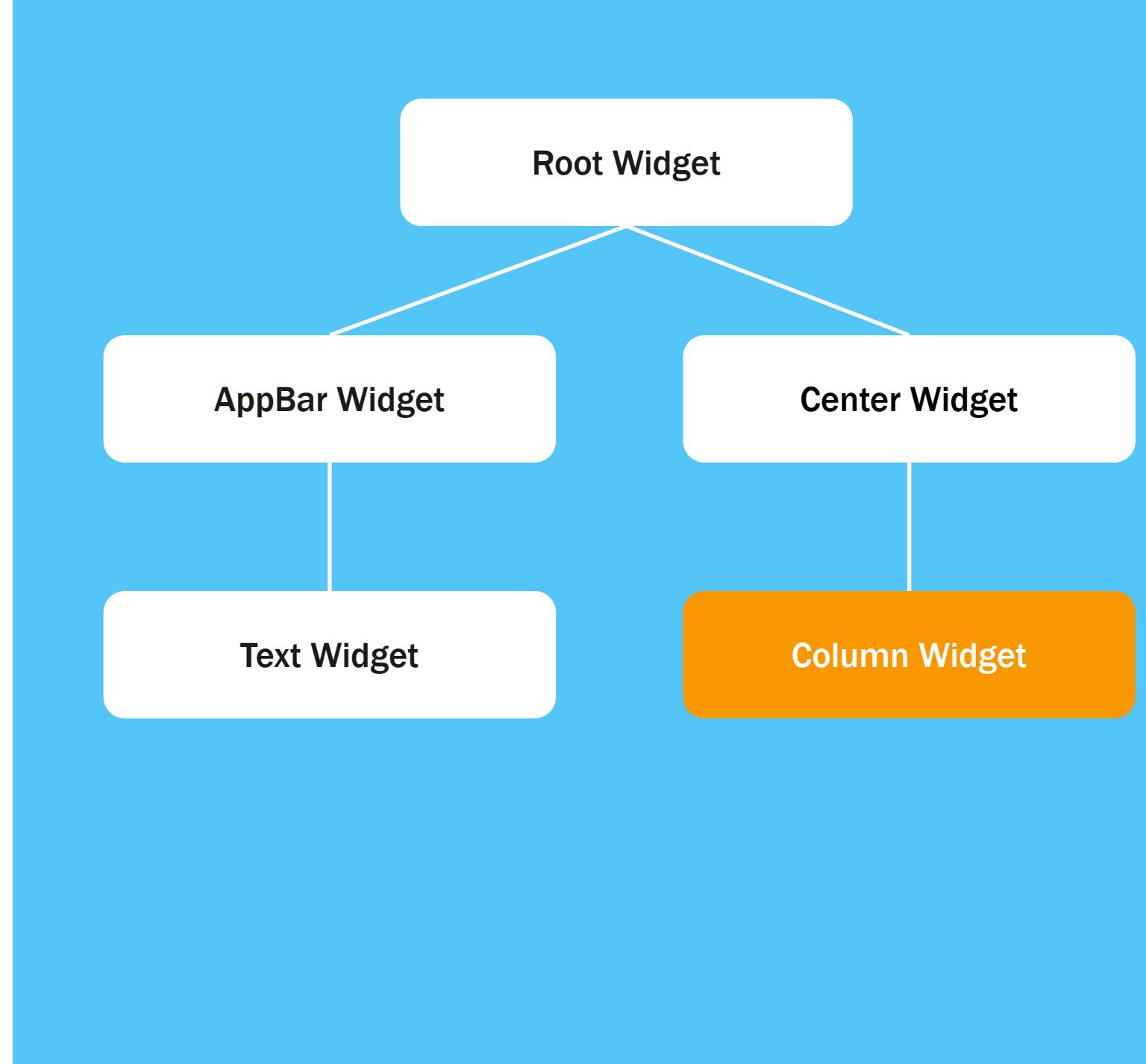
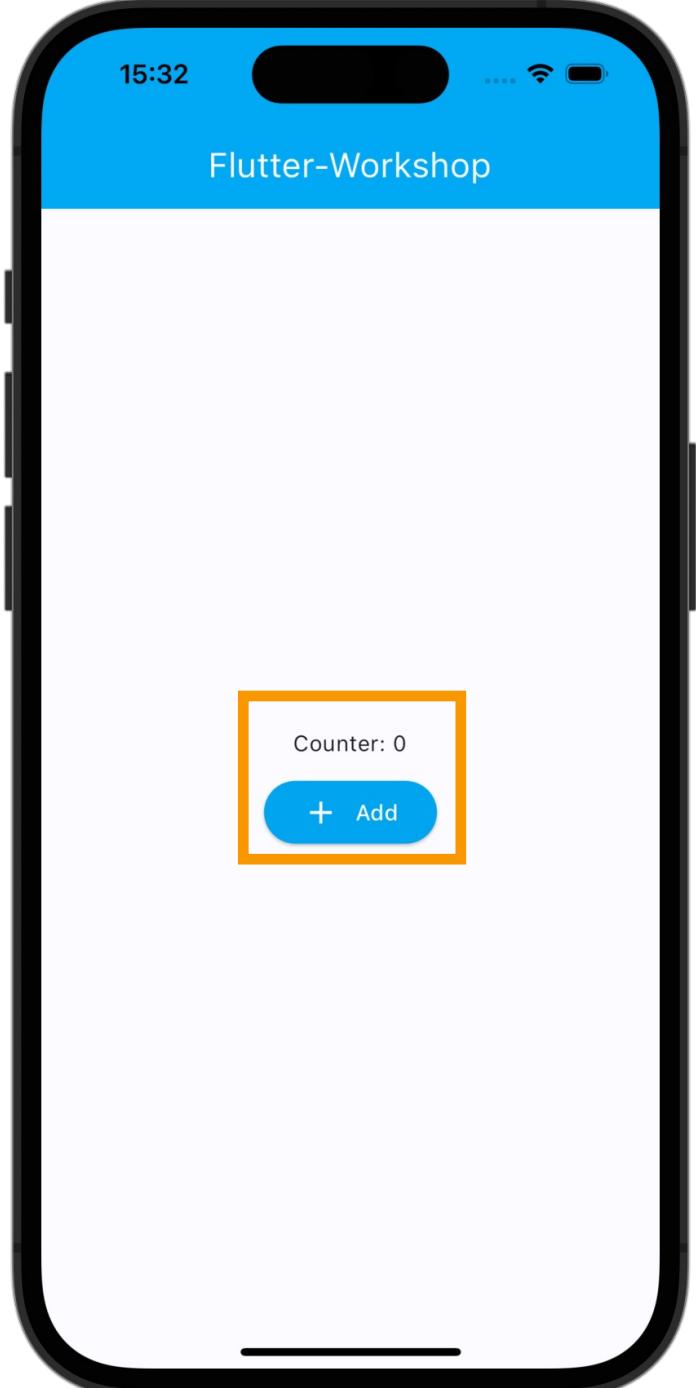


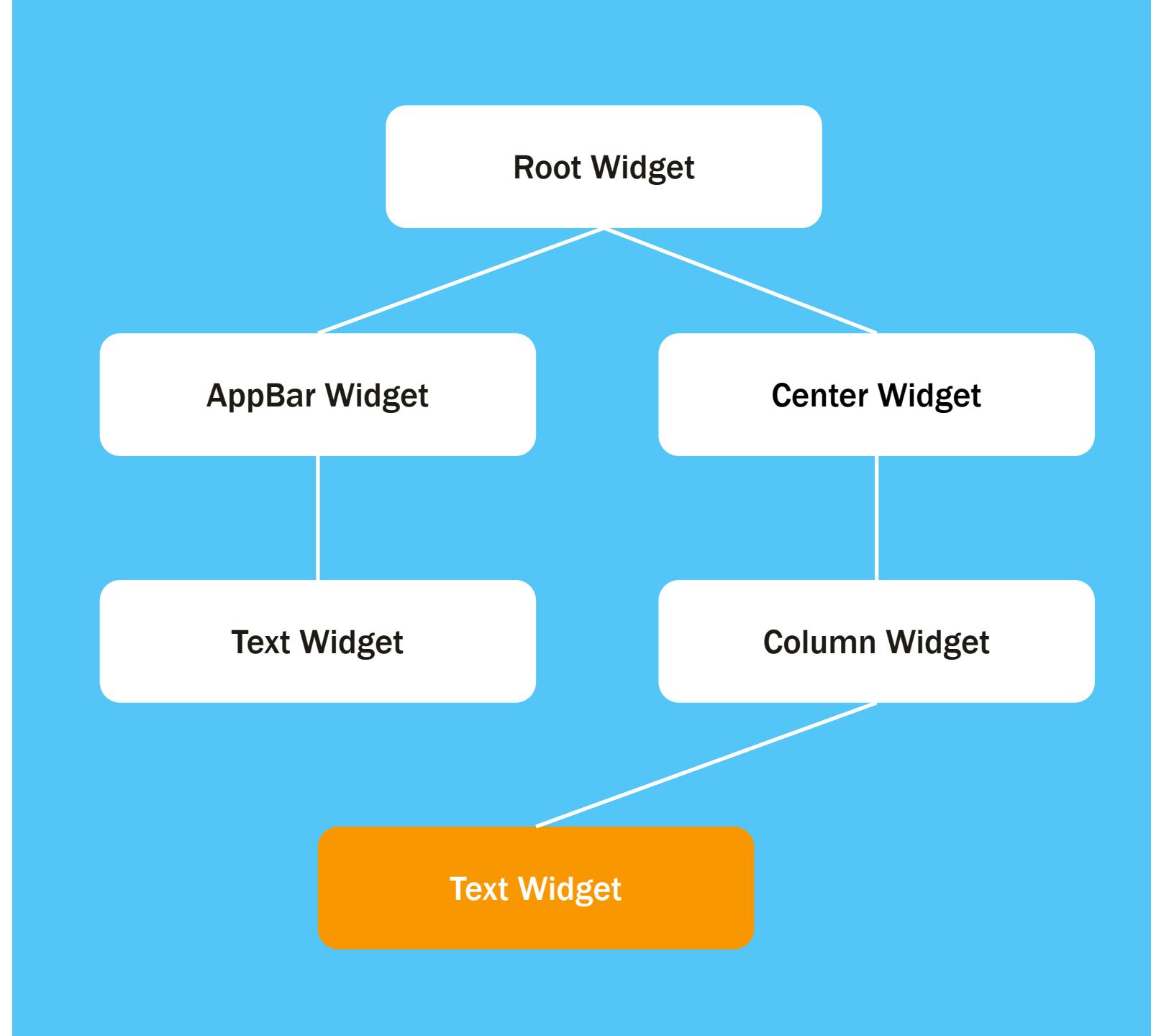
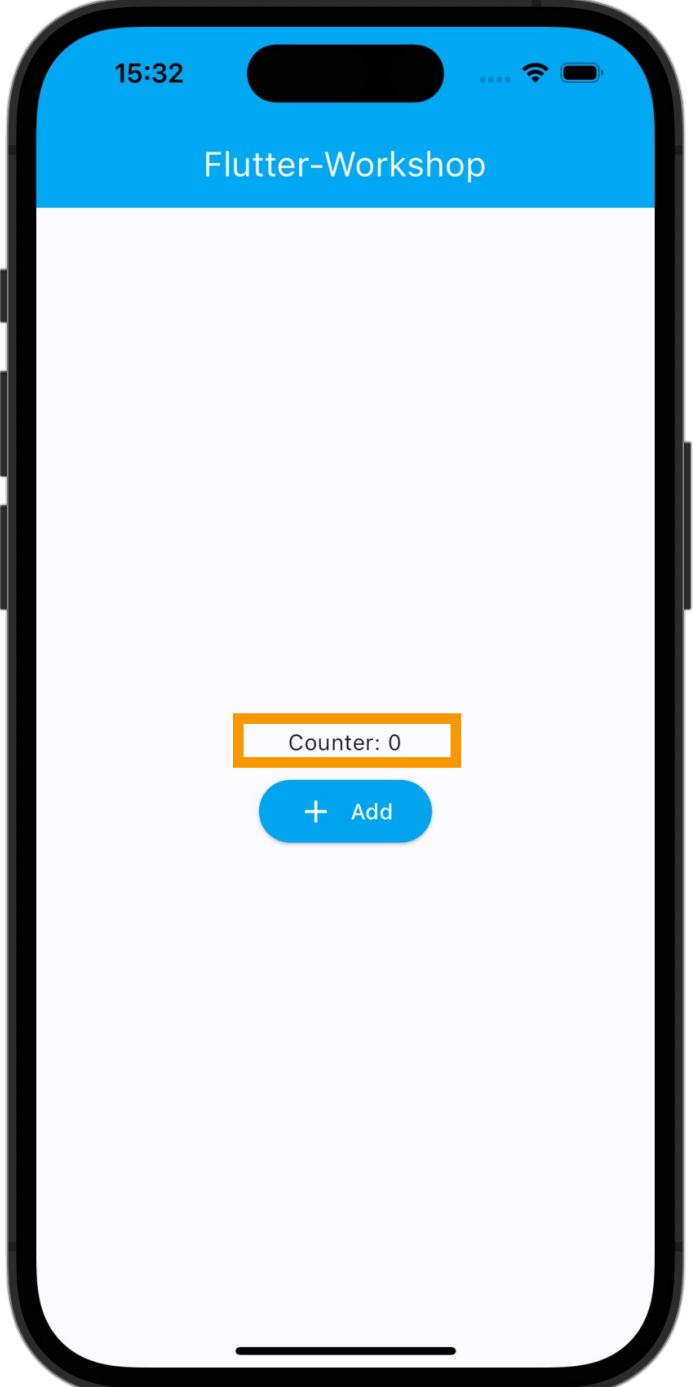


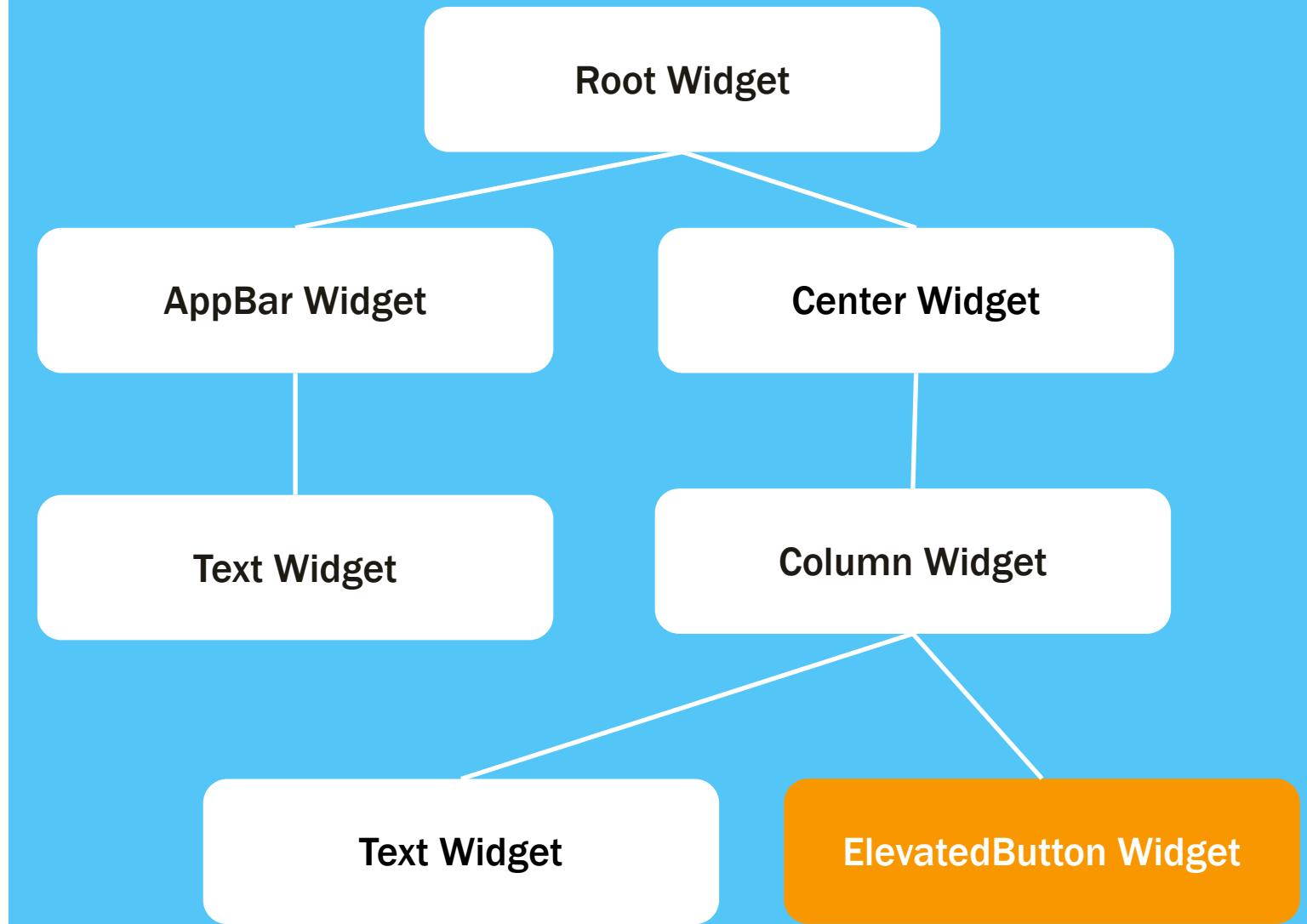
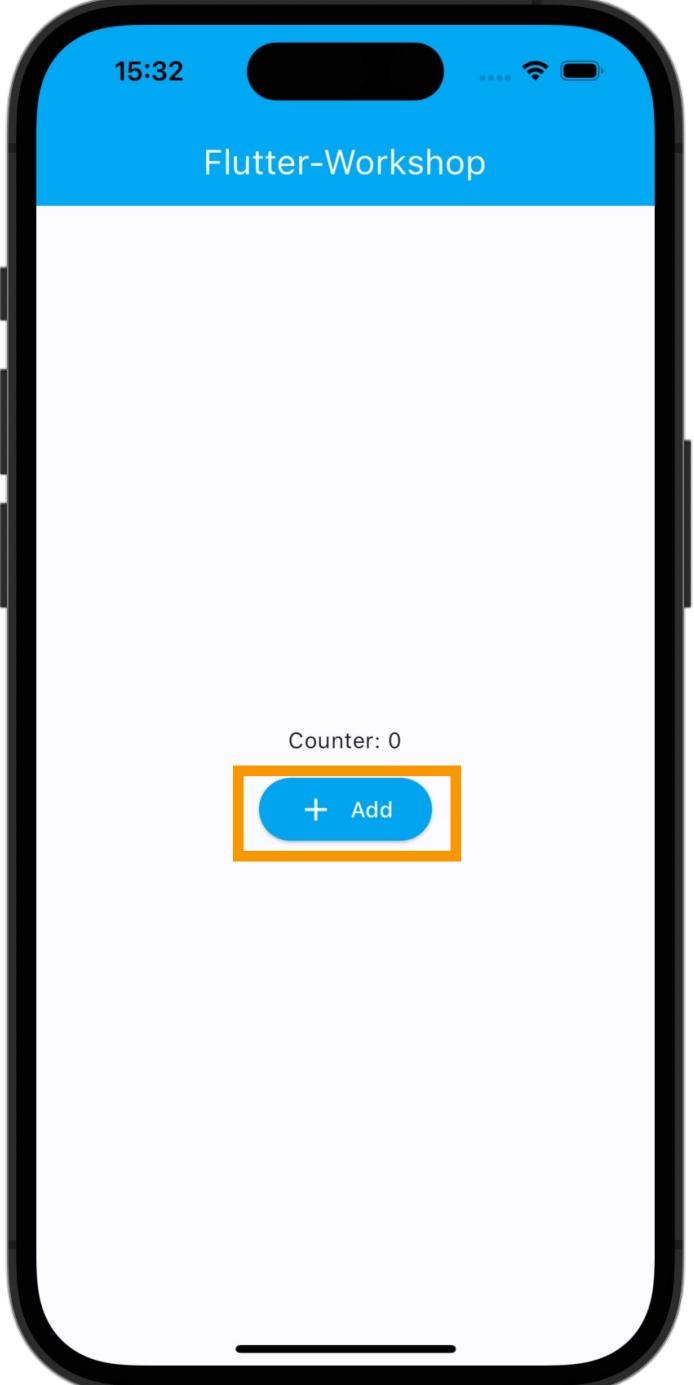


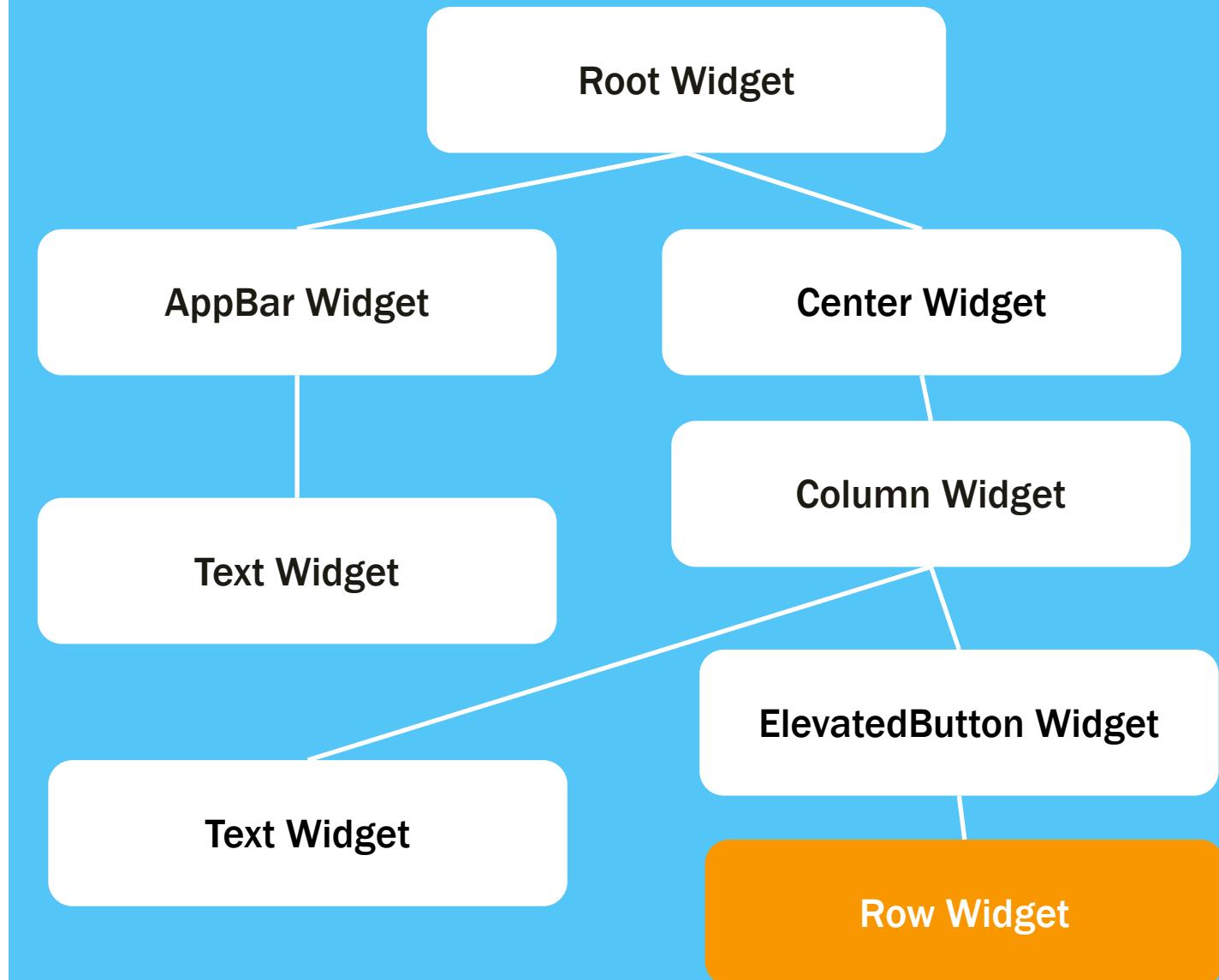
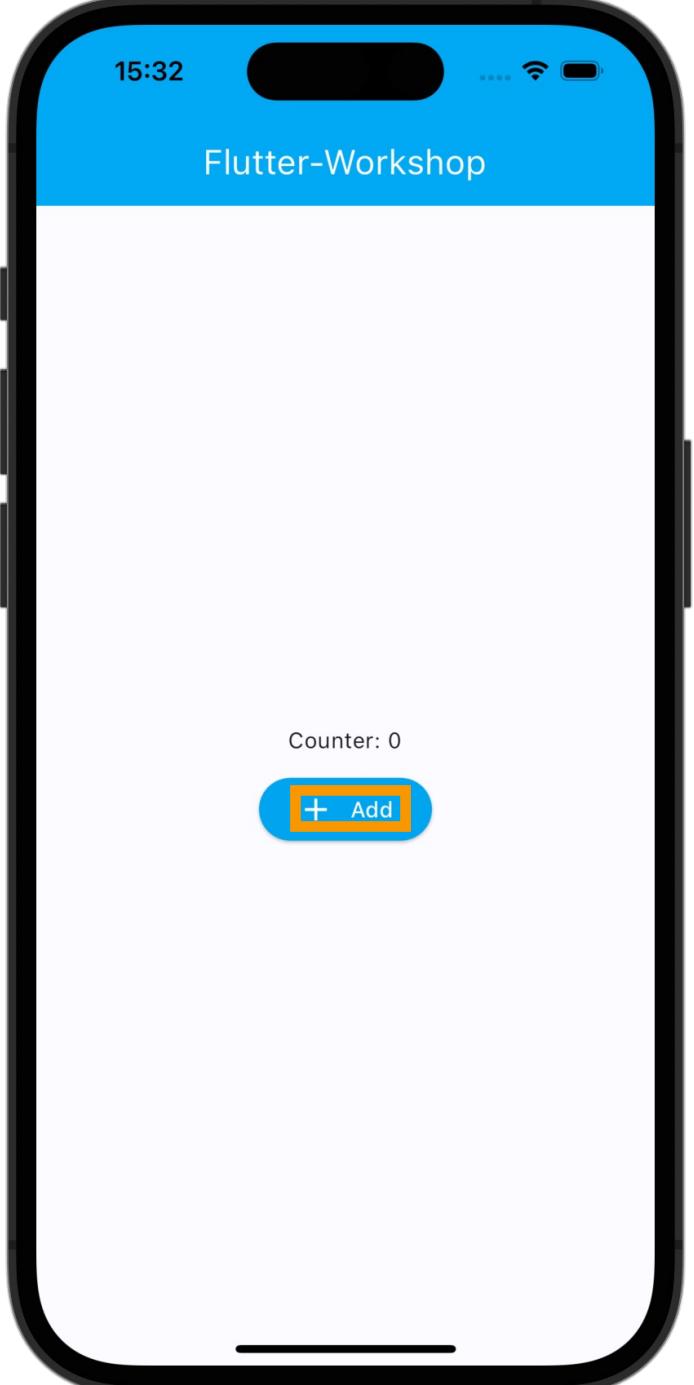


