React Native: Crossplatform fast dive

About me

About me

- Vladimir Ivanov - Lead software engineer

About me

- Vladimir Ivanov - Lead software engineer
- More than 7 years in Android development

About me

- Vladimir Ivanov - Lead software engineer
- More than 7 years in Android development
- Wide interest in Mobile technologies

Why crossplatform?

Why crossplatform?

1. Effective development

Why crossplatform?

1. Effective development
2. because rapid

Why crossplatform?

1. Effective development
2. because rapid
3. 3x faster than native approach

Why crossplatform?

1. Effective development
2. because rapid
3. 3x faster than native approach
4. Native UX, at last

How to dive?

How to dive?

1. Install node

How to dive?

1. Install node
2. Learn React Native

How to dive?

1. Install node
2. Learn React Native
3. create-react-native-app

How to dive?

1. Install node
2. Learn React Native
3. create-react-native-app
4. Done

Let’s dive

What we get

```
$ ls -l
```

What have we got

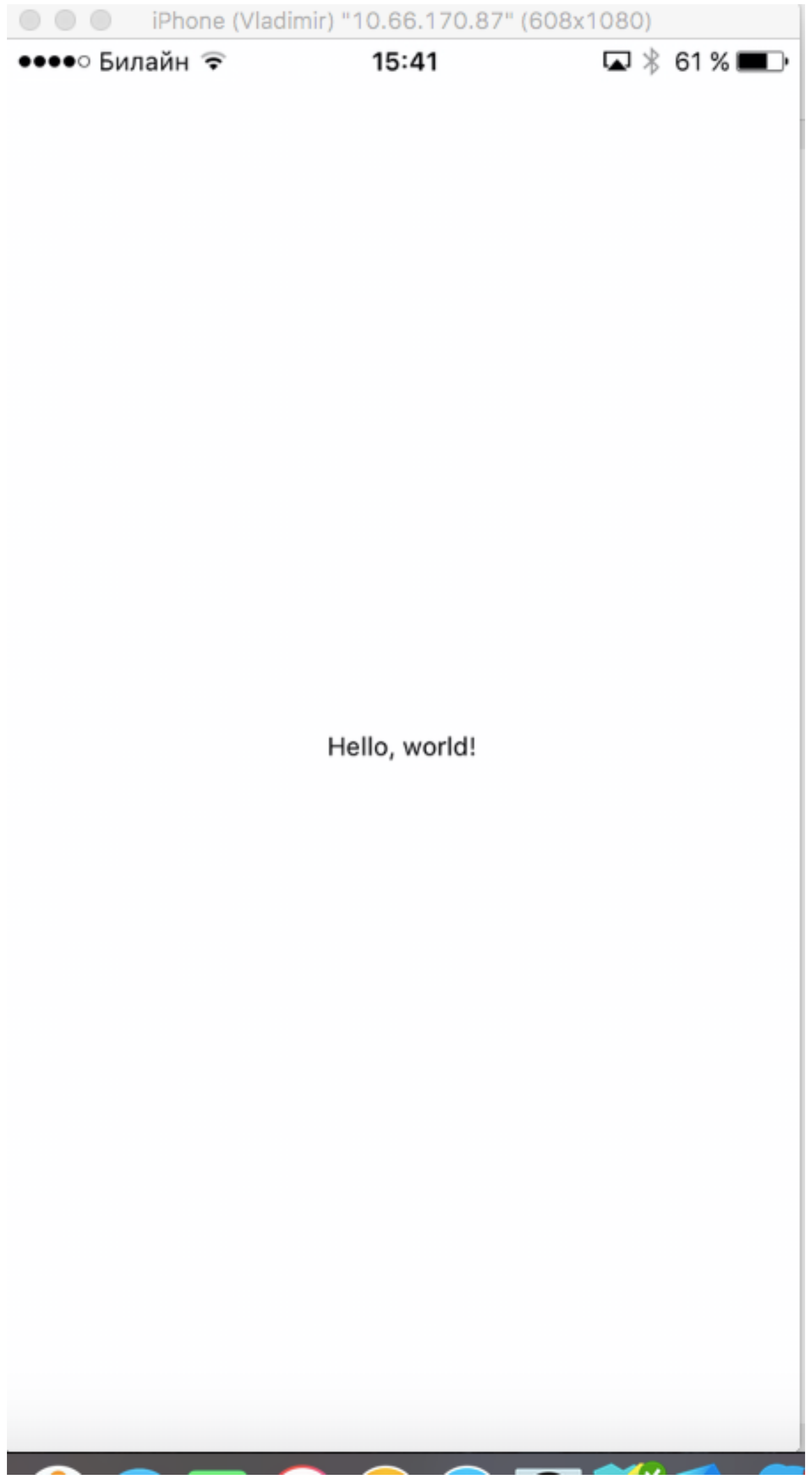
```
$ ls -l

App.js
App.test.js
README.md
app.json
node_modules
package.json
```