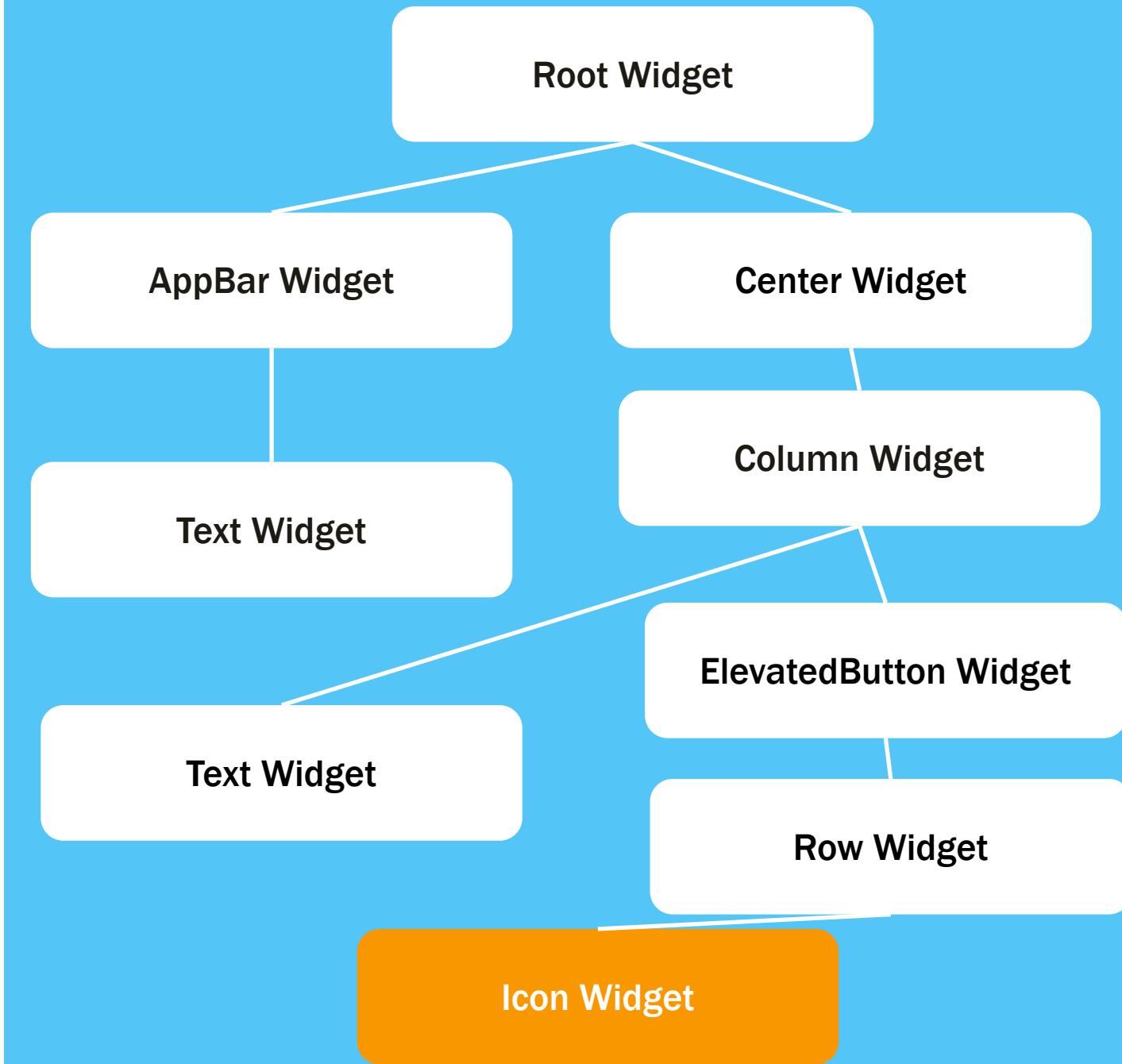
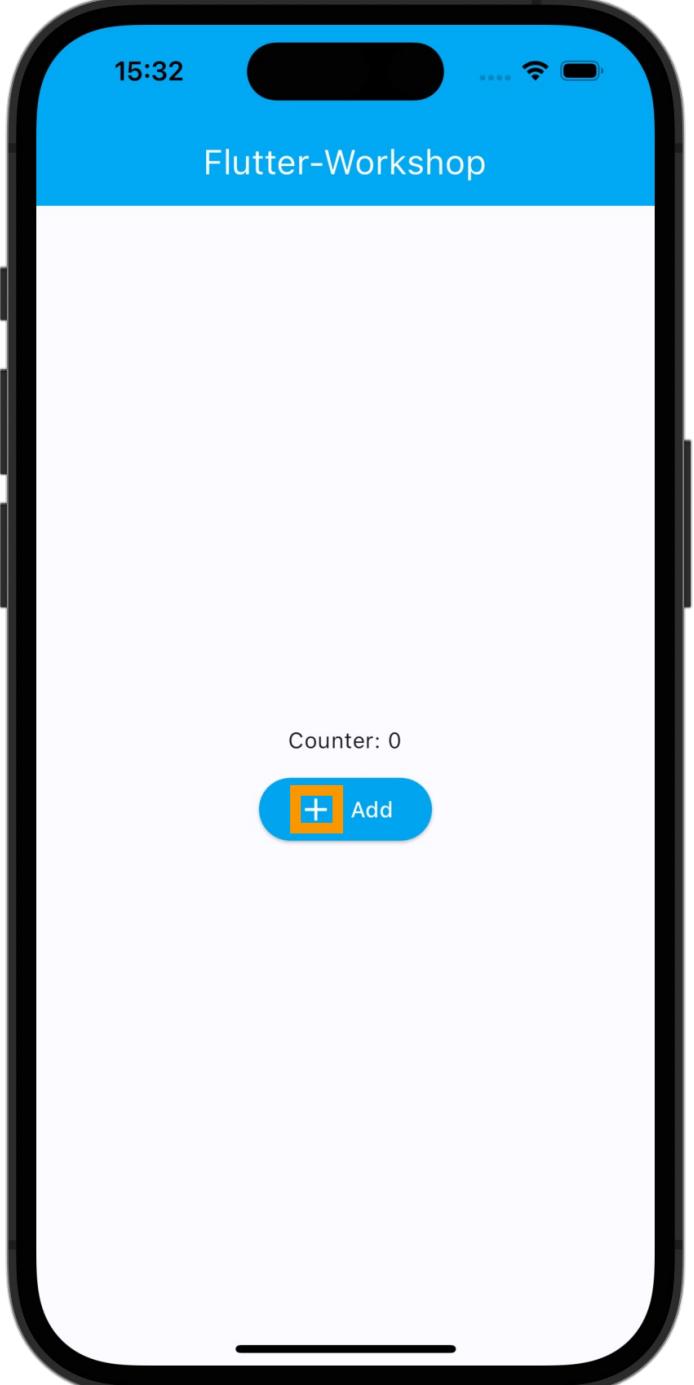


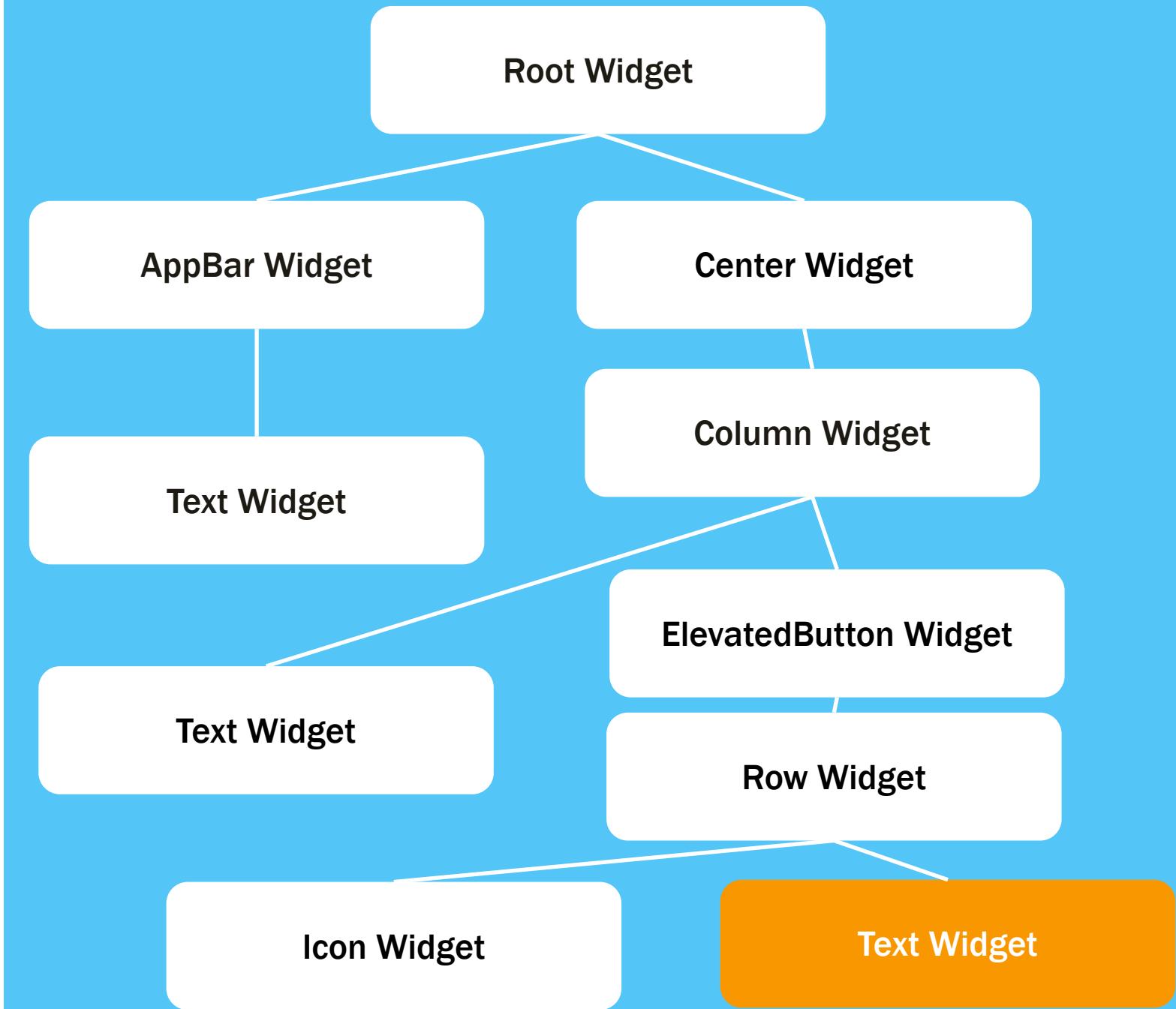
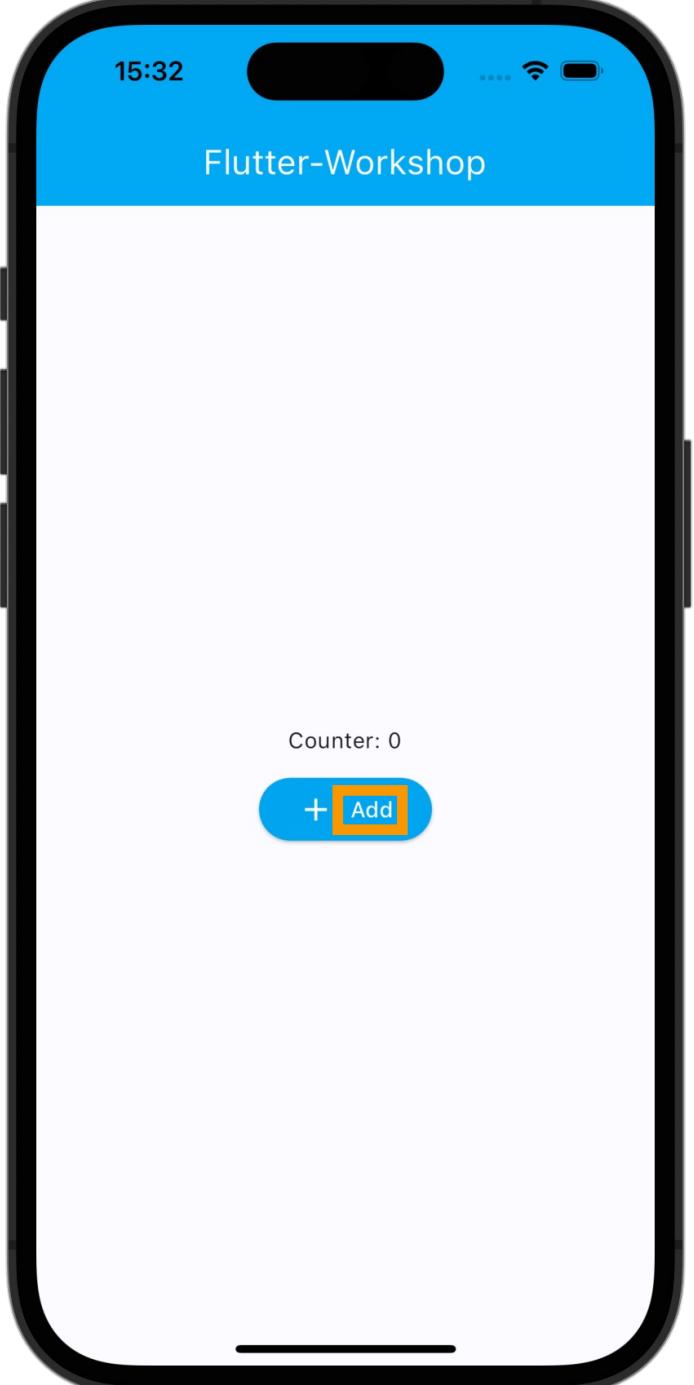


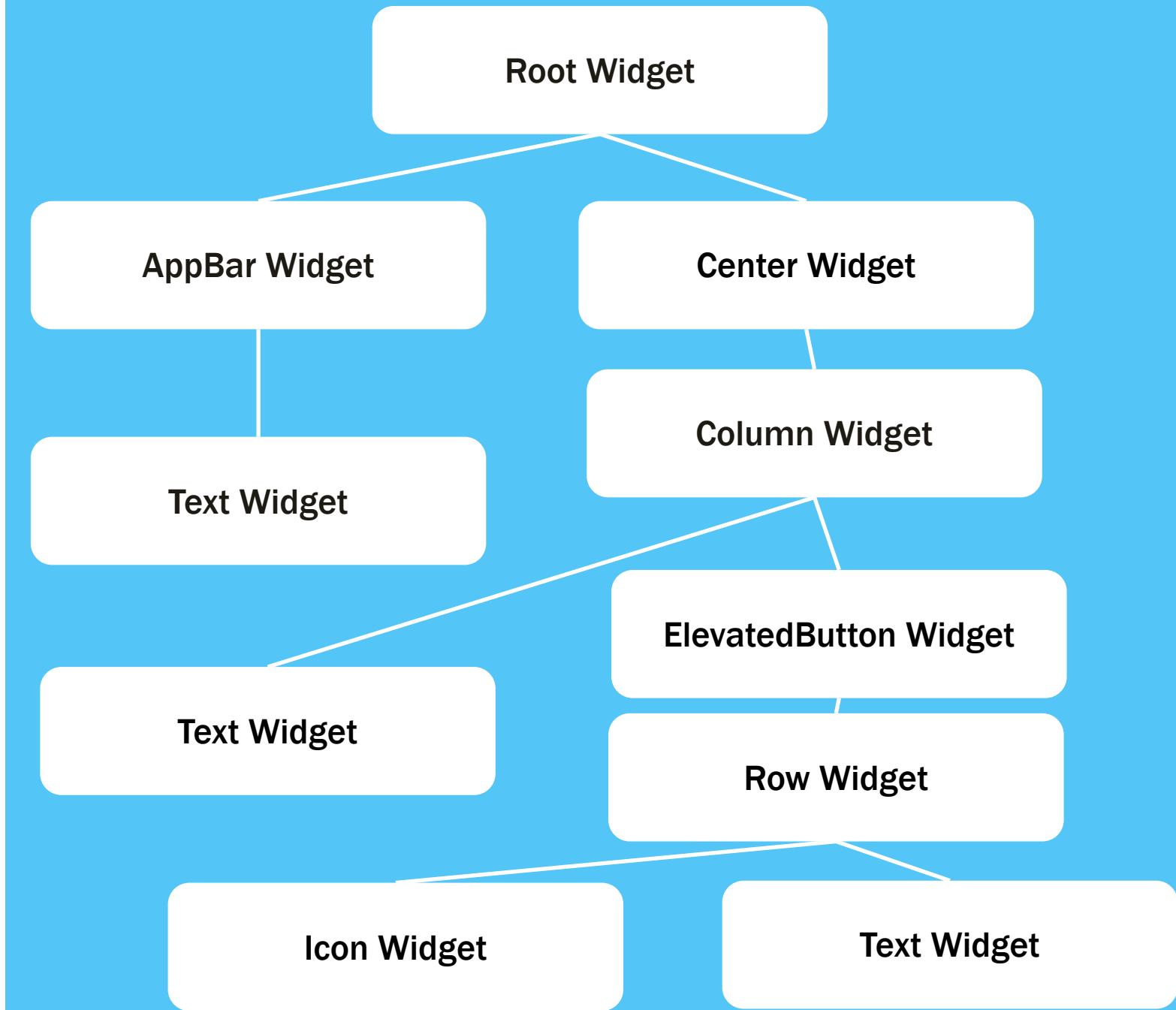
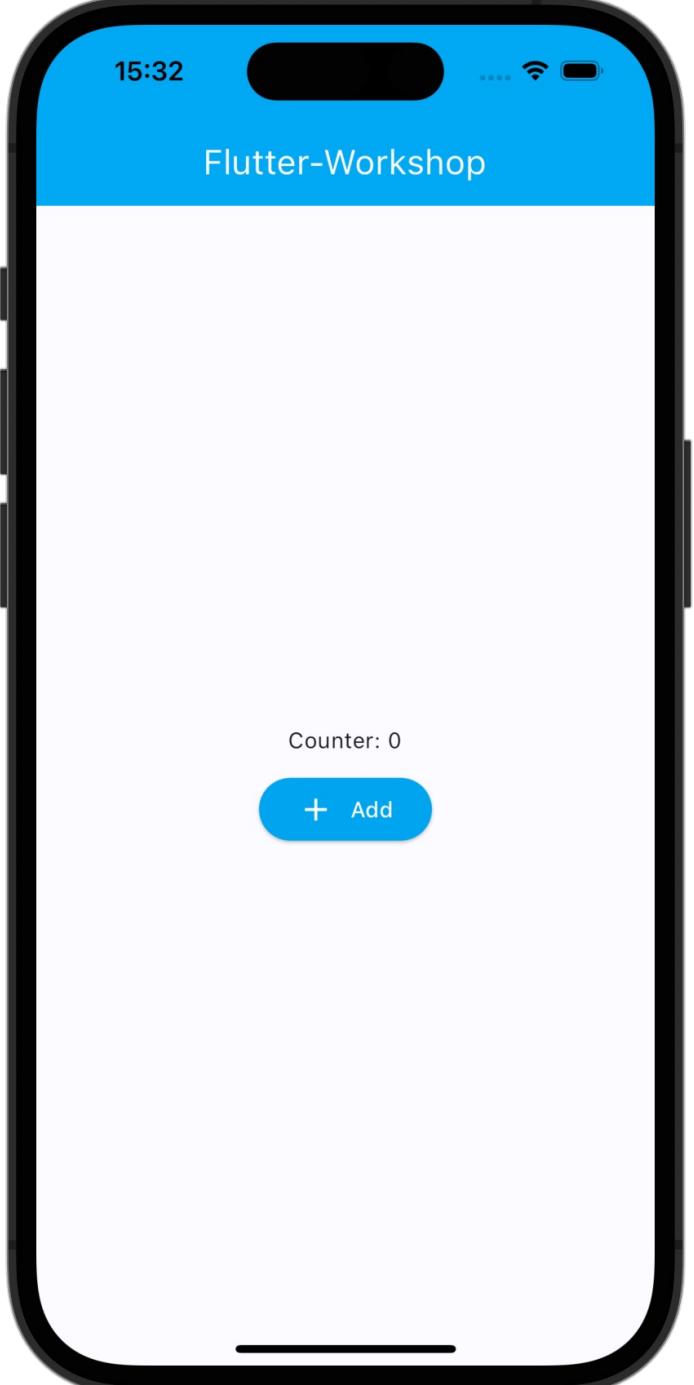


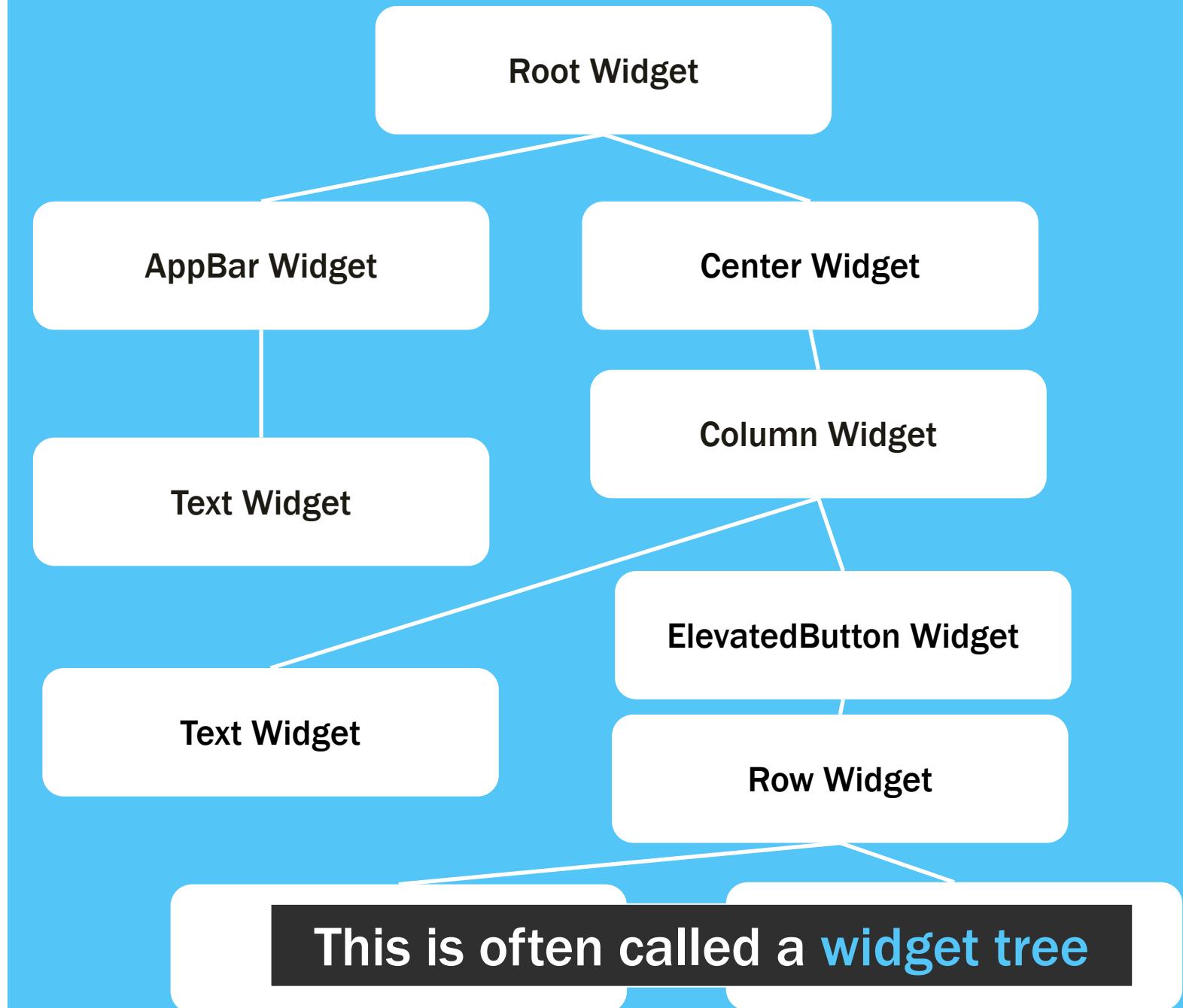
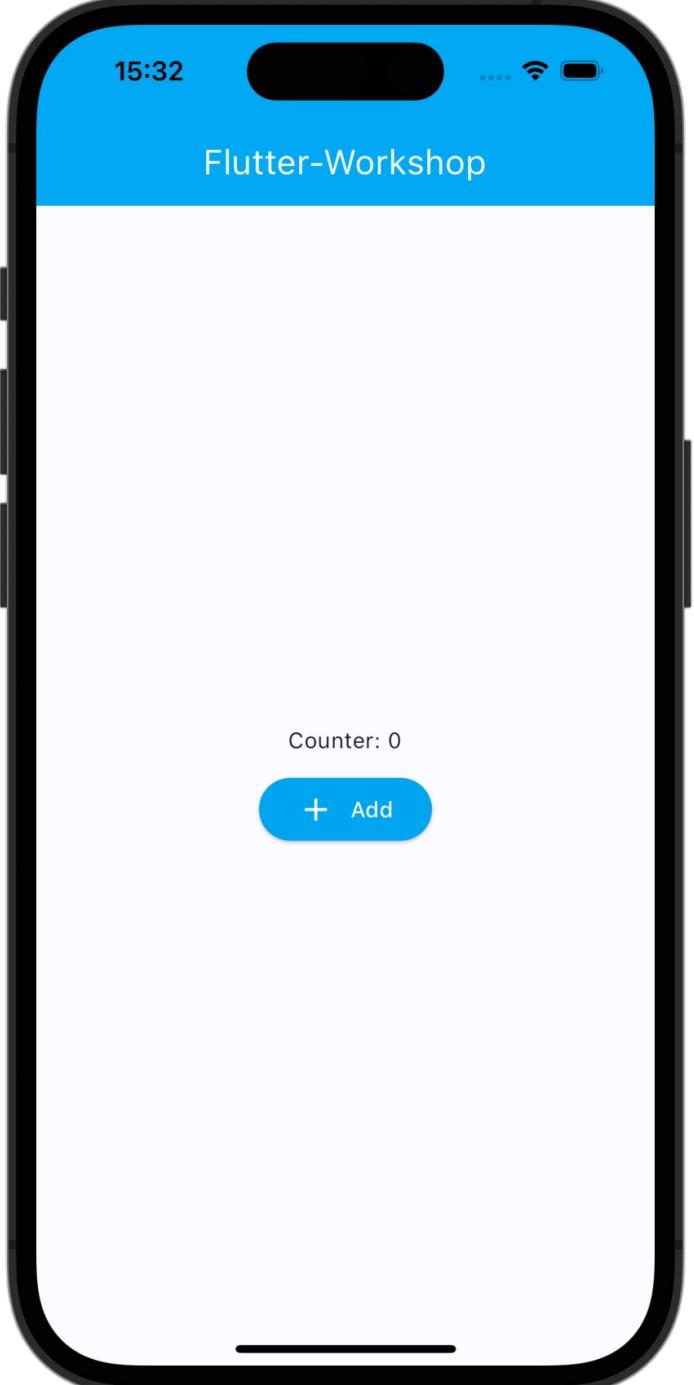












Widget Properties

Properties

Text Widget

ElevatedButton
Widget

Column Widget

AppBar Widget

Properties

Text Widget

ElevatedButton
Widget

Column Widget

AppBar Widget

style

textAlign

overflow

maxLines

...

Properties

Text Widget

style
textAlign
overflow
maxLines

...

ElevatedButton
Widget

style
onPressed

...

Column Widget

AppBar Widget

Properties

Text Widget

style
textAlign
overflow
maxLines

...

ElevatedButton
Widget

style
onPressed

...

Column Widget

mainAxisAlignment
crossAxisAlignment

...

AppBar Widget

Properties

Text Widget

style
textAlign
overflow
maxLines

...

ElevatedButton
Widget

style
onPressed

...

Column Widget

mainAxisAlignment
crossAxisAlignment

...

AppBar Widget

title
backgroundColor
leading

...

Summary

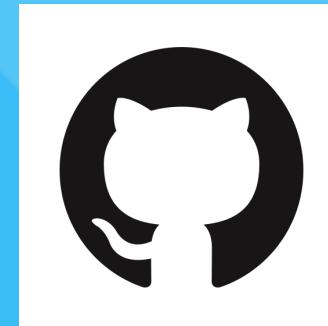


- “Everything is a widget”
- A page is just a tree of widgets (including itself)
- Widgets can be customized using their properties

Chat-GPT Clone

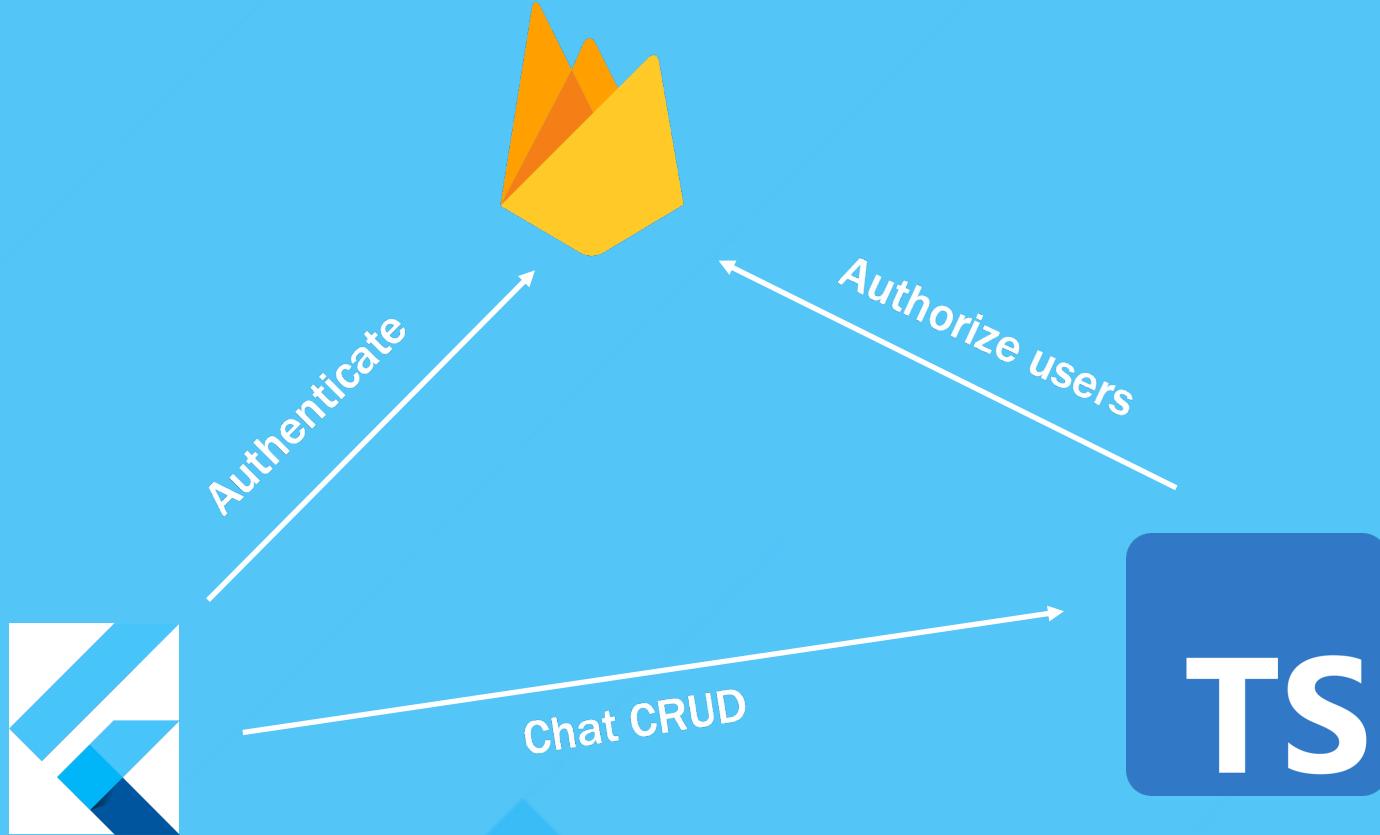
Demo

Chat-GPT Clone



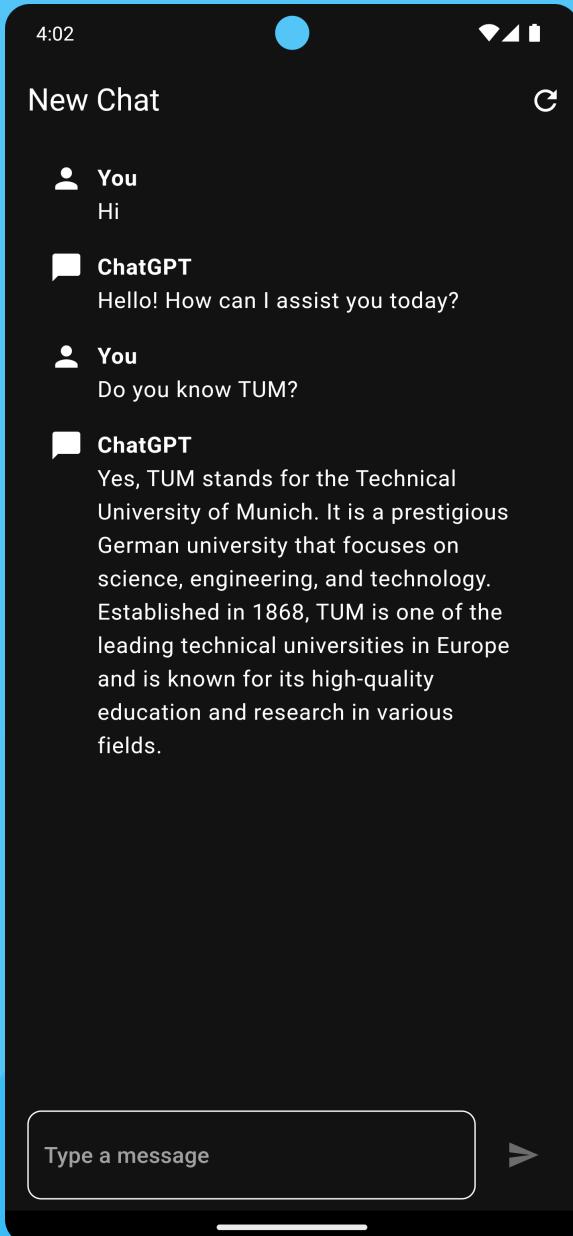
Architecture

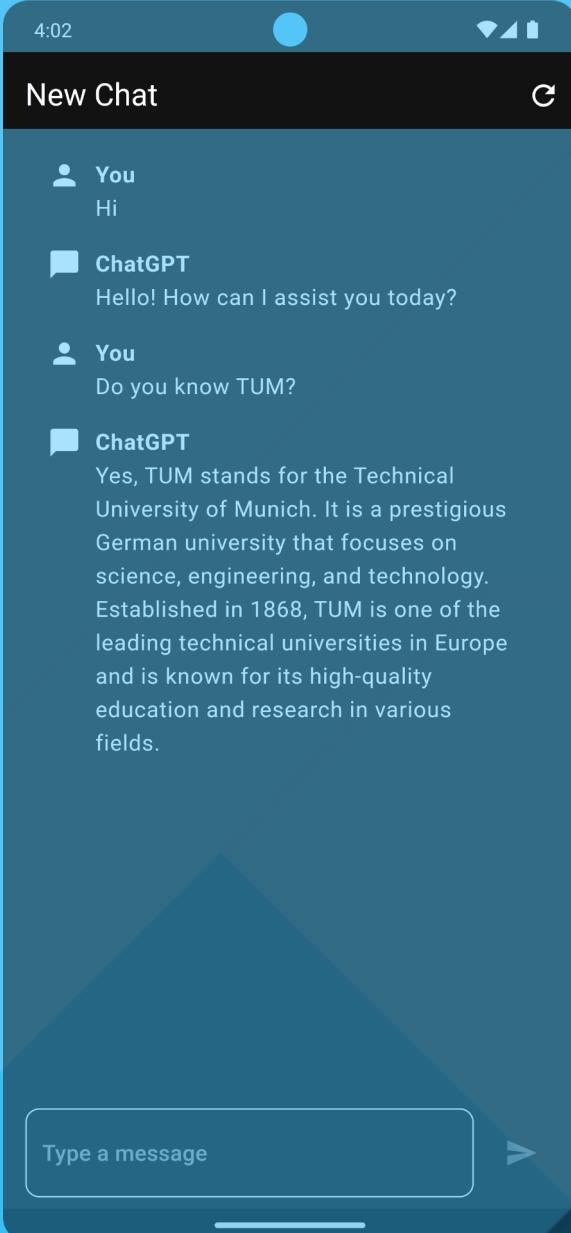
Chat-GPT Clone

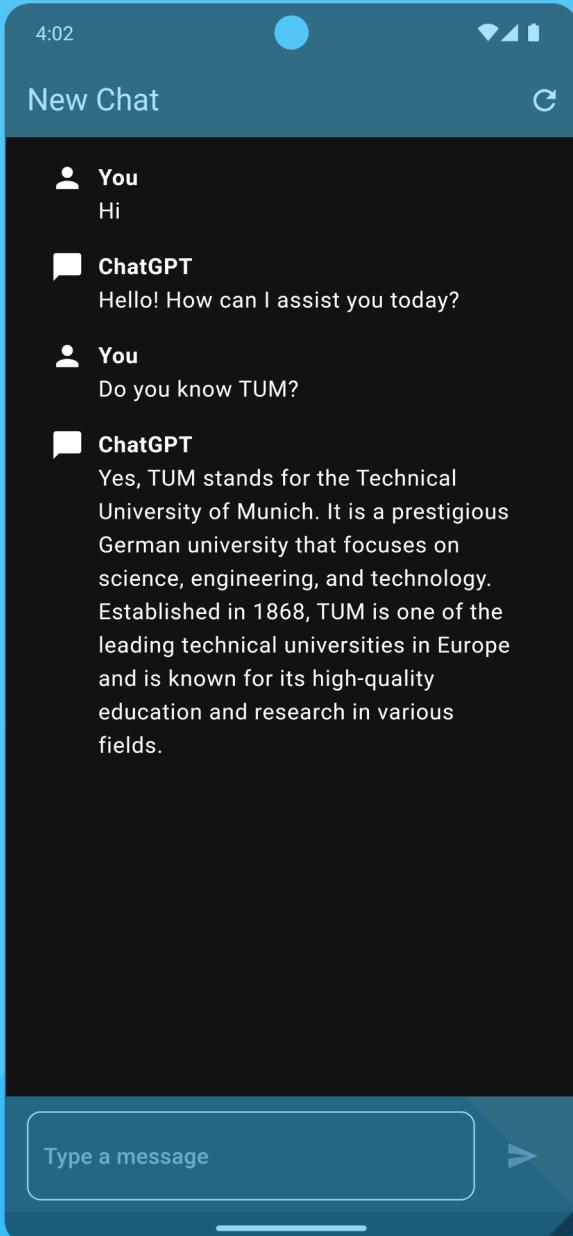


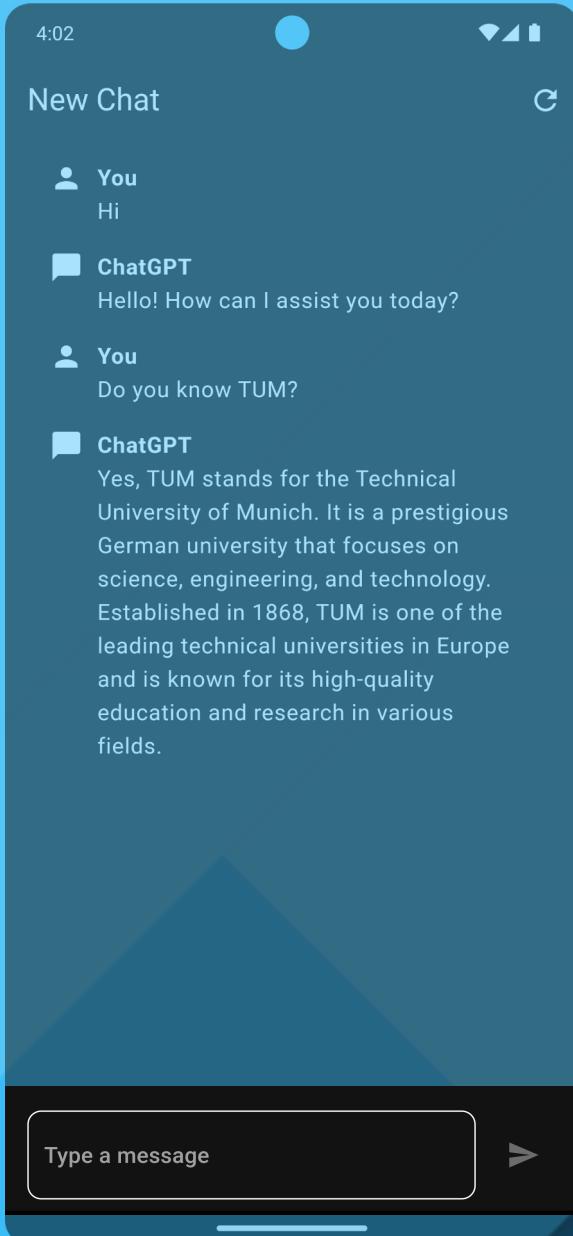
Design

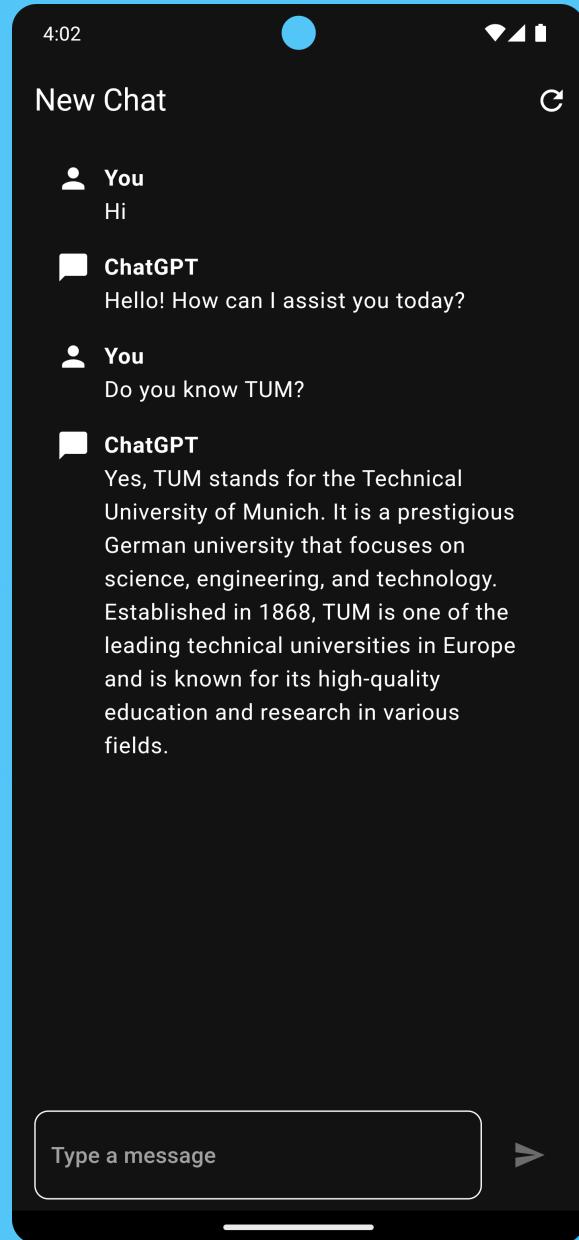
Chat-GPT Clone



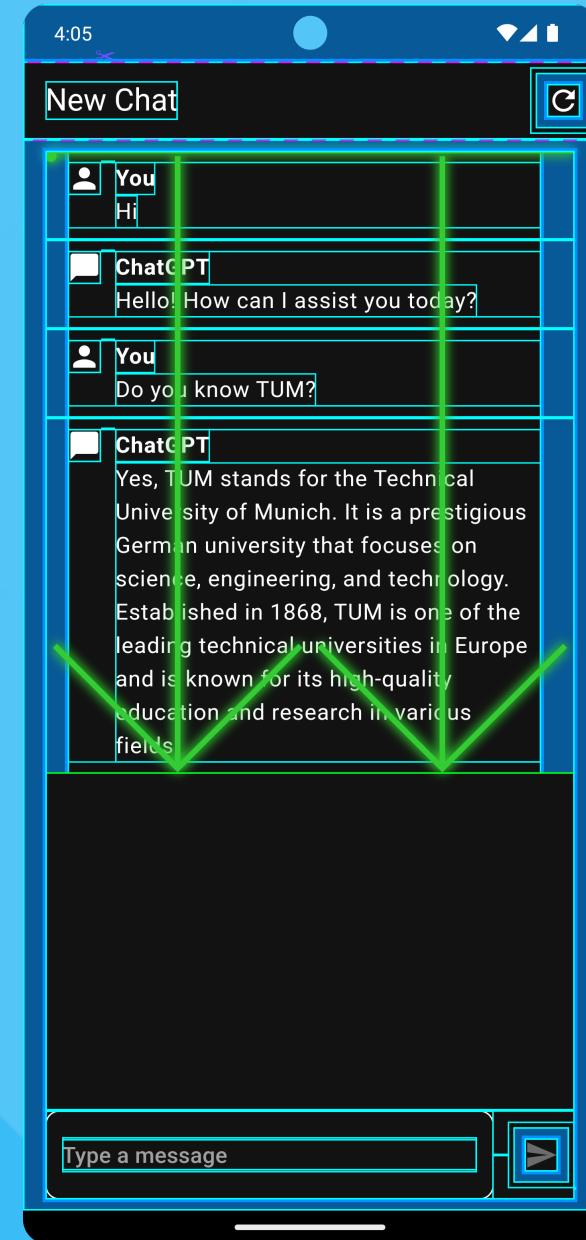


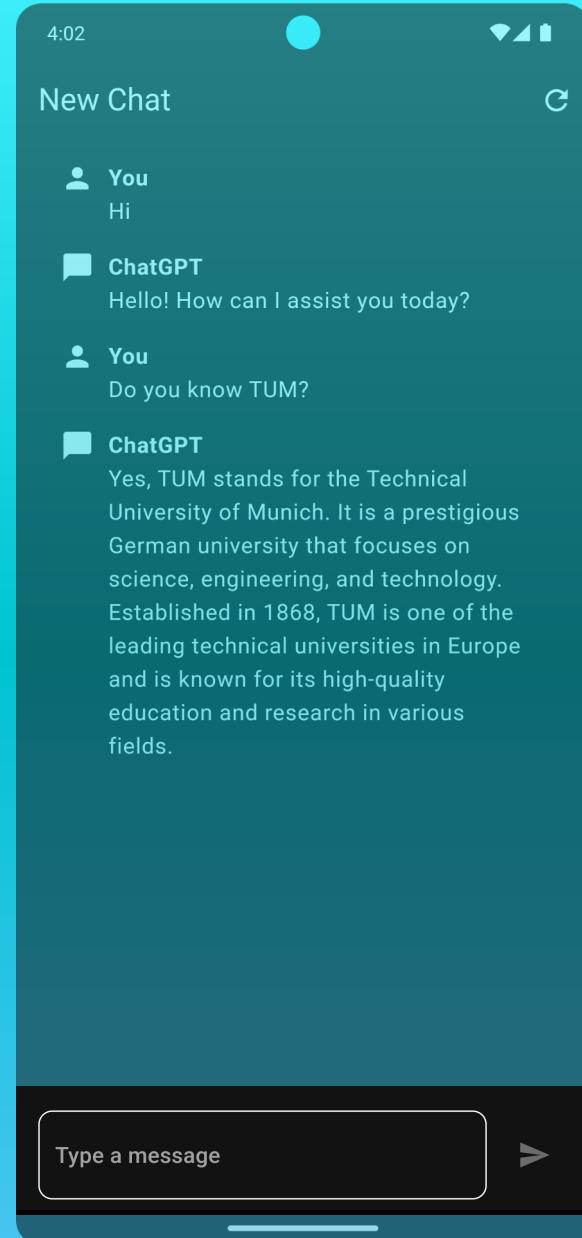


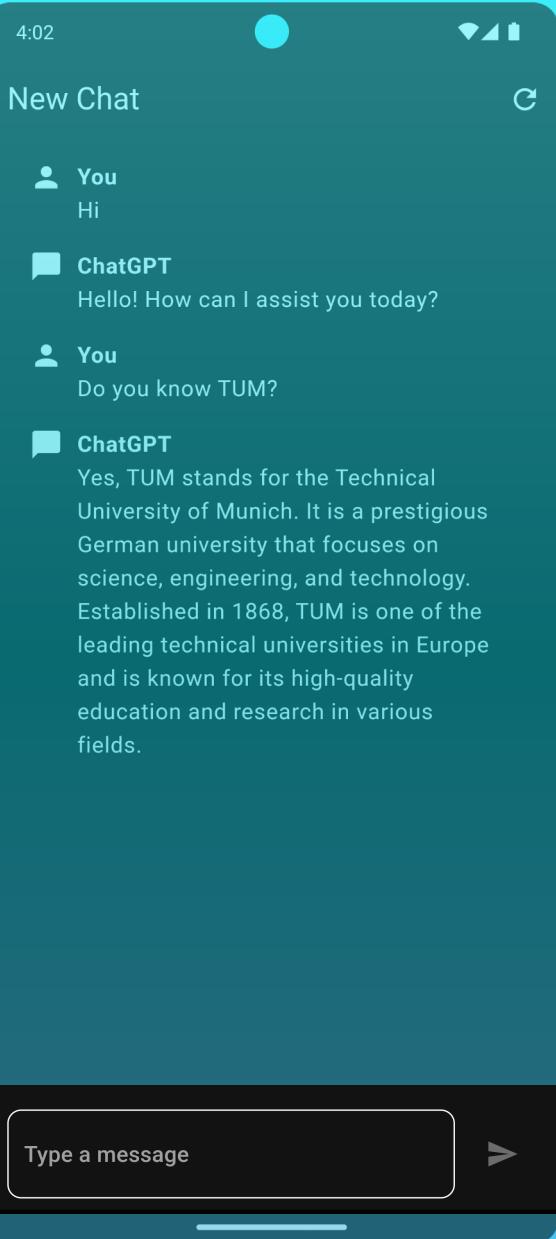


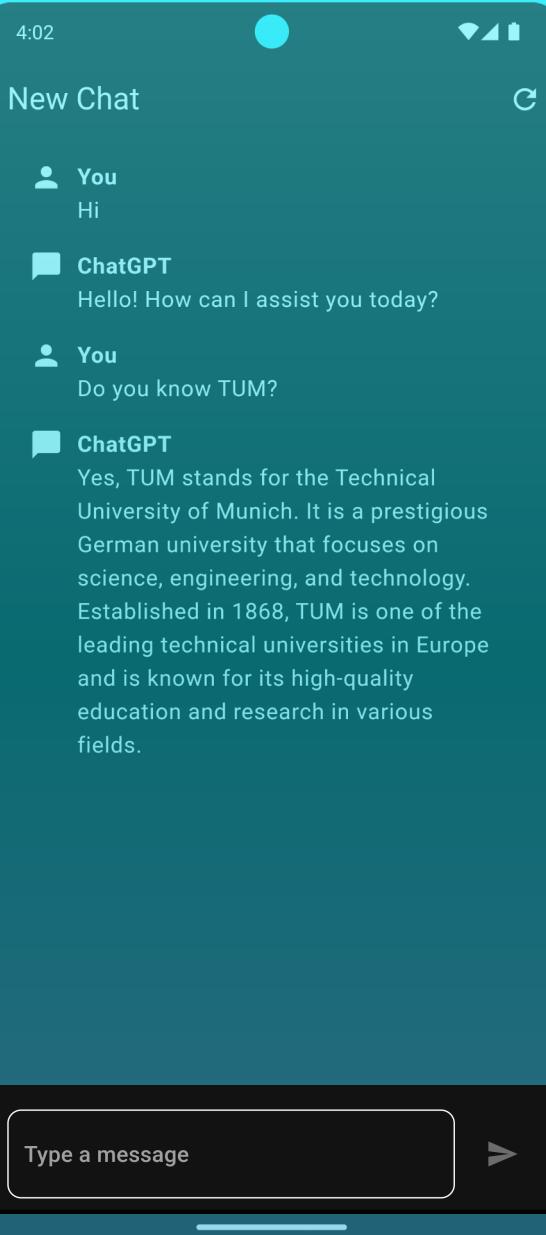


Flutter Inspector









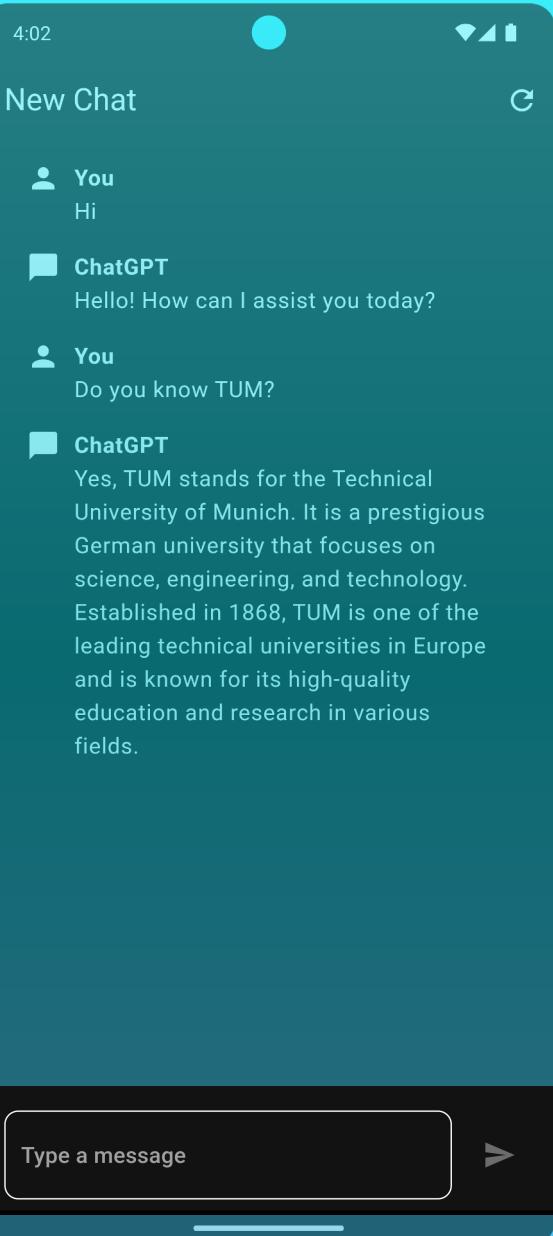
```
class ChatInput extends StatefulWidget {
    const ChatInput({super.key});

    @override
    State<ChatInput> createState() => _ChatInputState();
}

class _ChatInputState extends State<ChatInput> {
    final TextEditingController _inputController =
    TextEditingController();

    @override
    void dispose() {
        super.dispose();
        _inputController.dispose();
    }

    @override
    Widget build(BuildContext context) {
        return Row(
            children: [],
        );
    }
}
```



```
class ChatInput extends StatefulWidget {
  const ChatInput({super.key});

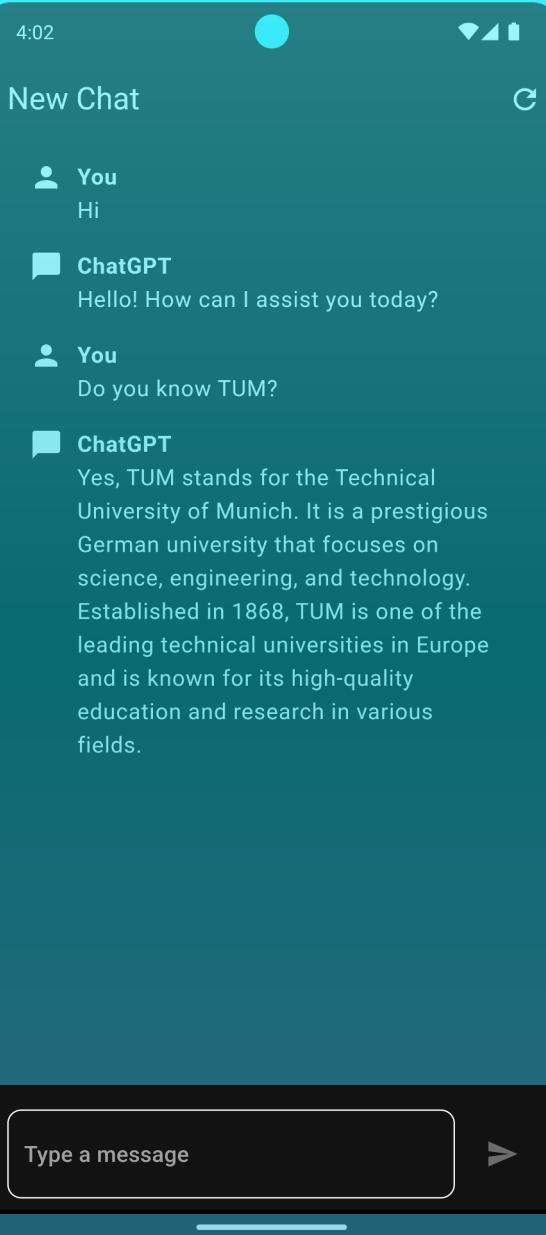
  @override
  State<ChatInput> createState() => _ChatInputState();
}

class _ChatInputState extends State<ChatInput> {
  final TextEditingController _inputController =
  TextEditingController();

  @override
  void dispose() {
    super.dispose();
    _inputController.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Row(
      children: [],
    );
  }
}
```

You can create widget boilerplate code using live templates in your editor



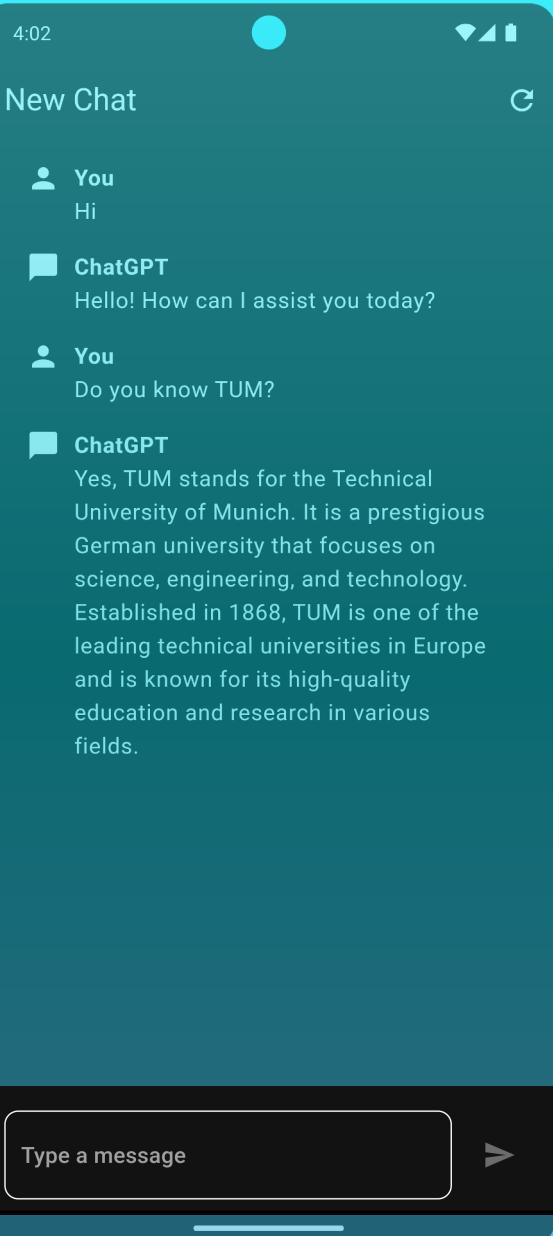
```
class ChatInput extends StatefulWidget {
  const ChatInput({super.key});
  @override
  State<ChatInput> createState() => _ChatInputState();
}

class _ChatInputState extends State<ChatInput> {
  final TextEditingController _inputController = TextEditingController();

  @override
  void dispose() {
    super.dispose();
    _inputController.dispose();
  }

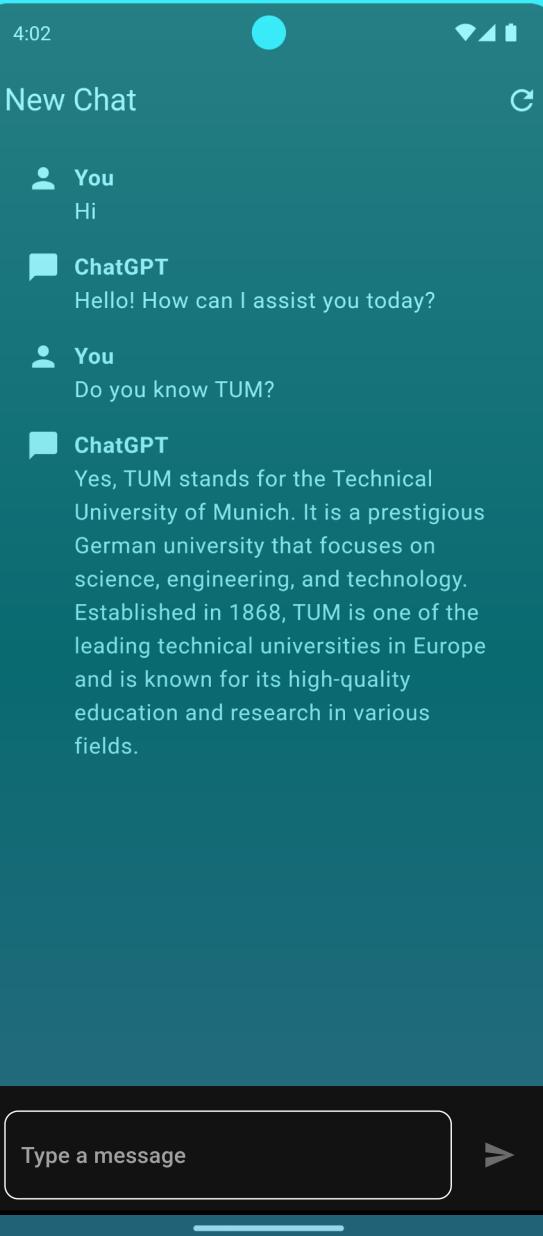
  @override
  Widget build(BuildContext context) {
    return Row(
      children: [],
    );
  }
}
```

Widget has mutable state