App.js

```
import React from 'react';
import { Text, View } from 'react-native';

export default class App extends React.Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <Text>Hello, world!</Text>
      </View>
    );
  }
}
```



Hello, world!

App.js

```
import React from 'react';
import { Text, View } from 'react-native';

export default class App extends React.Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <Text>Hello, world!</Text>
      </View>
    );
  }
}
```

App.js

```
import React from 'react';
import { Text, View } from 'react-native';

export default class App extends React.Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <Text>Hello, world!</Text>
      </View>
    );
  }
}
```

App.js

```
import React from 'react';
import { Text, View } from 'react-native';

export default class App extends React.Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <Text>Hello, world!</Text>
      </View>
    );
  }
}
```

App.js

```
import React from 'react';
import { Text, View } from 'react-native';

export default class App extends React.Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <Text>Hello, world!</Text>
      </View>
    );
  }
}
```

App.js

```
import React from 'react';
import { Text, View } from 'react-native';

export default class App extends React.Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <Text>Hello, world!</Text>
      </View>
    );
  }
}
```

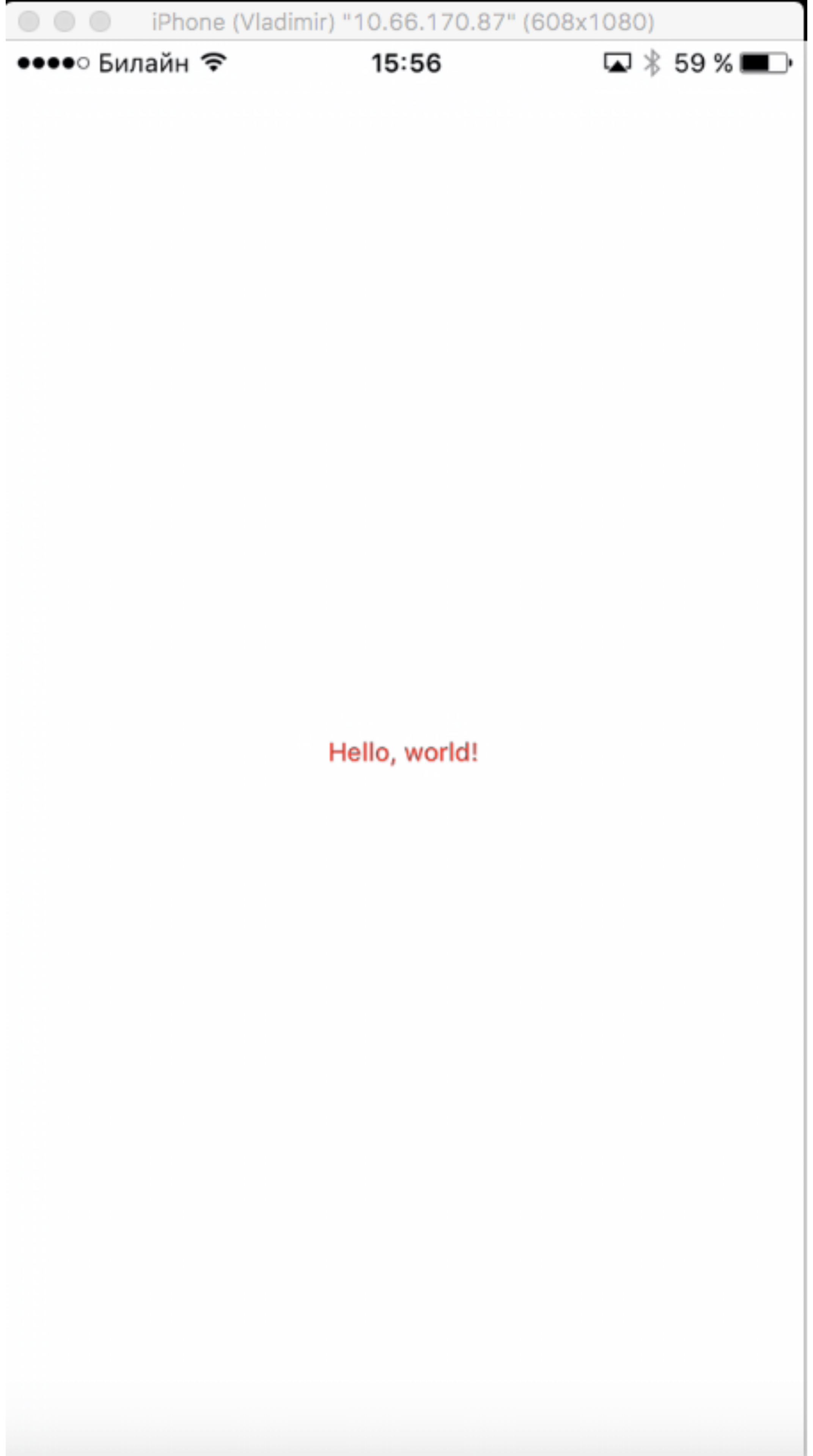
App.js

```
import React from 'react';
import { Text, View } from 'react-native';

export default class App extends React.Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <Text>Hello, world!</Text>
      </View>
    );
  }
}
```

App.js

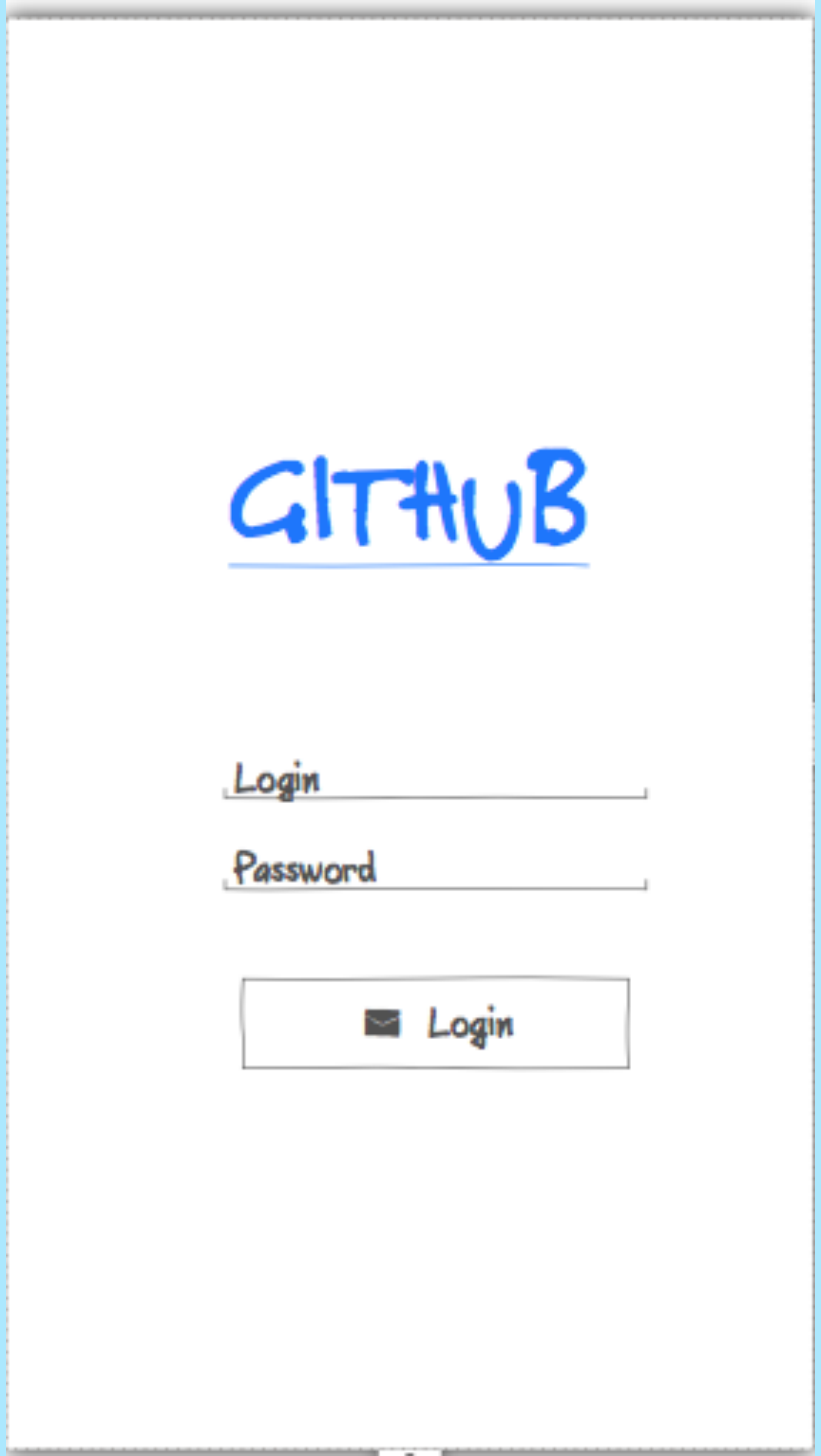
```
<Text style={{ color: '#F00' }}>
    Hello, world!
</Text>
```