```
class ChatInput extends StatefulWidget {  
    const ChatInput({super.key});  Widget has mutable state  
  
    @override  
    State<ChatInput> createState() => _ChatInputState();  
}  
  
class _ChatInputState extends State<ChatInput> {  
    final TextEditingController _inputController =  
    TextEditingController();  
  
    @override  
    void dispose() {  
        super.dispose();  
        _inputController.dispose();  
    }  
  
    @override  
    Widget build(BuildContext context) {  
        return Row(  
            children: [],  
        );  
    }  
}
```

State of the ChatInput Widget



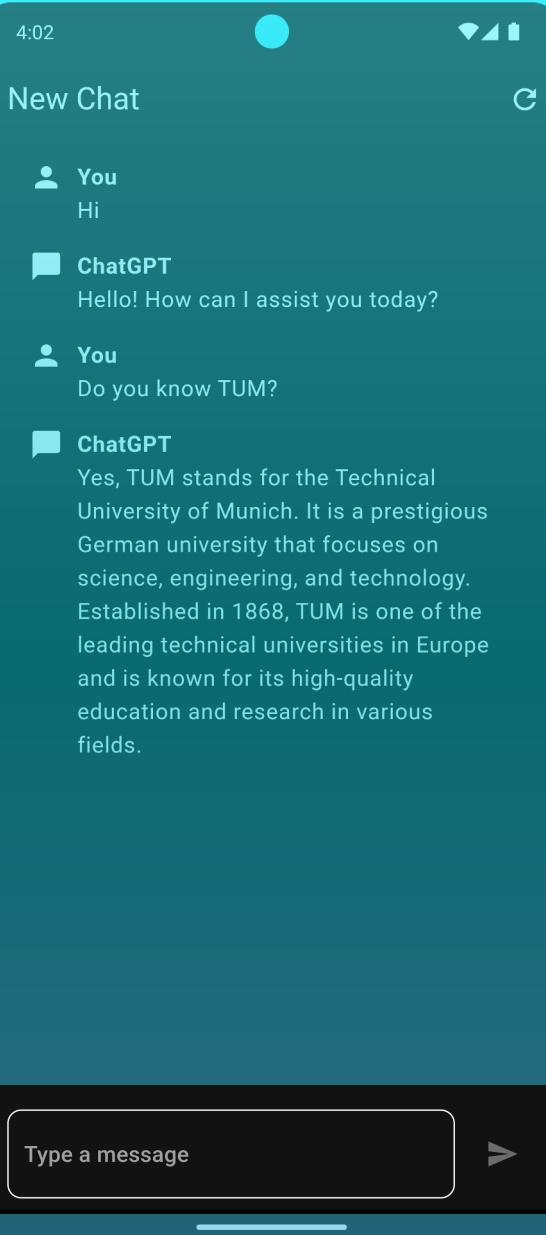
```
class ChatInput extends StatefulWidget {
    const ChatInput({super.key});

    @override
    State<ChatInput> createState() => _ChatInputState();
}

class _ChatInputState extends State<ChatInput> {
    final TextEditingController _inputController =
    TextEditingController();

    @override
    void dispose() {
        super.dispose();
        _inputController.dispose();
    }

    @override
    Widget build(BuildContext context) {
        return Row(
            children: [],
        );
    }
}
```



```
class ChatInput extends StatefulWidget {
  const ChatInput({super.key});

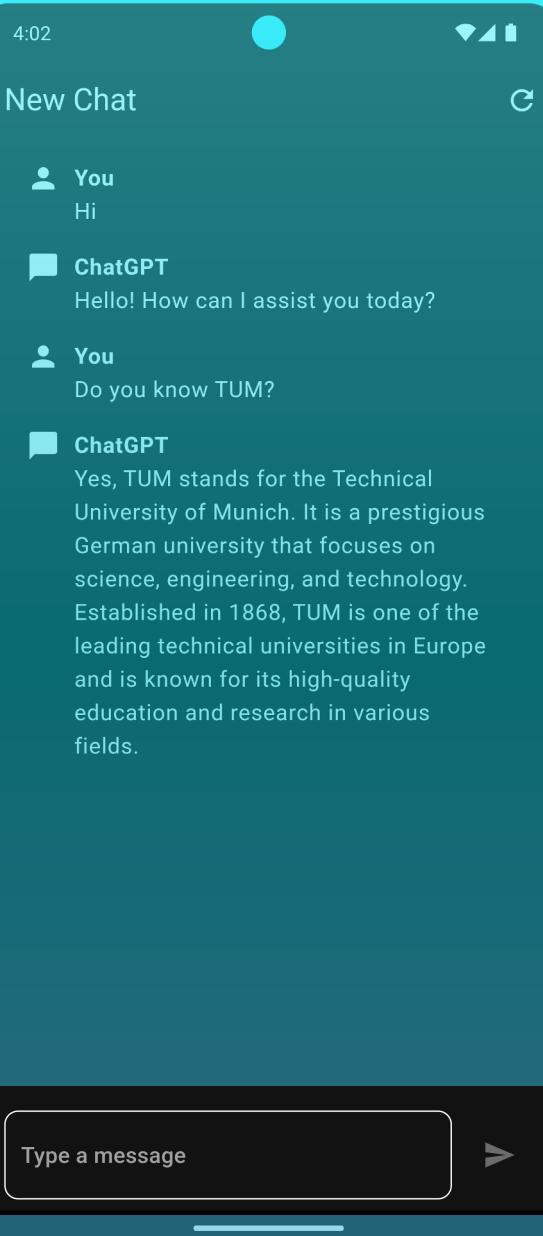
  @override
  State<ChatInput> createState() => _ChatInputState();
}

class _ChatInputState extends State<ChatInput> {
  final TextEditingController _inputController =
  TextEditingController();

  @override
  void dispose() {
    super.dispose();
    _inputController.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Row(
      children: [],
    );
  }
}
```

👉 Controller for the text field



```
class ChatInput extends StatefulWidget {
    const ChatInput({super.key});

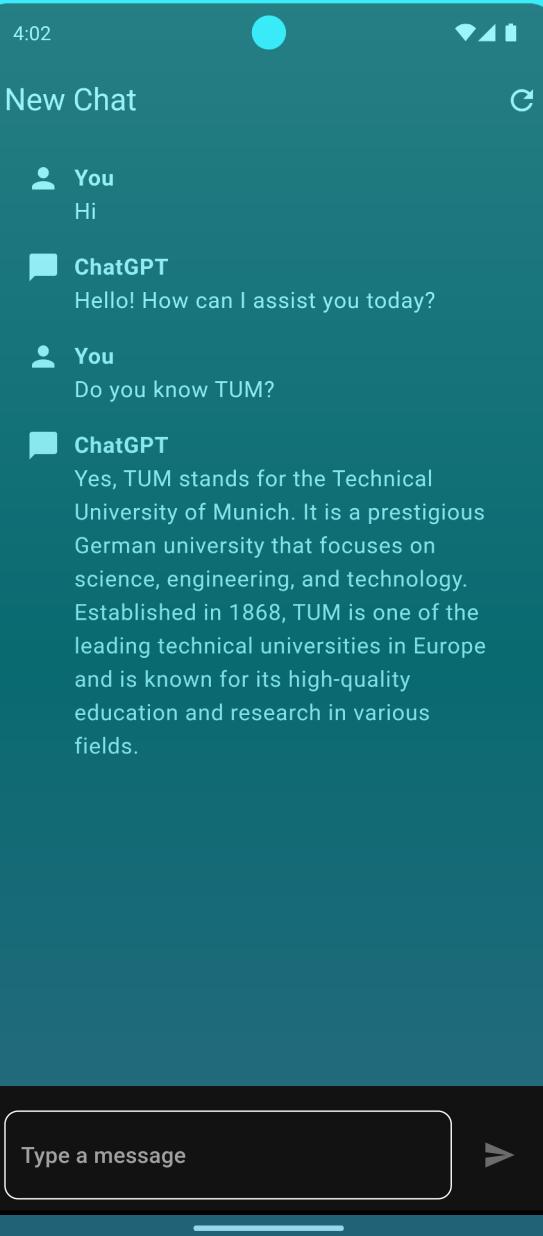
    @override
    State<ChatInput> createState() => _ChatInputState();
}

class _ChatInputState extends State<ChatInput> {
    final TextEditingController _inputController =
    TextEditingController();

    @override
    void dispose() { ➡️ Called on end of widget lifecycle
        super.dispose();
        _inputController.dispose();
    }

    @override
    Widget build(BuildContext context) {
        return Row(
            children: [],
        );
    }
}
```

👉 Controller for the text field



```
class ChatInput extends StatefulWidget {
    const ChatInput({super.key});

    @override
    State<ChatInput> createState() => _ChatInputState();
}

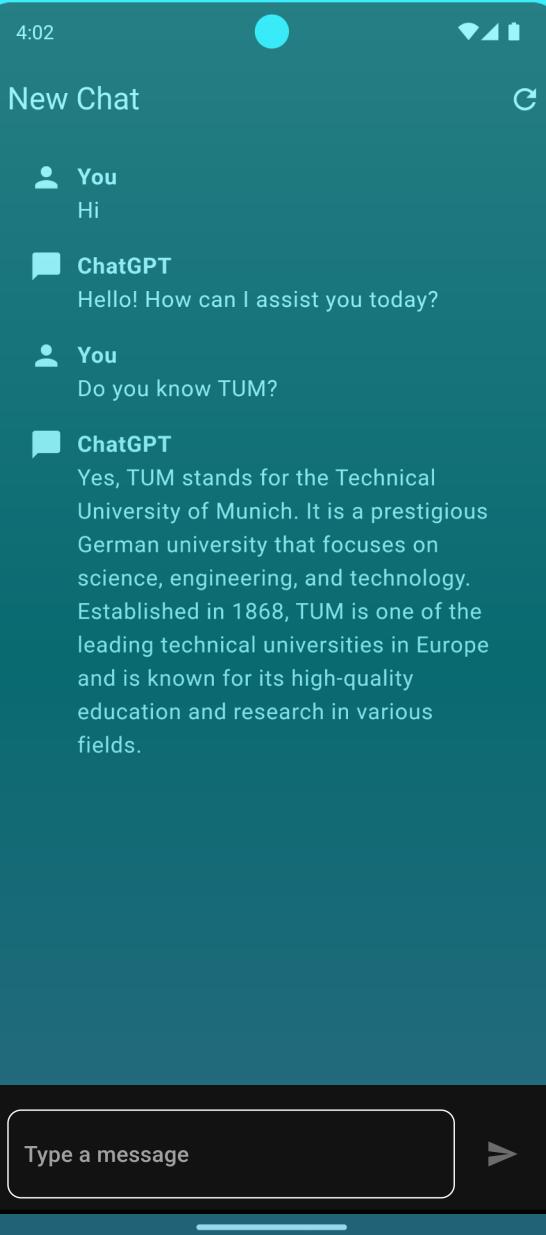
class _ChatInputState extends State<ChatInput> {
    final TextEditingController _inputController =
    TextEditingController();

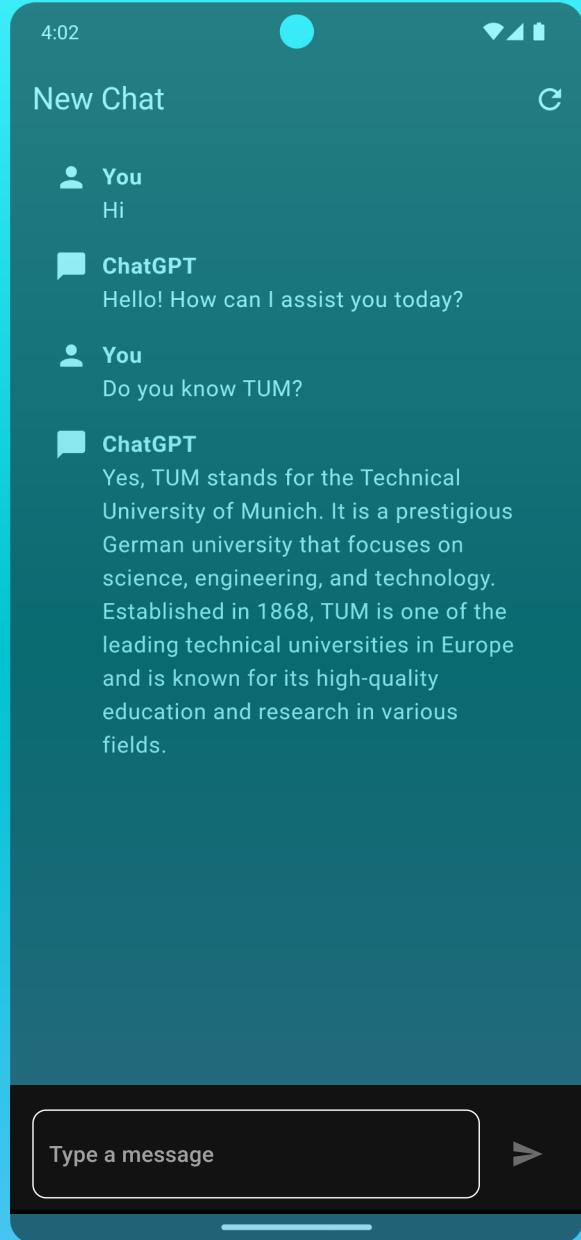
    @override
    void dispose() {    ➡️ Called on end of widget lifecycle
        super.dispose();
        _inputController.dispose();
    }

    @override
    Widget build(BuildContext context) {
        return Row(
            children: [],
        );
    }
}
```

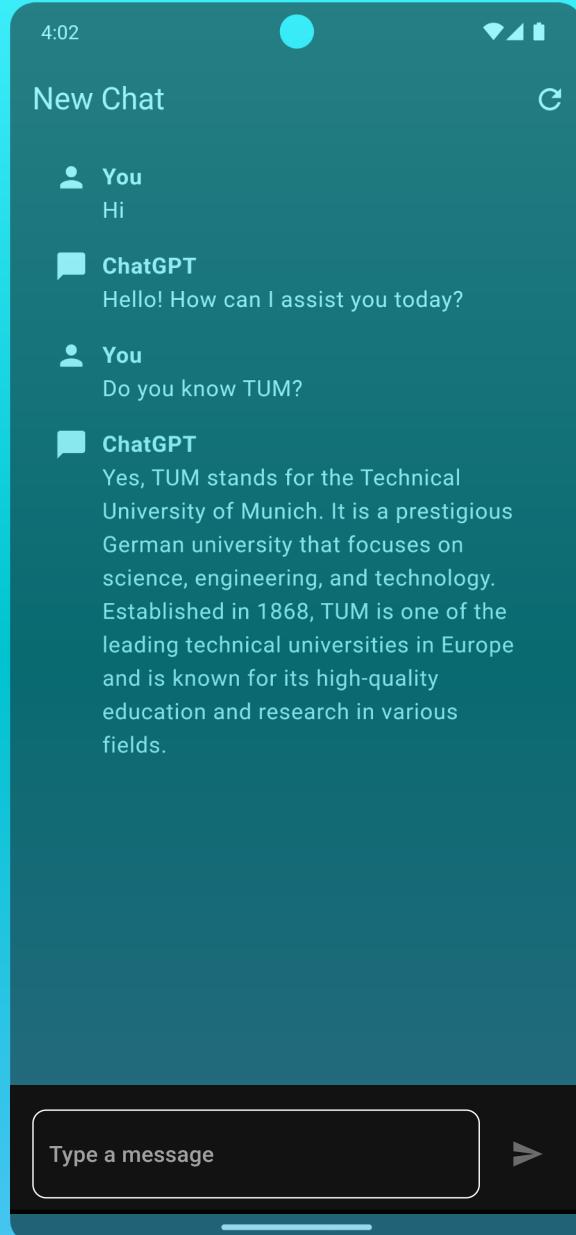
👉 Controller for the text field

👉 Defines the content of the widget

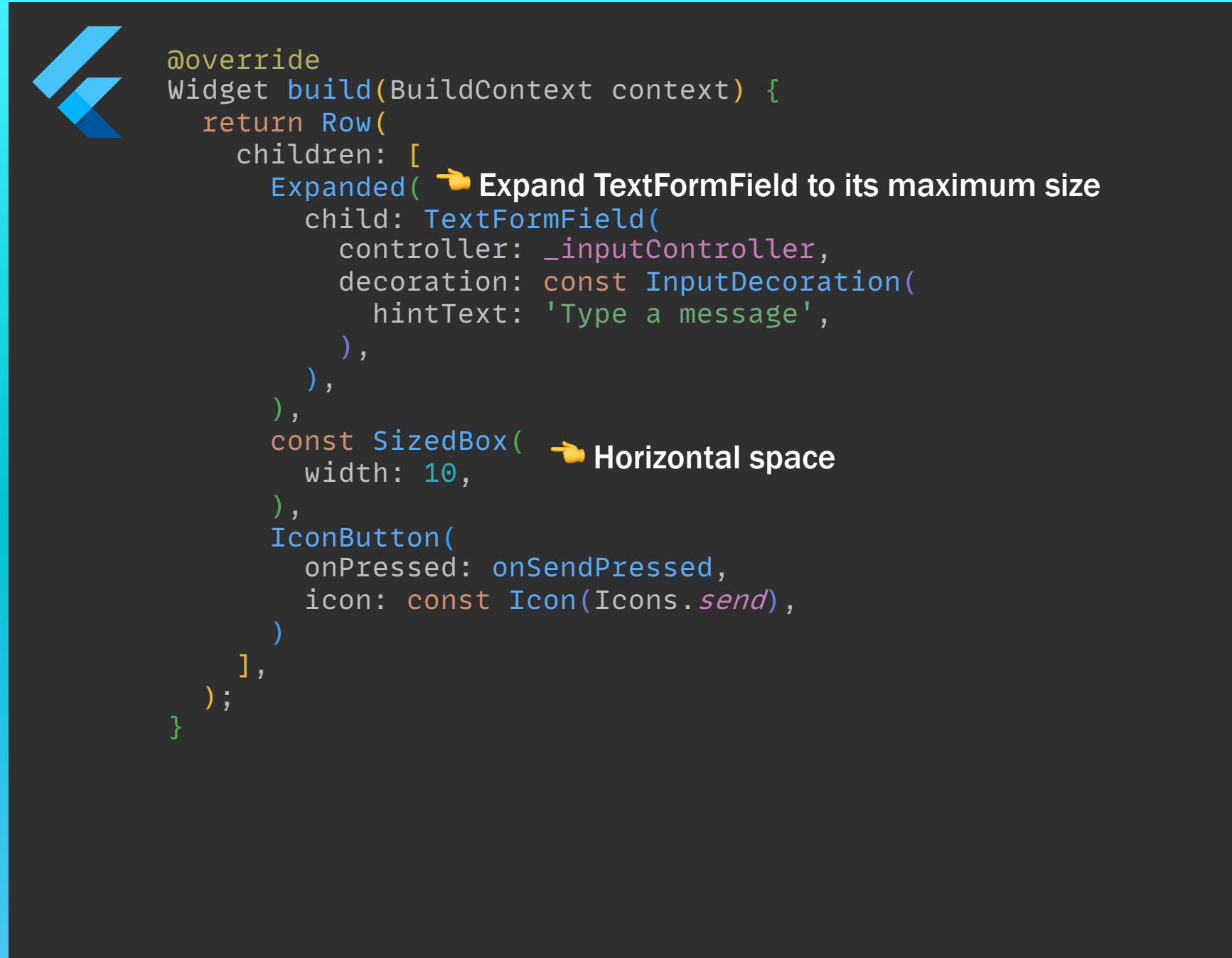
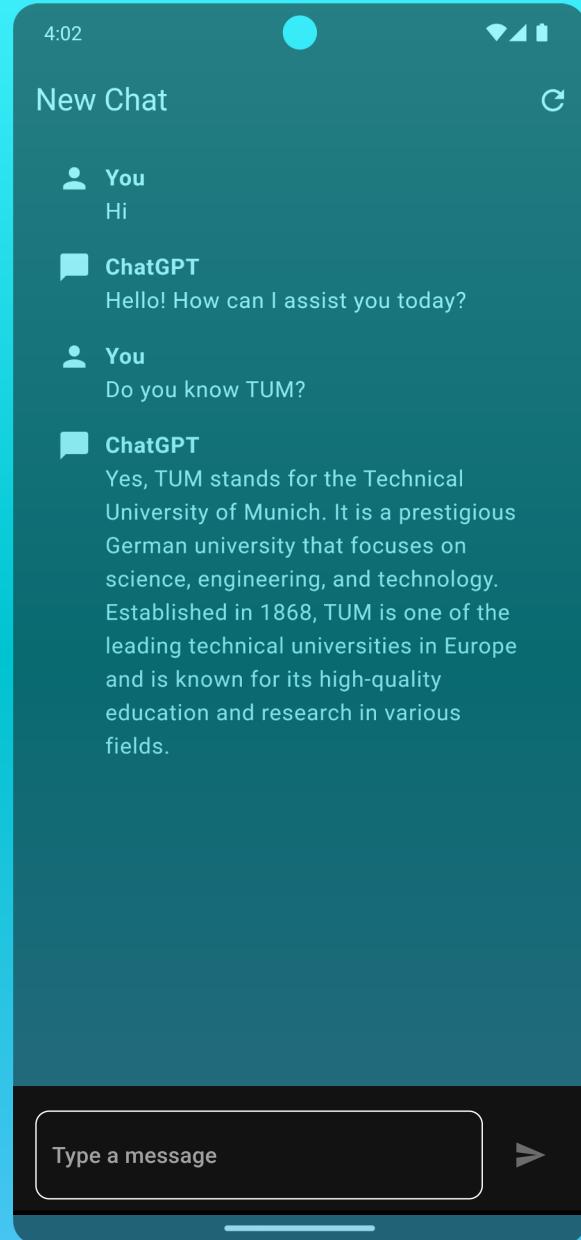


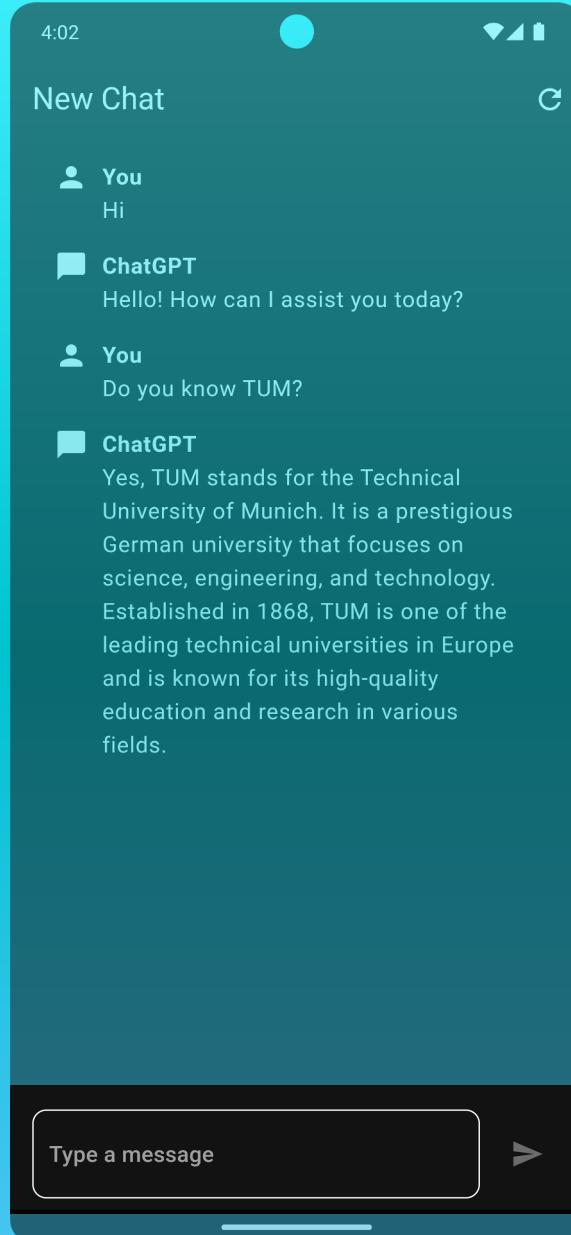


```
  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        Expanded(
          child: TextFormField(
            controller: _inputController,
            decoration: const InputDecoration(
              hintText: 'Type a message',
            ),
          ),
        ),
        const SizedBox(
          width: 10,
        ),
        IconButton(
          onPressed: onSendPressed,
          icon: const Icon(Icons.send),
        ),
      ],
    );
  }
}
```

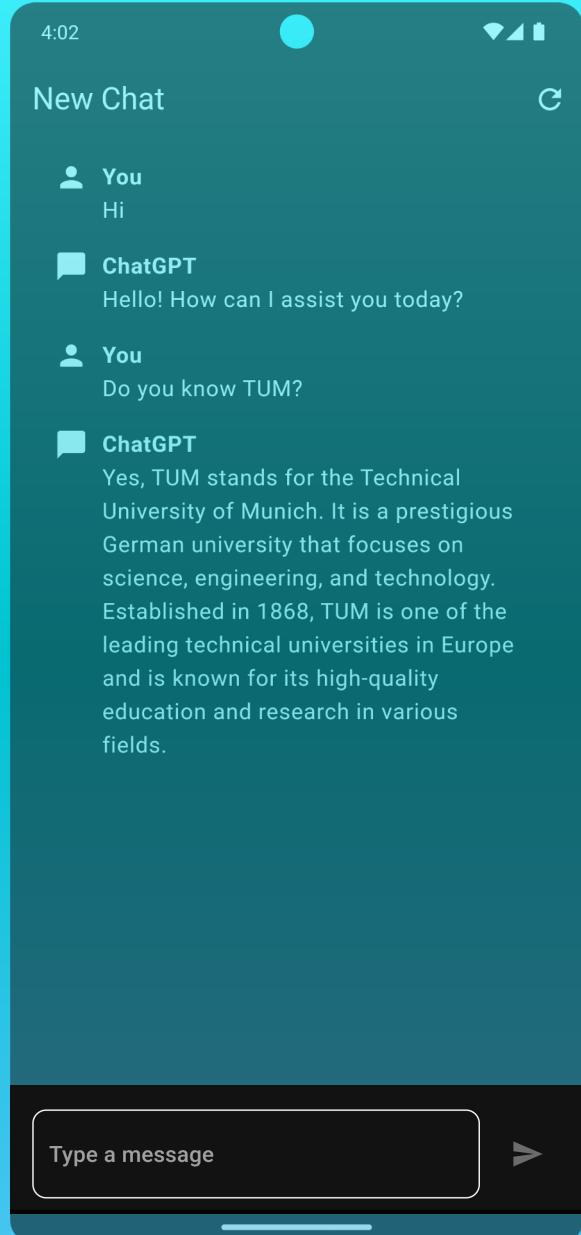


```
@override  
Widget build(BuildContext context) {  
  return Row(  
    children: [  
      Expanded( ➡ Expand TextFormField to its maximum size  
        child: TextFormField(  
          controller: _inputController,  
          decoration: const InputDecoration(  
            hintText: 'Type a message',  
          ),  
        ),  
      ),  
      const SizedBox(  
        width: 10,  
      ),  
      IconButton(  
        onPressed: onSendPressed,  
        icon: const Icon(Icons.send),  
      ),  
    ],  
  );  
}
```

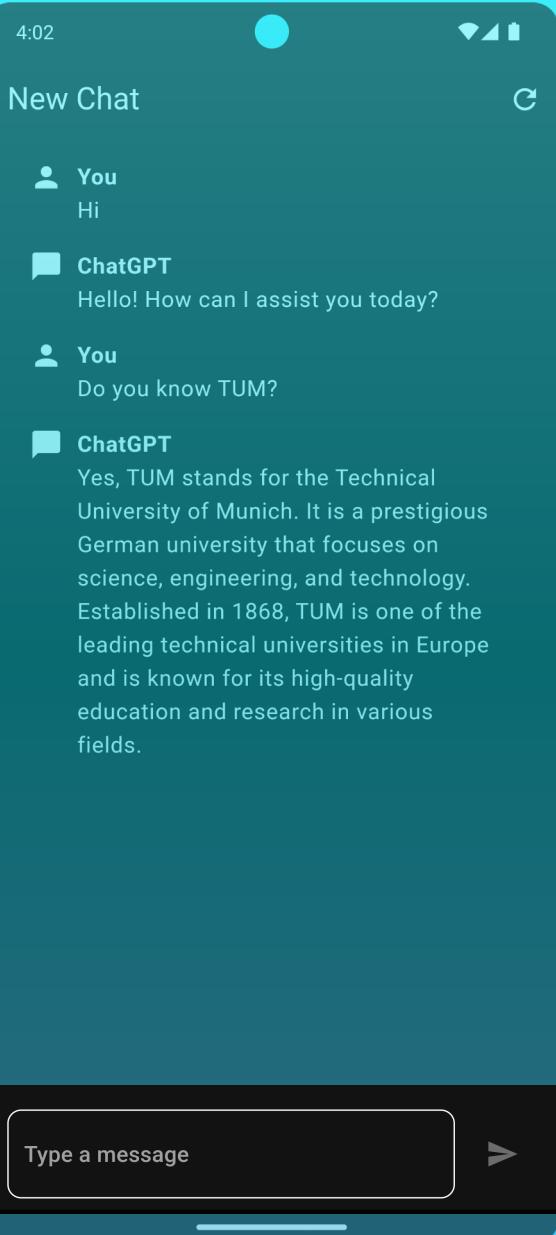




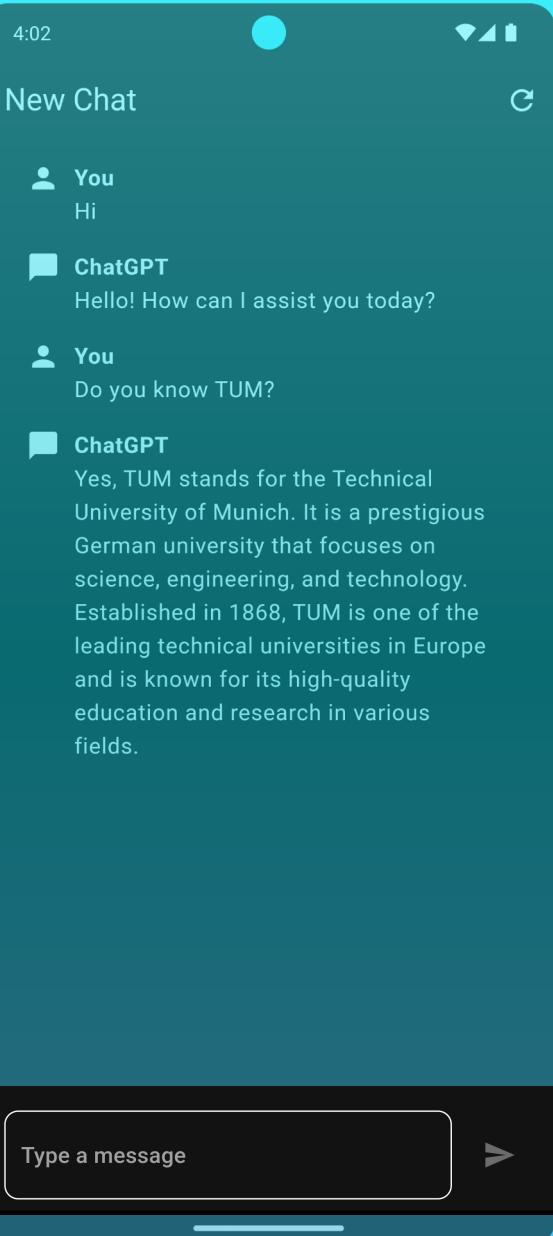
```
override  
Widget build(BuildContext context) {  
  return Row(  
    children: [  
      Expanded( ➔ Expand TextFormField to its maximum size  
        child: TextFormField(  
          controller: _inputController,  
          decoration: const InputDecoration(  
            hintText: 'Type a message',  
          ),  
        ),  
      ),  
      const SizedBox( ➔ Horizontal space  
        width: 10,  
      ),  
      IconButton(  
        onPressed: onSendPressed, ➔ Called when button is pressed  
        icon: const Icon(Icons.send),  
      ),  
    ],  
  );  
}
```



```
  @override  
  Widget build(BuildContext context) {  
    return Row(  
      children: [  
        Expanded(  
          child: TextFormField(  
            controller: _inputController,  
            decoration: const InputDecoration(  
              hintText: 'Type a message',  
            ),  
          ),  
        ),  
        const SizedBox(  
          width: 10,  
        ),  
        if (_isSending) ➡ Show progress indicator while sending a message  
        const CircularProgressIndicator()  
      else  
        IconButton(  
          onPressed: onSendPressed,  
          icon: const Icon(Icons.send),  
        )  
      ],  
    );  
  }
```

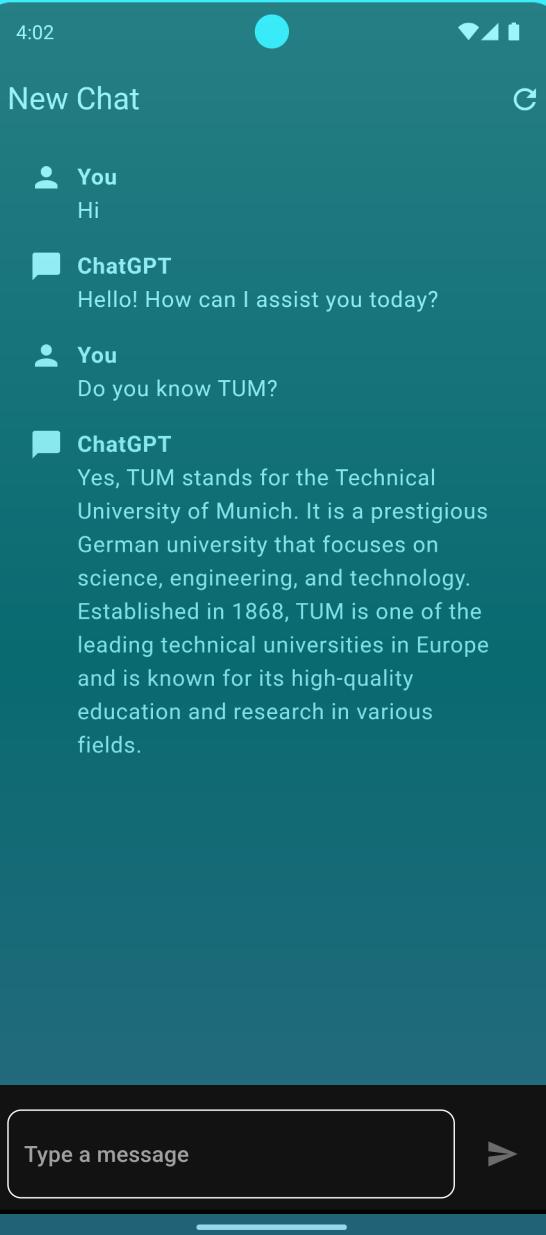


```
bool _isSending = false;  
  
void onSendPressed() async {  
  try {  
    setState(() {  
      _isSending = true;  
    });  
    ...  
  } finally {  
    setState(() {  
      _isSending = false;  
    });  
  }  
}
```