iPhone (Vladimir) "10.66.170.87" (608x1080)

●●●●○ Билайн 15:56 59 %

Hello, world!

Let’s make something more interesting



How?

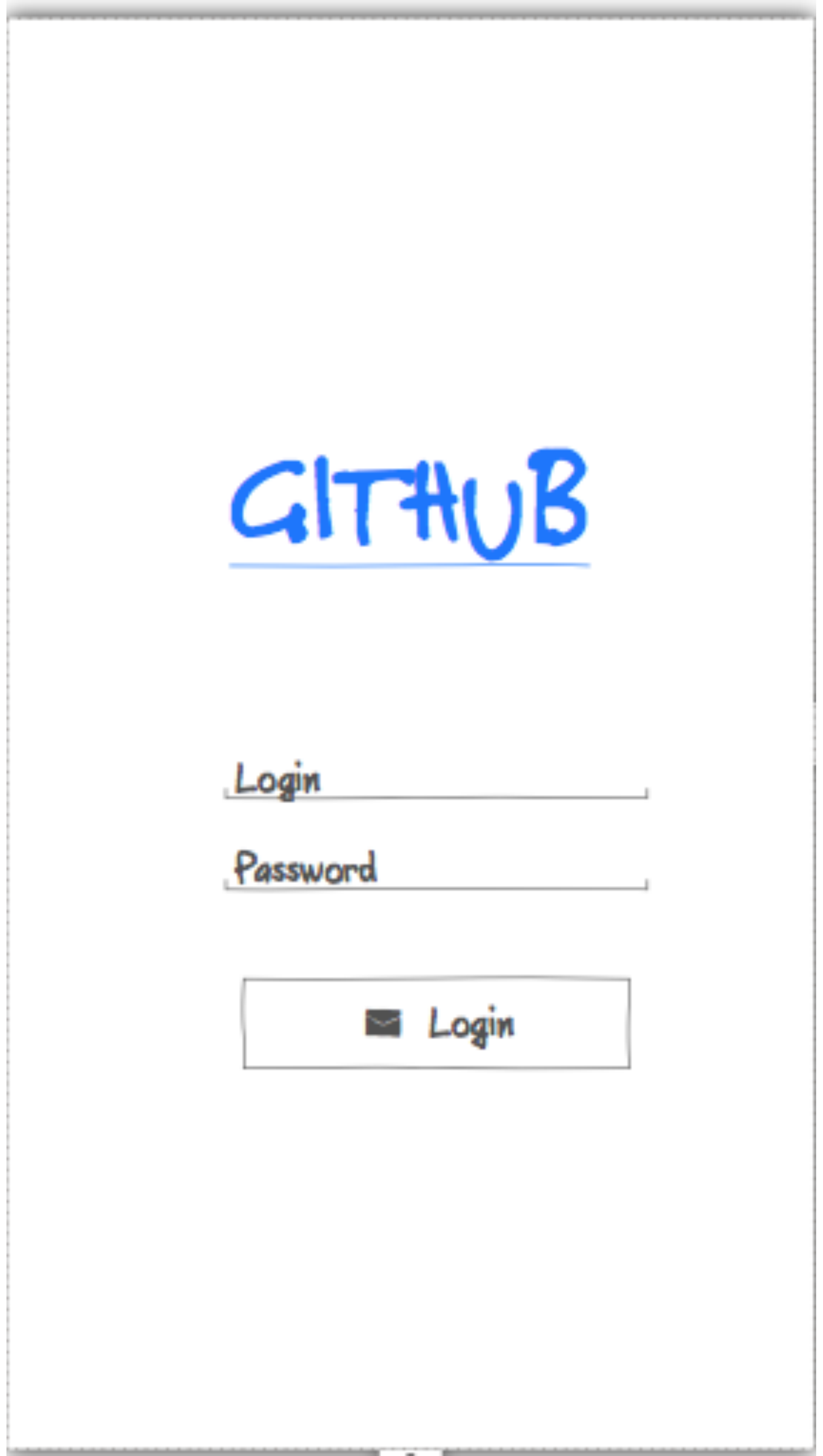
How?