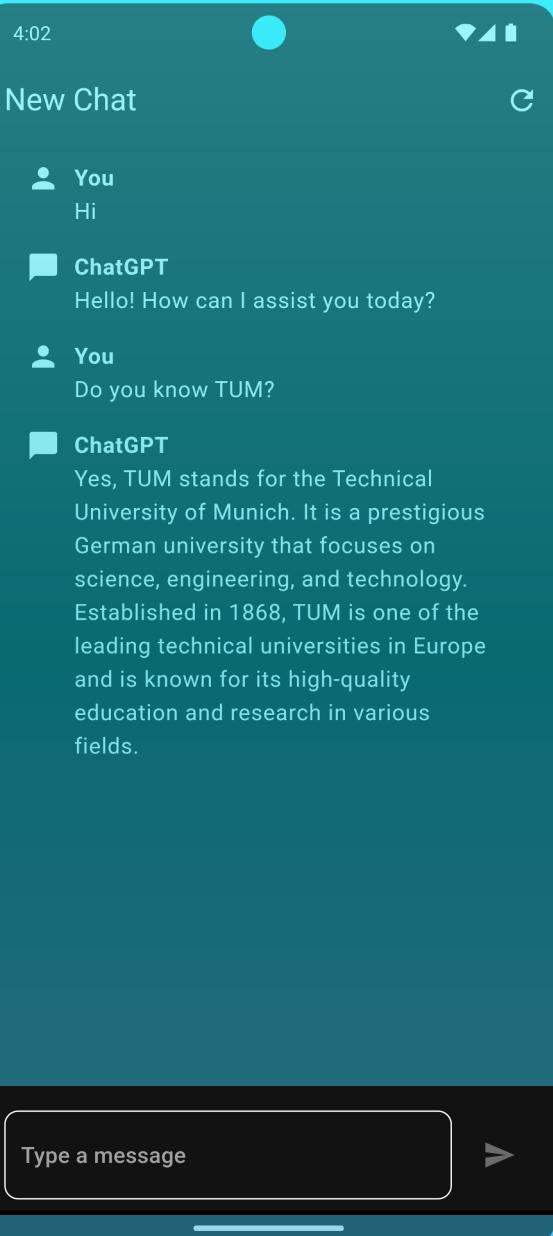
```
bool _isSending = false;

void onSendPressed() async {
  try {
    setState(() { ➡ Rebuild children with updated state
      _isSending = true;
    });
    ...
  } finally {
    setState(() {
      _isSending = false;
    });
  }
}
```



```
bool _isSending = false;

void onSendPressed() async {
  try {
    setState(() { ➡ Rebuild children with updated state
      _isSending = true;
    });
    ...
    ➡ Send message logic
  } finally {
    setState(() {
      _isSending = false;
    });
  }
}
```



```
bool _isSending = false;  
  
void onSendPressed() async {  
  try {  
    setState(() { ➡ Rebuild children with updated state  
    _isSending = true;  
  });  
  ... ➡ Send message logic  
} finally {  
  setState(() {  
    _isSending = false;  
  });  
} ➡ Set sending flag to false  
}
```



```
class ChatInput extends StatefulWidget {
  const ChatInput({super.key});

  @override
  State<ChatInput> createState() => _ChatInputState();
}

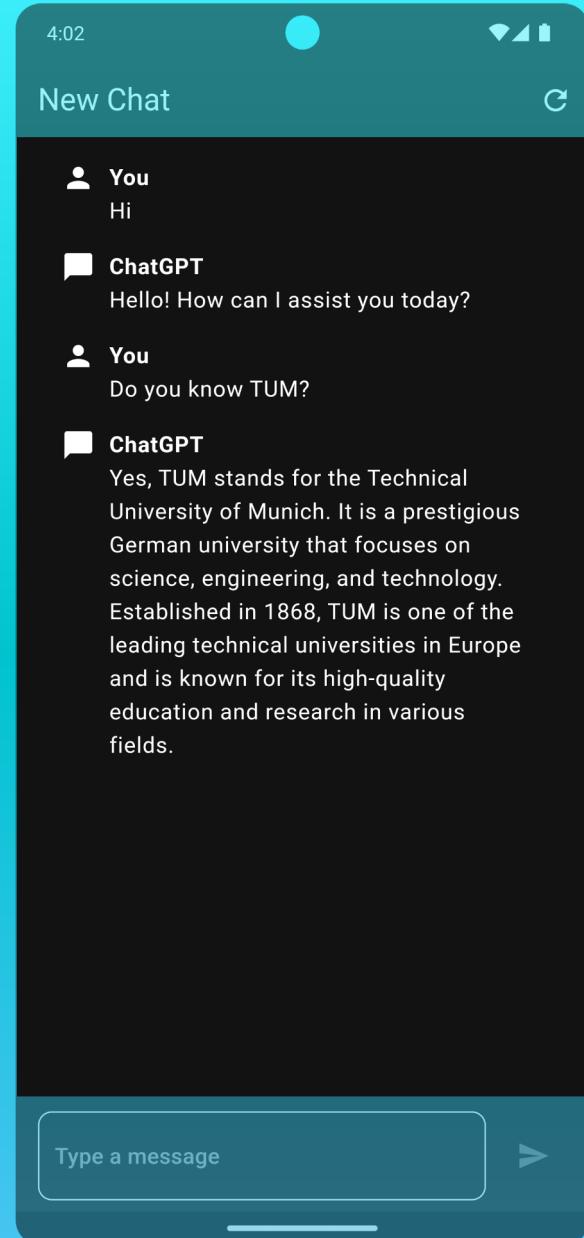
class _ChatInputState extends State<ChatInput> {
  final TextEditingController _inputController =
  TextEditingController();

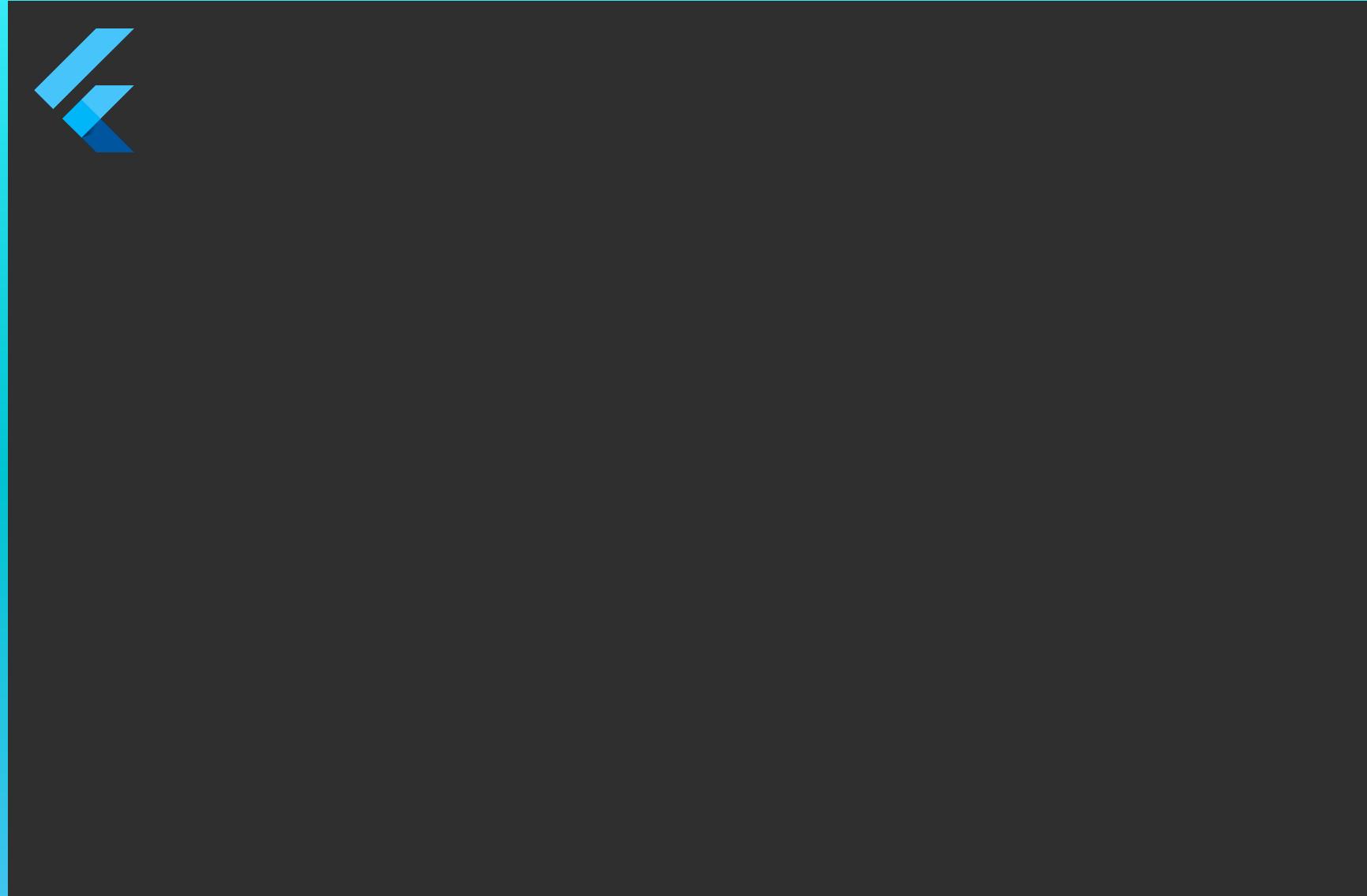
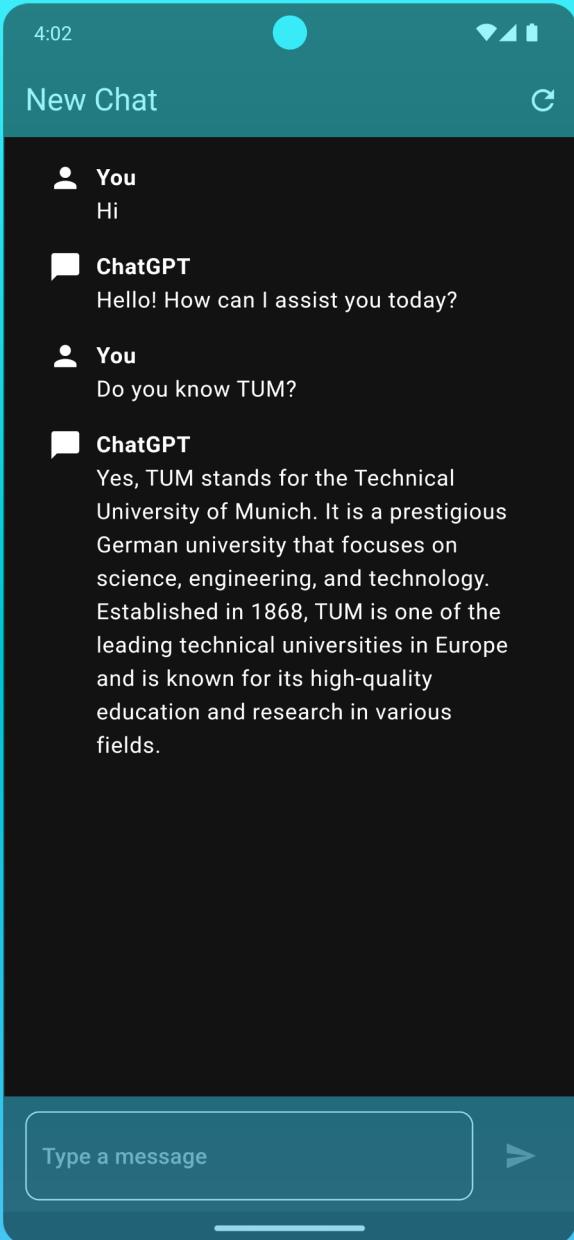
  @override
  void dispose() {
    super.dispose();
    _inputController.dispose();
  }

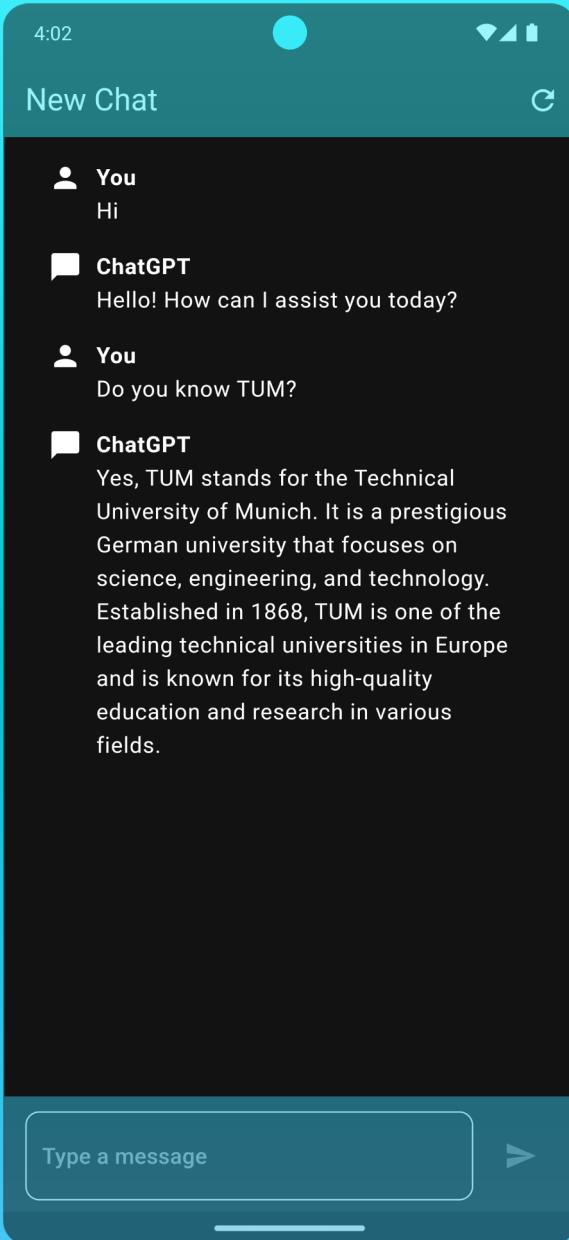
  bool _isSending = false;

  void onSendPressed() async {
    try {
      setState(() {
        _isSending = true;
      });
      final chatProvider = context.read<ChatProvider>();
      await chatProvider.sendMessage(
        _inputController.text.trim(),
      );
      _inputController.clear();
      await chatProvider.requestResponse();
    } finally {
      setState(() {
        _isSending = false;
      });
    }
  }
}

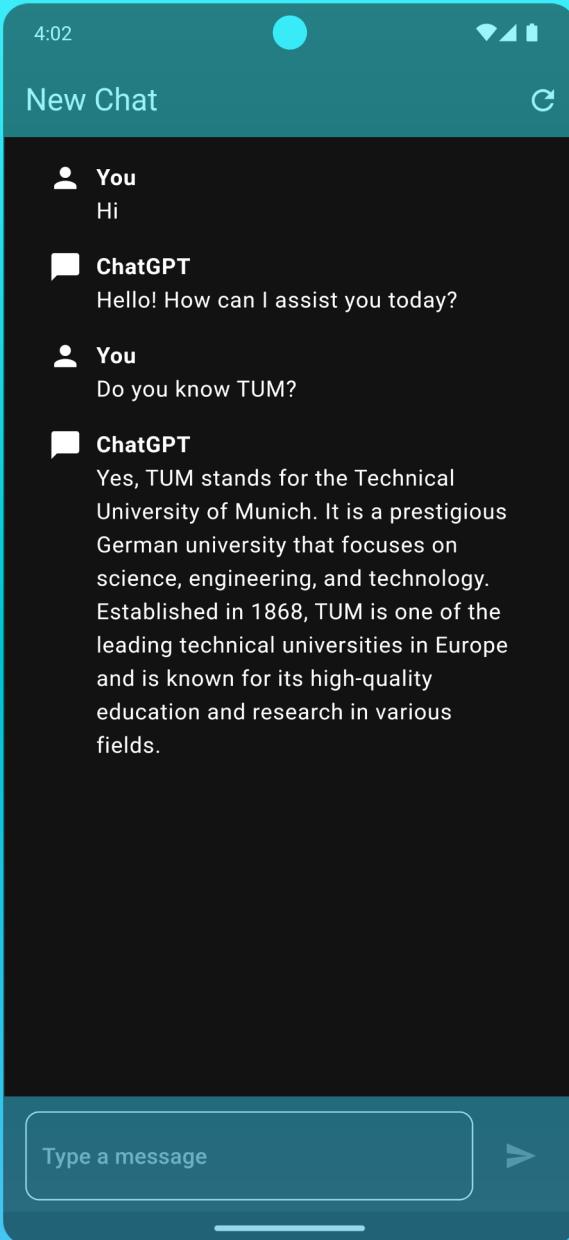
@override
Widget build(BuildContext context) {
  return Row(
    children: [
      Expanded(
        child: TextFormField(
          controller: _inputController,
          decoration: const InputDecoration(
            hintText: 'Type a message',
          ),
        ),
      ),
      const SizedBox(
        width: 10,
      ),
      if (_isSending)
        const CircularProgressIndicator()
      else
        ListenableBuilder(
          listenable: _inputController,
          builder: (context, _) {
            return IconButton(
              onPressed:
              _inputController.text.isEmpty ? null : onSendPressed,
              icon: const Icon(Icons.send),
            );
          },
        ),
    ],
  );
}
```



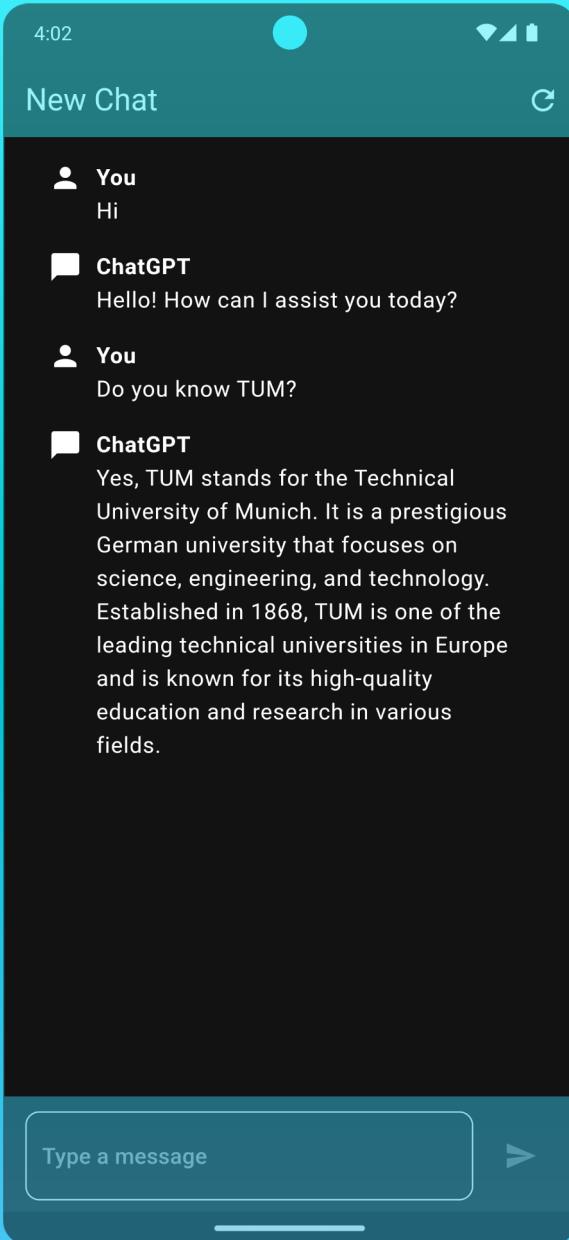




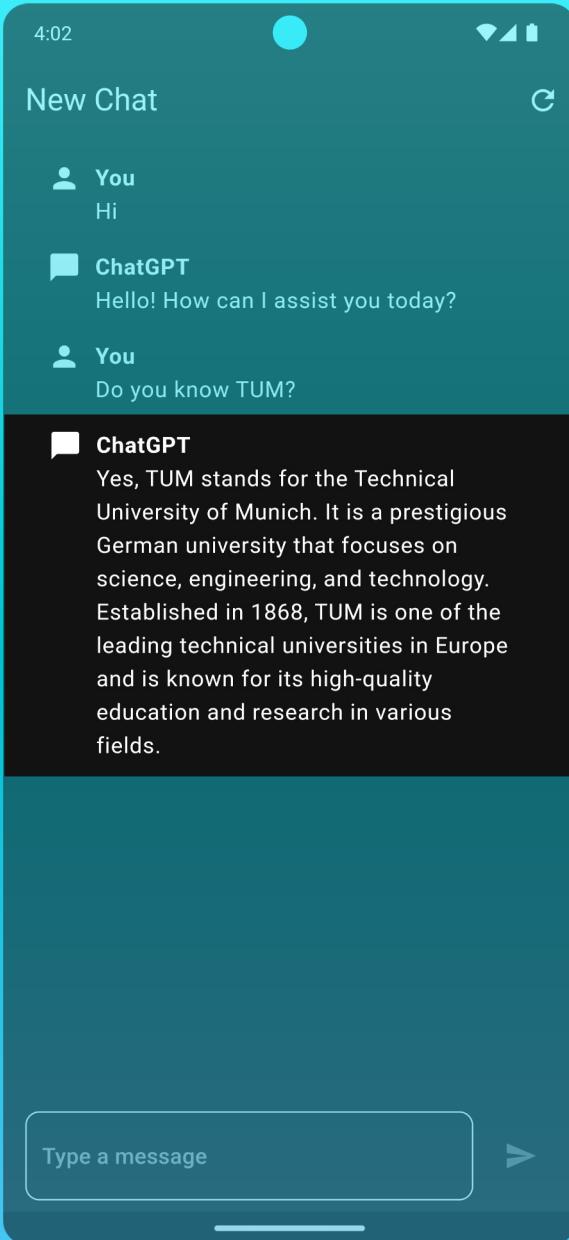
```
class ChatDisplay extends StatelessWidget {  
  
  const ChatDisplay({super.key, required this.chat});  
  
  final Chat chat;  
  
  @override  
  Widget build(BuildContext context) {  
    return ListView.builder(  
      itemBuilder: (context, index) {  
        final message = chat.messages[index];  
        return ChatMessageTile(message: message);  
      },  
      itemCount: chat.messages.length,  
    );  
  }  
}
```



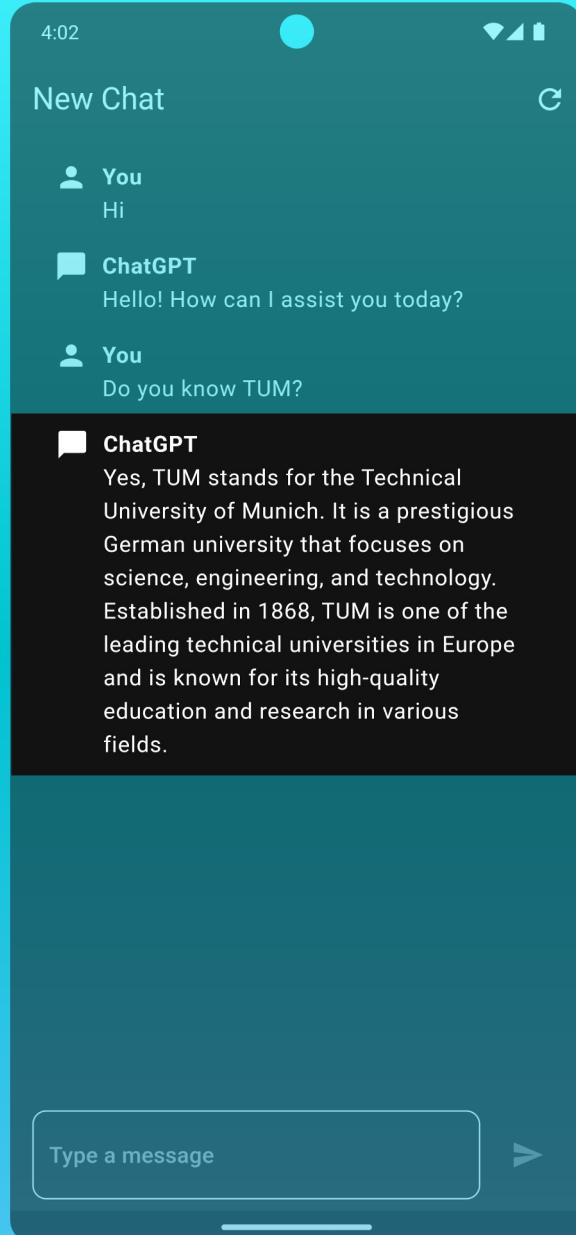
```
class ChatDisplay extends StatelessWidget {  
  Widget has no mutable state  
  const ChatDisplay({super.key, required this.chat});  
  
  final Chat chat;  
  
  @override  
  Widget build(BuildContext context) {  
    return ListView.builder(  
      itemBuilder: (context, index) {  
        final message = chat.messages[index];  
        return ChatMessageTile(message: message);  
      },  
      itemCount: chat.messages.length,  
    );  
  }  
}
```



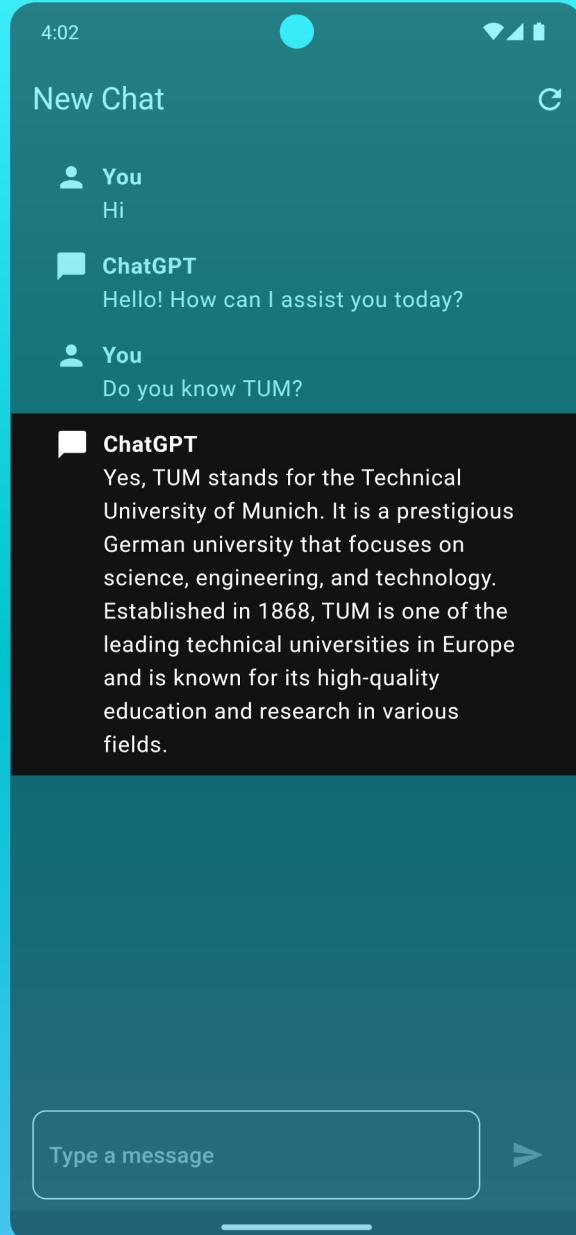
```
class ChatDisplay extends StatelessWidget {  
  Widget has no mutable state  
  const ChatDisplay({super.key, required this.chat});  
  
  final Chat chat;  
  
  @override  
  Widget build(BuildContext context) {  
    return ListView.builder(  
      // Build Widget for each message  
      itemBuilder: (context, index) {  
        final message = chat.messages[index];  
        return ChatMessageTile(message: message);  
      },  
      itemCount: chat.messages.length,  
    );  
  }  
}
```



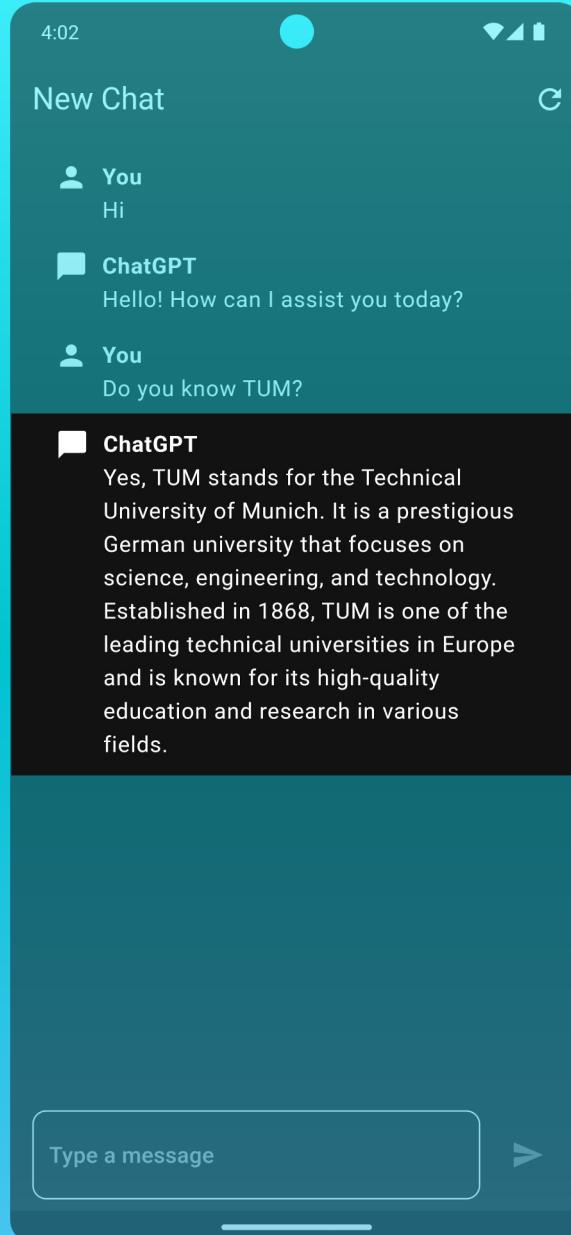
```
class ChatMessageTile extends StatelessWidget {  
  const ChatMessageTile({  
    super.key,  
    required this.message,  
  });  
  
  final ChatMessage message;  
  
  @override  
  Widget build(BuildContext context) {  
    return ListTile(  
      );  
  }  
}
```



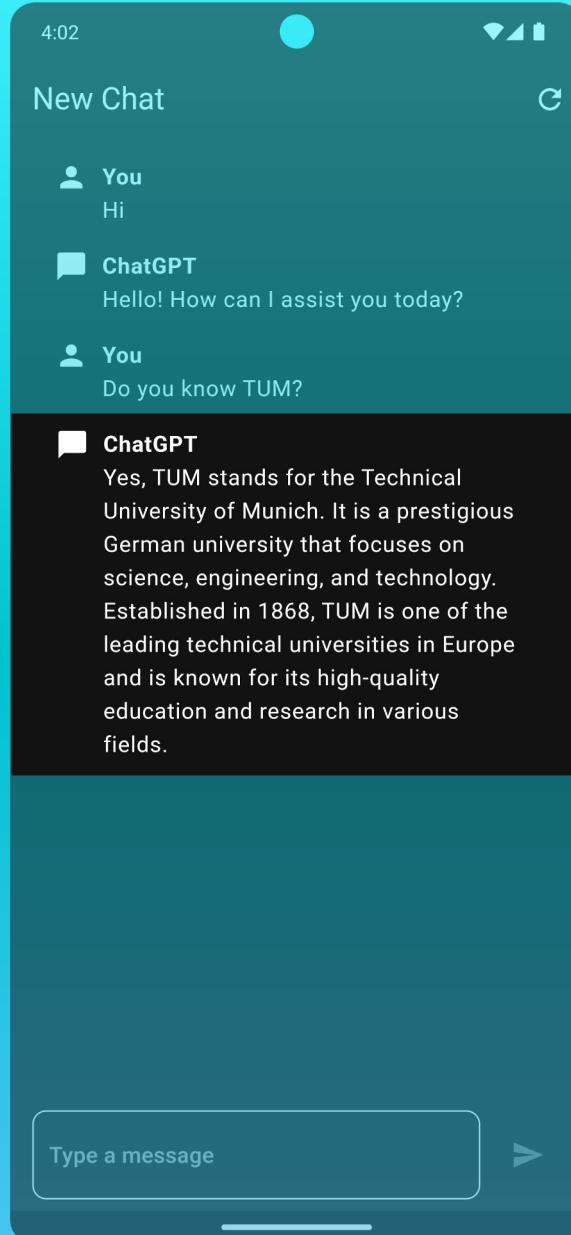
```
  @override
  Widget build(BuildContext context) {
    return ListTile(
      title: Row(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
          if (message.sender == Sender.user)
            const Icon(Icons.person)
          else
            const Icon(Icons.chat_bubble),
          const SizedBox(
            width: 10,
          ),
          Expanded(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.start,
              children: [
                Text(
                  message.sender.displayText,
                  style: const TextStyle(
                    fontWeight: FontWeight.bold,
                  ),
                ),
                Text(message.text),
              ],
            ),
          ),
        ],
      );
    }
}
```



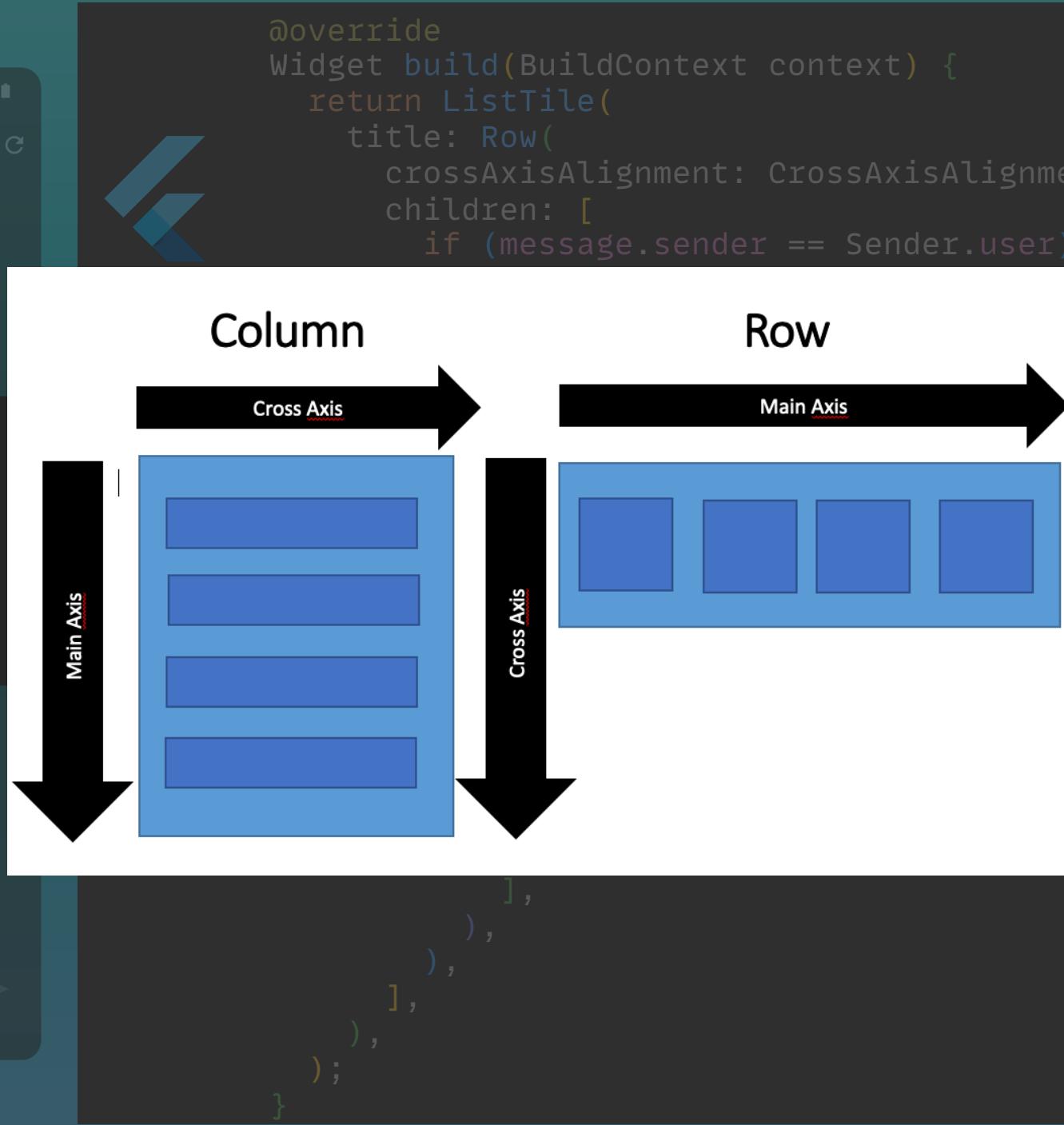
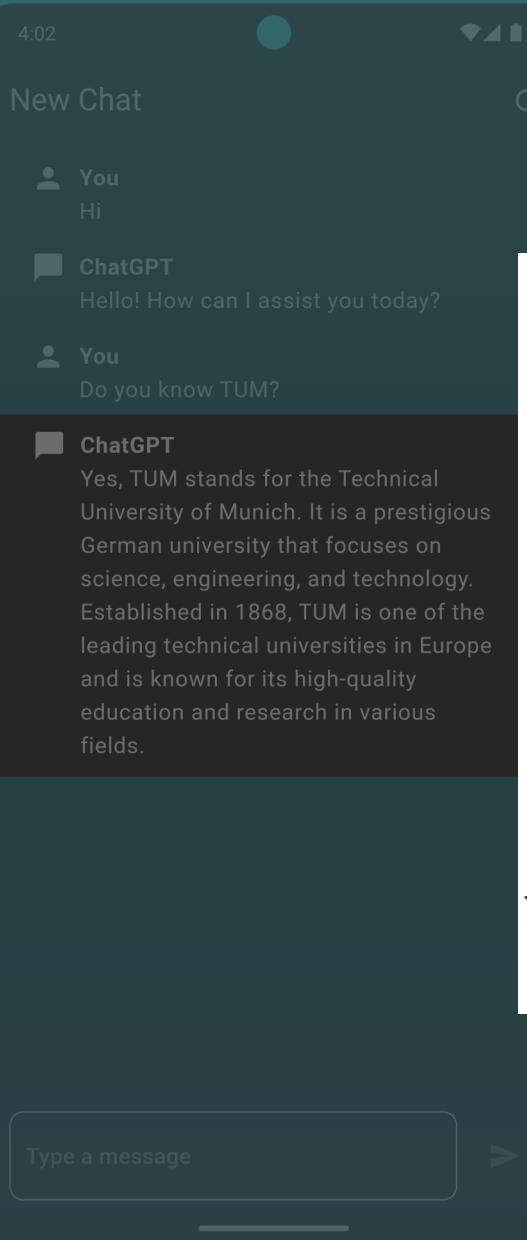
```
Widget build(BuildContext context) {
  return ListTile(
    title: Row(
      mainAxisAlignment: MainAxisAlignment.start,
      children: [
        if (message.sender == Sender.user)
          const Icon(Icons.person)
        else
          const Icon(Icons.chat_bubble),
        Choose icon
      ],
      depending on sender
    ),
    const SizedBox(
      width: 10,
    ),
    Expanded(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
          Text(
            message.sender.displayText,
            style: const TextStyle(
              fontWeight: FontWeight.bold,
            ),
            Choose icon
          ),
          Text(message.text),
        ],
      ),
    ),
  );
}
```



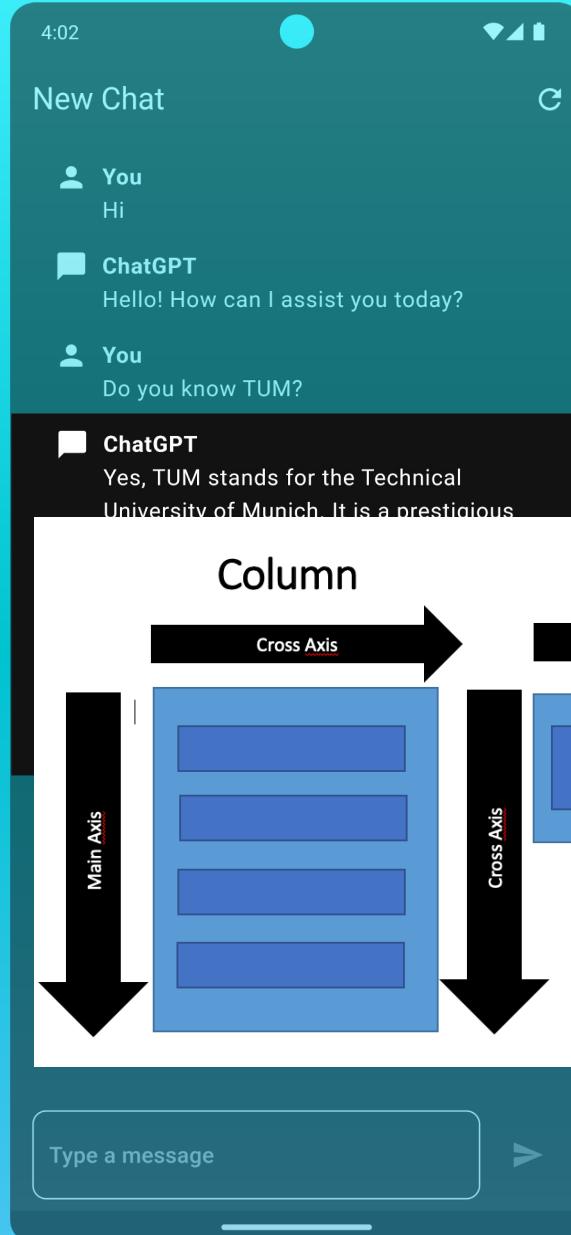
```
override  
Widget build(BuildContext context) {  
  return ListTile(  
    title: Row(  
      mainAxisAlignment: MainAxisAlignment.start,  
      children: [  
        if (message.sender == Sender.user)  
          const Icon(Icons.person)  Choose icon  
        else  
          const Icon(Icons.chat_bubble), depending on sender  
        const SizedBox(  
          width: 10,  Horizontal space  
        ),  
        Expanded(  
          child: Column(  
            mainAxisAlignment: MainAxisAlignment.start,  
            children: [  
              Text(  
                message.sender.displayText,  
                style: const TextStyle(  
                  fontWeight: FontWeight.bold,  
                ),  
                ),  
              Text(message.text),  
            ],  
          ),  
        ),  
      ],  
    );  
}
```



```
override  
Widget build(BuildContext context) {  
  return ListTile(  
    title: Row(  
      mainAxisAlignment: MainAxisAlignment.start,  
      children: [  
        if (message.sender == Sender.user)  
          const Icon(Icons.person)  Choose icon  
        else  
          const Icon(Icons.chat_bubble), depending on sender  
        const SizedBox(  
          width: 10,  Horizontal space  
        ),  
        Expanded(  
          child: Column(  
            mainAxisAlignment: MainAxisAlignment.start,  
            children: [  Align Texts to the left   
              Text(  
                message.sender.displayText,  
                style: const TextStyle(  
                  fontWeight: FontWeight.bold,  
                ),  
                ),  
              Text(message.text),  
            ],  
          ),  
        ),  
      ],  
    );  
}
```



alignment.start,
old,

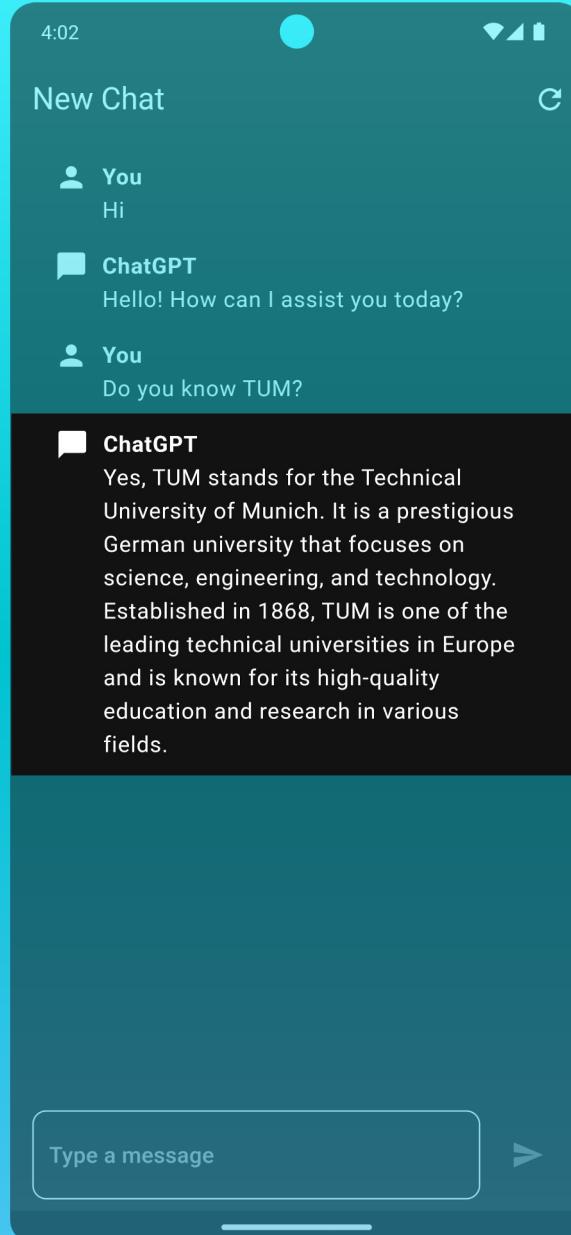


```
override  
Widget build(BuildContext context) {  
  return ListTile(  
    title: Row(  
      mainAxisAlignment: MainAxisAlignment.start,  
      children: [  
        if (message.sender == Sender.user)  
          const Icon(Icons.person)  
        else  
          const Icon(Icons.chat_bubble),  
        const SizedBox(  
          width: 10, ➡ Choose icon  
        ),  
        Expanded(  
          child: Column(  
            mainAxisAlignment: MainAxisAlignment.start,  
            children: [ Align Texts to the left ➤  
              Text(  
                message.sender.displayText,  
                style: const TextStyle(  
                  fontWeight: FontWeight.bold,  
                ),  
                ),  
              Text(message.text),  
            ],  
          ),  
        ),  
      ],  
    );  
}
```

➡ Choose icon

➡ Horizontal space

➡ Align Texts to the left ➤



```
override  
Widget build(BuildContext context) {  
  return ListTile(  
    title: Row(  
      mainAxisAlignment: MainAxisAlignment.start,  
      children: [  
        if (message.sender == Sender.user)  
          const Icon(Icons.person)  Choose icon  
        else  
          const Icon(Icons.chat_bubble), depending on sender  
        const SizedBox(  
          width: 10,  Horizontal space  
        ),  
        Expanded(  
          child: Column(  
            mainAxisAlignment: MainAxisAlignment.start,  
            children: [  Align Texts to the left   
              Text(  
                message.sender.displayName,  
                style: const TextStyle(  
                  fontWeight: FontWeight.bold,  
                ),  Make sender name bold   
              Text(message.text),  
            ],  
          ),  
        );  
      );  
    };
```



```
class ChatDisplay extends StatelessWidget {
  const ChatDisplay({super.key, required this.chat});

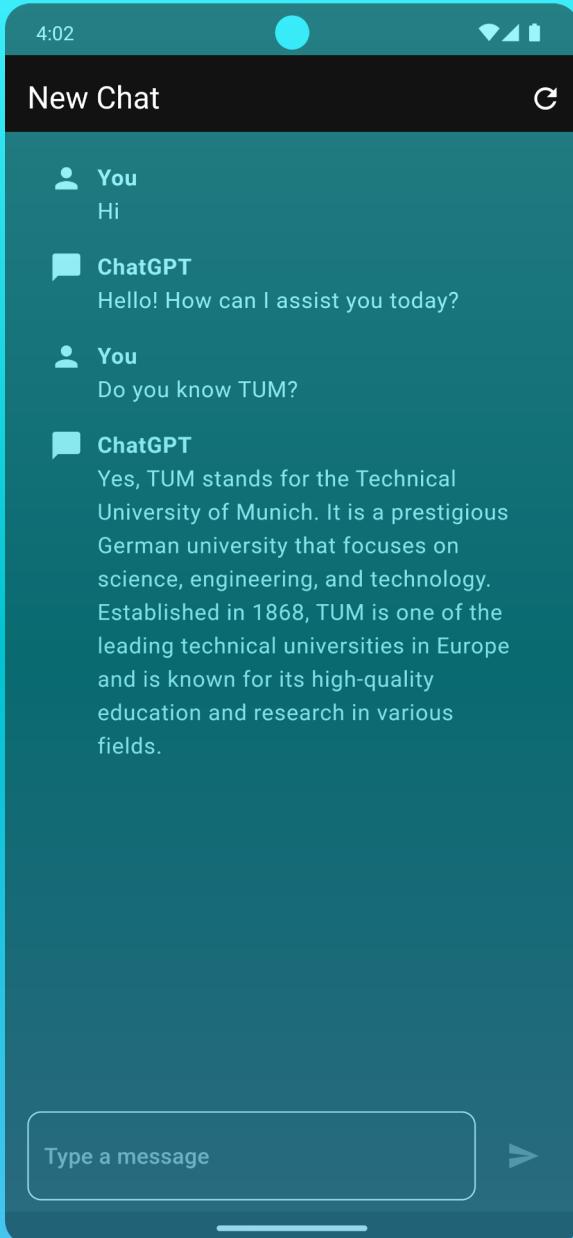
  final Chat chat;

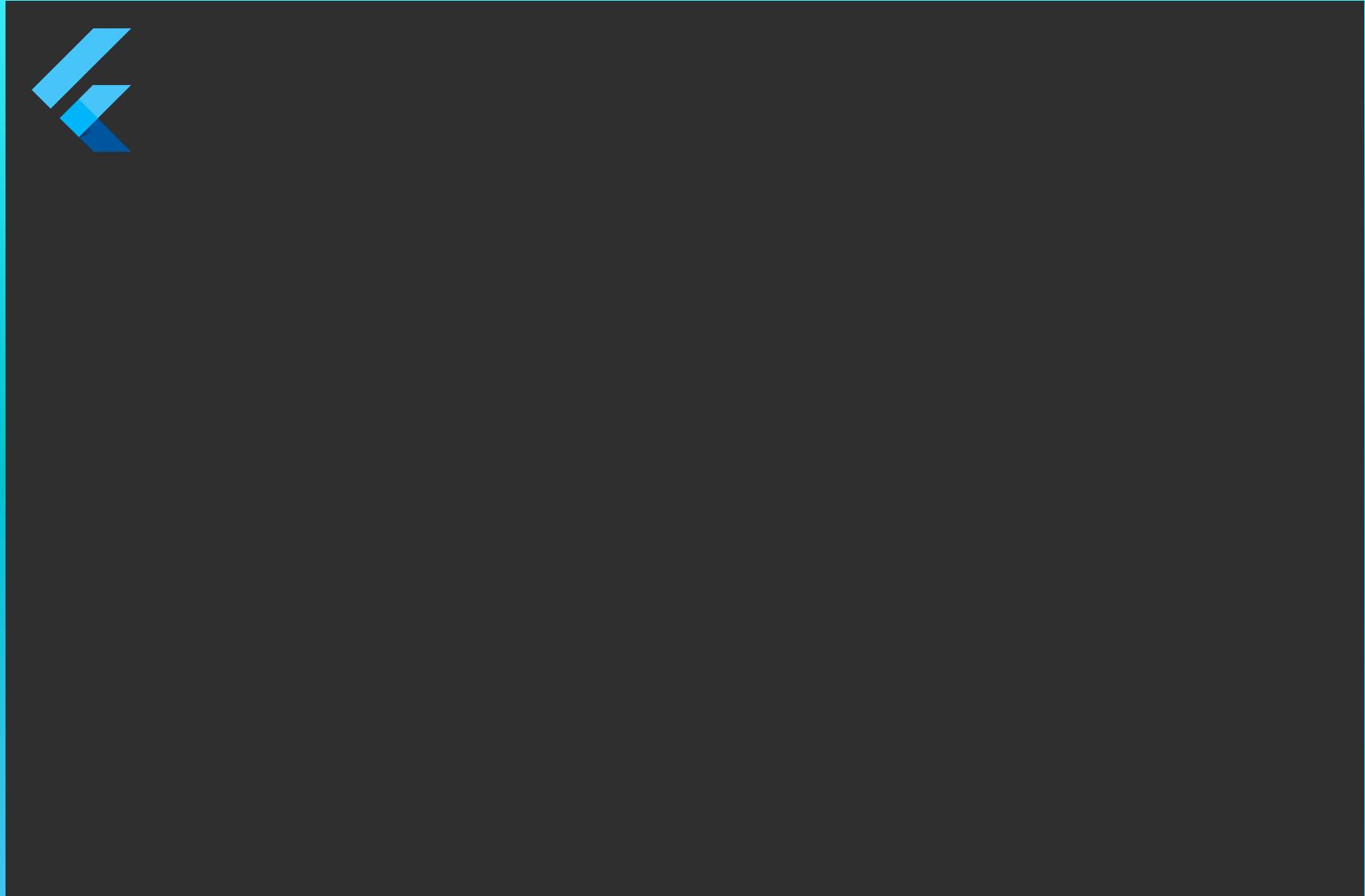
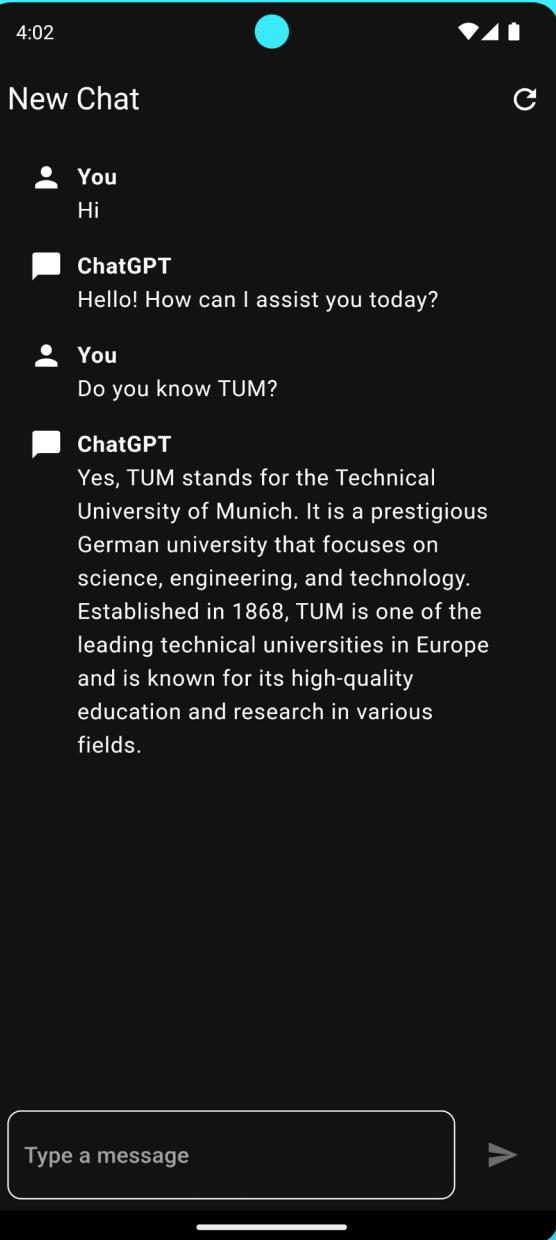
  @override
  Widget build(BuildContext context) {
    if (chat.messages.isEmpty) {
      return Center(
        child: ConstrainedBox(
          constraints: const BoxConstraints(
            maxWidth: 300,
          ),
          child: const FractionallySizedBox(
            widthFactor: 0.5,
            child: Image(
              image:
                AssetImage('assets/images/chatgpt.png'),
            ),
          ),
        );
    }
    return ListView.builder(
      itemBuilder: (context, index) {
        final message = chat.messages[index];
        return ChatMessageTile(message: message);
      },
      itemCount: chat.messages.length,
    );
  }
}

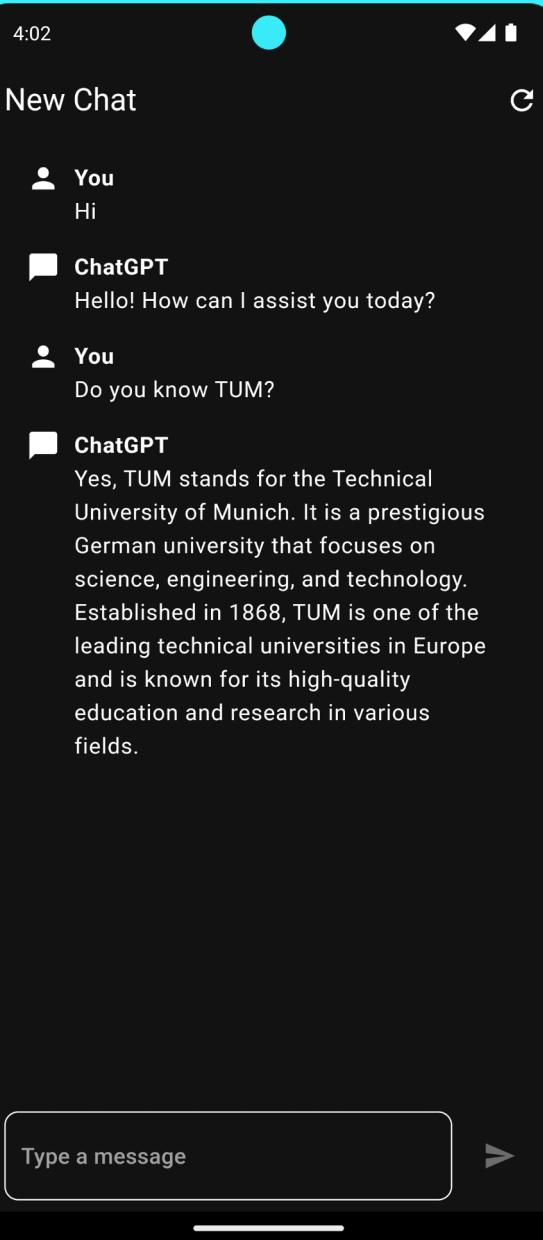
class ChatMessageTile extends StatelessWidget {
  const ChatMessageTile({
    super.key,
    required this.message,
  });

  final ChatMessage message;

  @override
  Widget build(BuildContext context) {
    return ListTile(
      title: Row(
        mainAxisAlignment: CrossAxisAlignment.start,
        children: [
          if (message.sender == Sender.user)
            const Icon(Icons.person)
          else
            const Icon(Icons.chat_bubble),
          const SizedBox(
            width: 10,
          ),
          Expanded(
            child: Column(
              crossAxisAlignment:
CrossAxisAlignment.start,
              children: [
                Text(
                  message.sender.displayText,
                  style: const TextStyle(
                    fontWeight: FontWeight.bold,
                  ),
                ),
                Text(message.text),
              ],
            ),
          ),
        ],
      );
  }
}
```







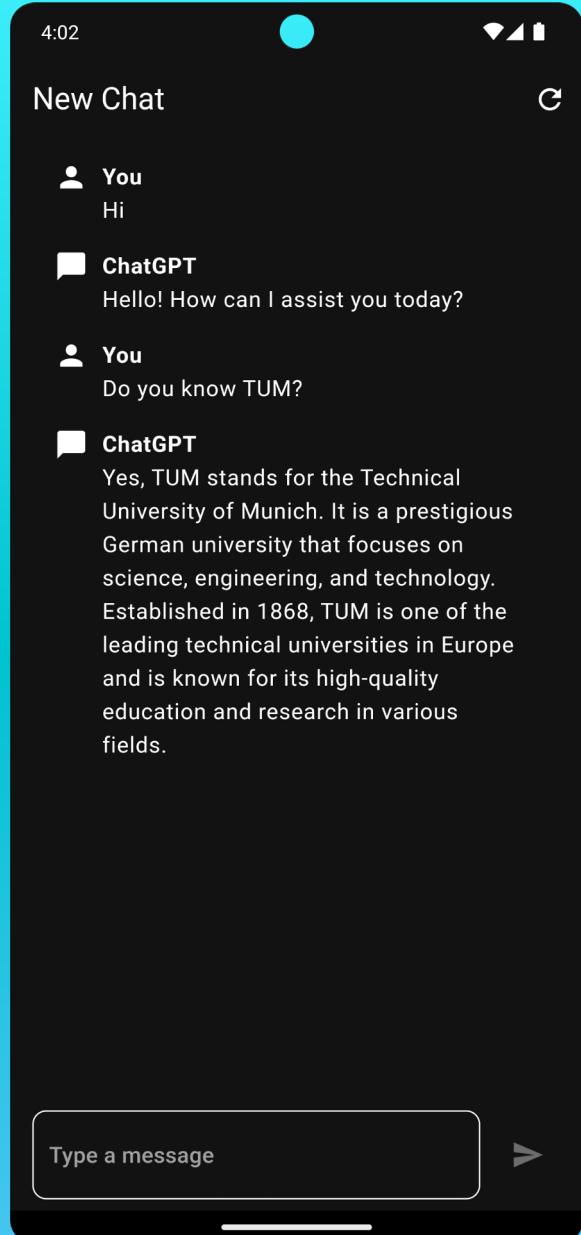
```
class Home extends StatefulWidget {
    const Home({super.key});

    @override
    State<Home> createState() => _HomeState();
}

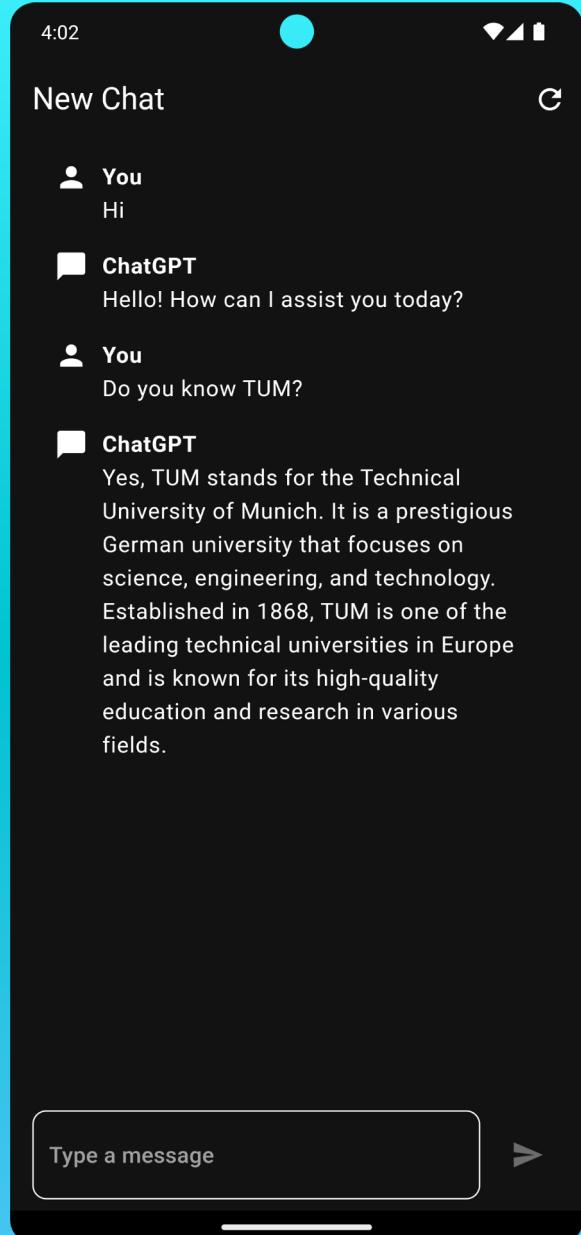
class _HomeState extends State<Home> {
    @override
    Widget build(BuildContext context) {

    }

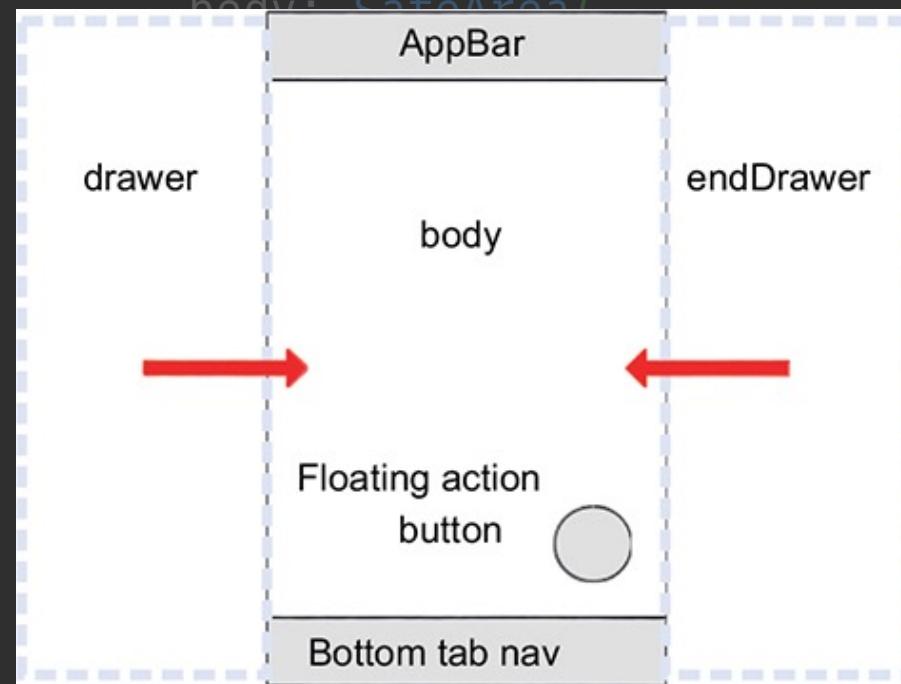
    AppBar appBar(Chat? chat, BuildContext context) {
        ...
    }
}
```



```
@override  
Widget build(BuildContext context) {  
  final chatProvider = context.watch<ChatProvider>();  
  final chat = chatProvider.currentChat;  
  return Scaffold(  
    appBar: buildAppBar(chat, context),  
    body: SafeArea(  
      child: Padding(  
        padding: const EdgeInsets.symmetric(horizontal:  
16, vertical: 8),  
        child: Column(  
          children: [  
            Expanded(  
              child: chat == null  
                ? const Center(  
                    child: CircularProgressIndicator(),  
                  )  
                : ChatDisplay(chat: chat),  
            ),  
            const ChatInput(),  
          ],  
        ),  
      );  
}
```



```
  @override  
  Widget build(BuildContext context) {  
    final chatProvider = context.watch<ChatProvider>();  
    final chat = chatProvider.currentChat;  
    return Scaffold( ➡ Places widgets in a predefined layout  
      appBar: buildAppBar(chat, context),  
      body: SafeArea(  
        child: Padding(  
          padding: const EdgeInsets.symmetric(horizontal:  
            16, vertical: 8),  
        child: Column(  
          children: [  
            Expanded(  
              child: chat == null  
                ? const Center(  
                    child: CircularProgressIndicator(),  
                  )  
                : ChatDisplay(chat: chat),  
            ),  
            const ChatInput(),  
          ],  
        ),  
      );  
  }
```

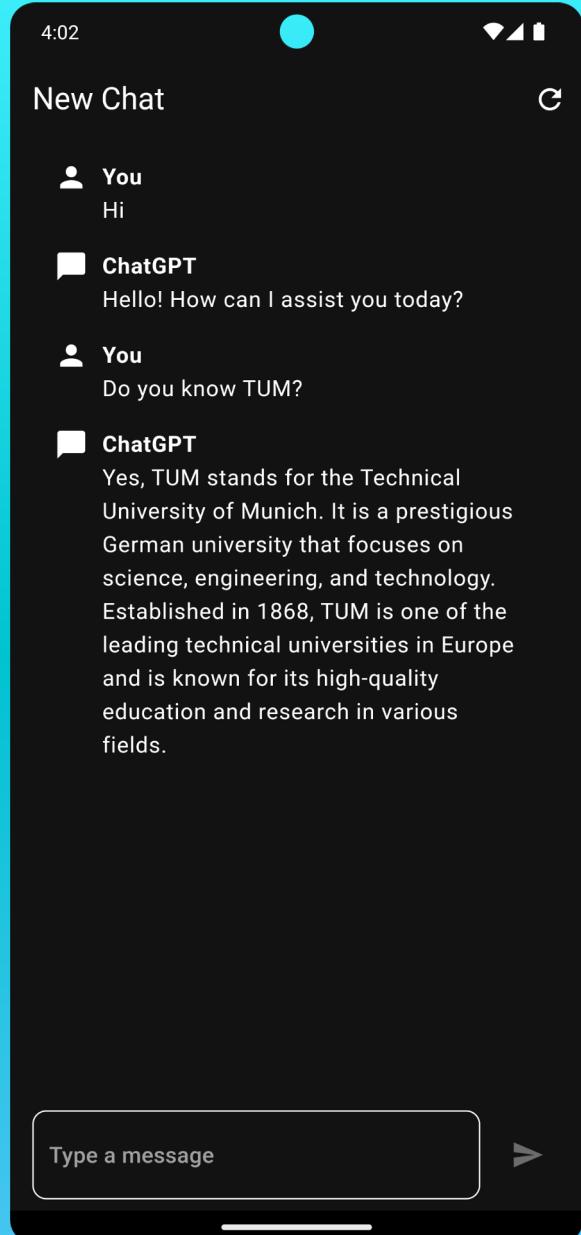


The diagram illustrates the structure of a Scaffold widget. It features a central white area labeled "body". At the top, there is an "AppBar" (gray bar). On the left, a vertical dashed line is labeled "drawer", indicating the position of a side navigation drawer. On the right, another vertical dashed line is labeled "endDrawer", indicating the position of a secondary side navigation drawer. A red arrow points right from the "body" area towards the "endDrawer" line. In the bottom-left corner of the "body" area, there is a "Floating action button" (circle icon). At the very bottom, there is a "Bottom tab nav" (gray bar).

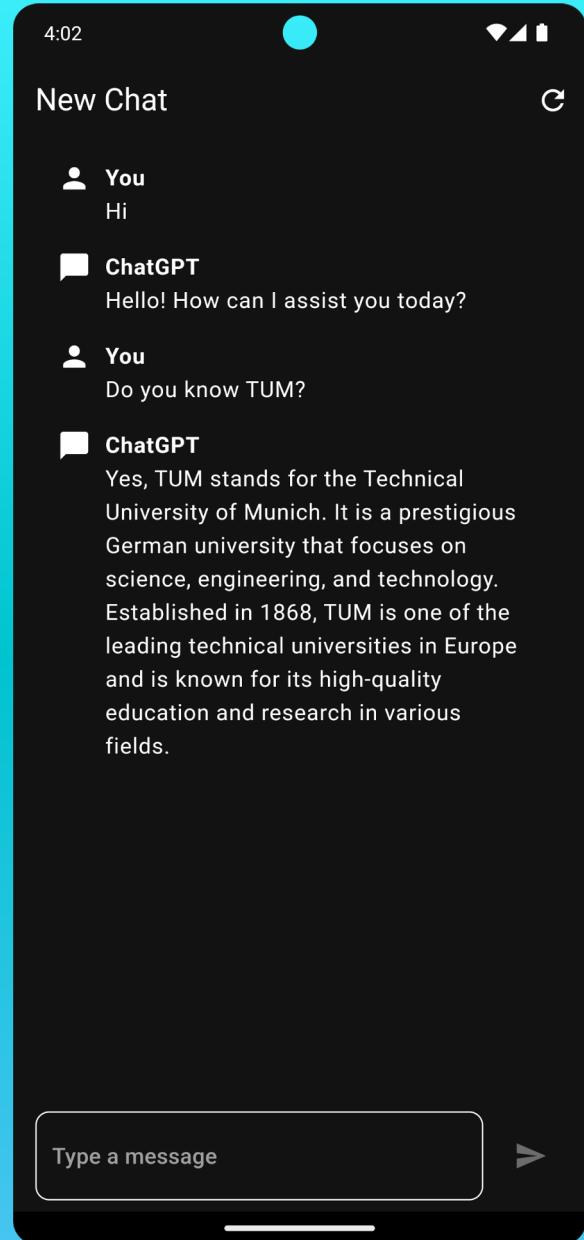
```
@override
Widget build(BuildContext context) {
  final chatProvider = context.watch<ChatProvider>();
  final chat = chatProvider.currentChat;
  return Scaffold( ➔ Places widgets in a predefined layout
    appBar: buildAppBar(chat, context),
    body: SafeArea(
```

ets.symmetric(horizontal:

ull
 er(
 CircularProgressIndicator(),
 (chat: chat),
);
],
),
),
),
),
);
}



```
  @override  
  Widget build(BuildContext context) {  
    final chatProvider = context.watch<ChatProvider>();  
    final chat = chatProvider.currentChat;  
    return Scaffold( ➔ Places widgets in a predefined layout  
      appBar: buildAppBar(chat, context),  
      body: SafeArea(  
        child: Padding( ➔ Applies padding to child  
          padding: const EdgeInsets.symmetric(horizontal:  
            16, vertical: 8),  
        child: Column(  
          children: [  
            Expanded(  
              child: chat == null  
                ? const Center(  
                    child: CircularProgressIndicator(),  
                  )  
                : ChatDisplay(chat: chat),  
            ),  
            const ChatInput(),  
          ],  
        ),  
      );  
  }  
}
```



```
override  
Widget build(BuildContext context) {  
    final chatProvider = context.watch<ChatProvider>();  
    final chat = chatProvider.currentChat;  
    return Scaffold( ➔ Places widgets in a predefined layout  
        appBar: buildAppBar(chat, context),  
        body: SafeArea(  
            child: Padding( ➔ Applies padding to child  
                padding: const EdgeInsets.symmetric(horizontal:  
16, vertical: 8),  
                child: Column(  
                    children: [  
                        Expanded(  
                            child: chat == null ➔ Show loading indicator  
                                ? const Center(  
                                    child: CircularProgressIndicator(),  
                                )  
                                : ChatDisplay(chat: chat),  
                            ),  
                            const ChatInput(),  
                        ],  
                    ),  
                ),  
            ),  
        );  
}
```



```
class Home extends StatefulWidget {
  const Home({super.key});

  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  @override
  void initState() {
    super.initState();
    WidgetsBinding.instance.addPostFrameCallback((_) {
      final authProvider = context.read<AuthProvider>();
      VoidCallback? listener;
      listener = () {
        if (authProvider.currentUser == null) {
          return;
        }
        final chatProvider = context.read<ChatProvider>();
        chatProvider.loadCurrentOrCreateChat();
        authProvider.removeListener(listener!);
      };
      authProvider.addListener(listener);
    });
  }

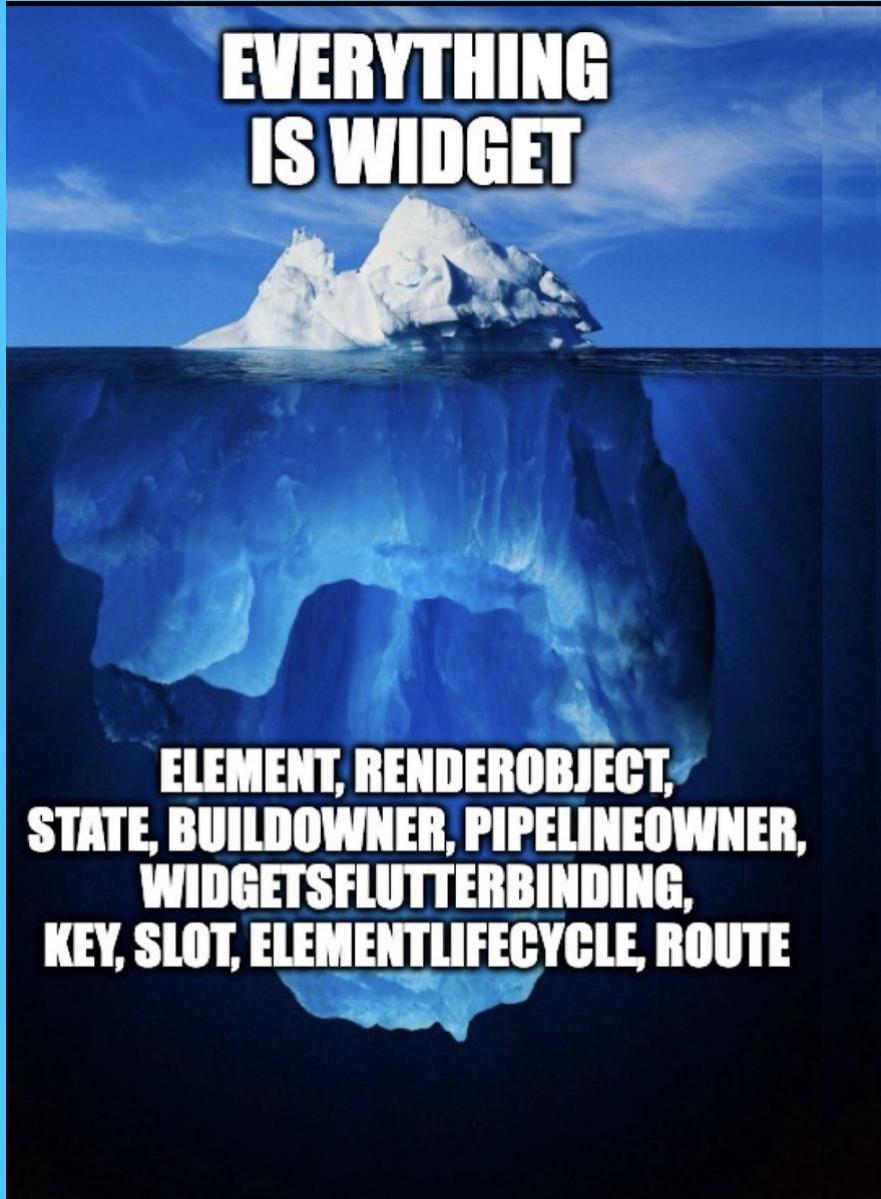
  AppBar buildAppBar(Chat? chat, BuildContext context) {
    return AppBar(
      title: Text(chat?.title ?? 'ChatGPT'),
      actions: [
        IconButton(
          onPressed: () async {
            await context.read<ChatProvider>().startNewChat();
          },
          icon: const Icon(Icons.refresh),
        ),
      ],
    );
  }
}

@override
Widget build(BuildContext context) {
  final authProvider =
  context.watch<AuthProvider>();
  if (authProvider.currentUser == null) {
    return const Scaffold(
      body: Center(
        child: CircularProgressIndicator(),
      ),
    );
  }
  final chatProvider =
  context.watch<ChatProvider>();
  final chat = chatProvider.currentChat;
  return Scaffold(
    appBar: buildAppBar(chat, context),
    body: SafeArea(
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
        child: Column(
          children: [
            Expanded(
              child: chat == null
                ? const Center(
                  child:
CircularProgressIndicator(),
                )
                : ChatDisplay(chat: chat),
            ),
            const ChatInput(),
          ],
        ),
      ),
    );
}
```

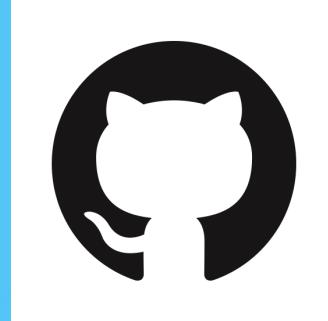




 Google Developer Student Clubs



**ELEMENT, RENDEROBJECT,
STATE, BUILDOWNER, PIPELINEOWNER,
WIDGETSFLUTTERBINDING,
KEY, SLOT, ELEMENTLIFECYCLE, ROUTE**



That's it!

Thank you for listening!
Do you have any
questions?