1. React page is a component tree

How?

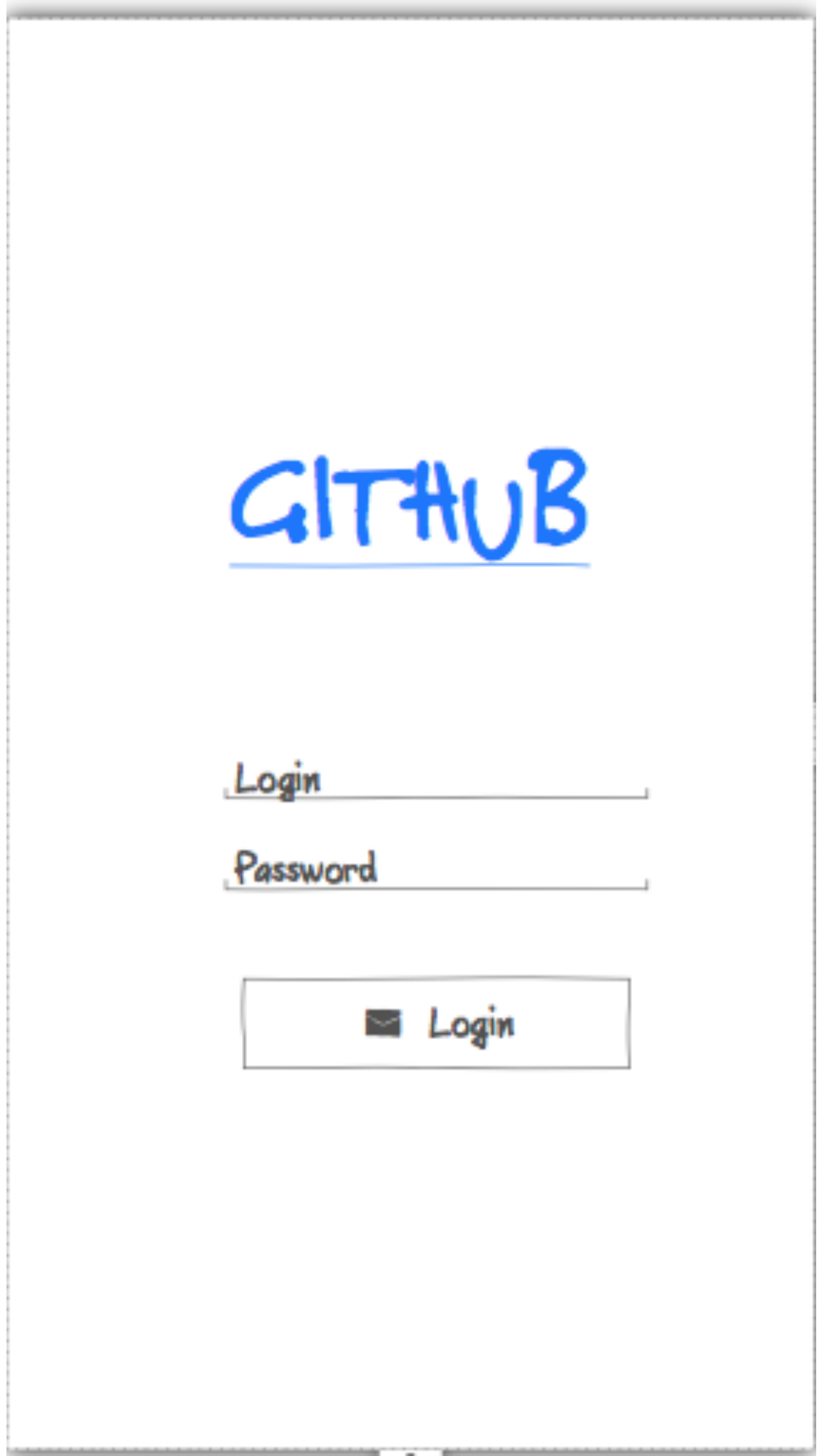
1. React page is a component tree
2. Each piece of UI should be a component

4 main components



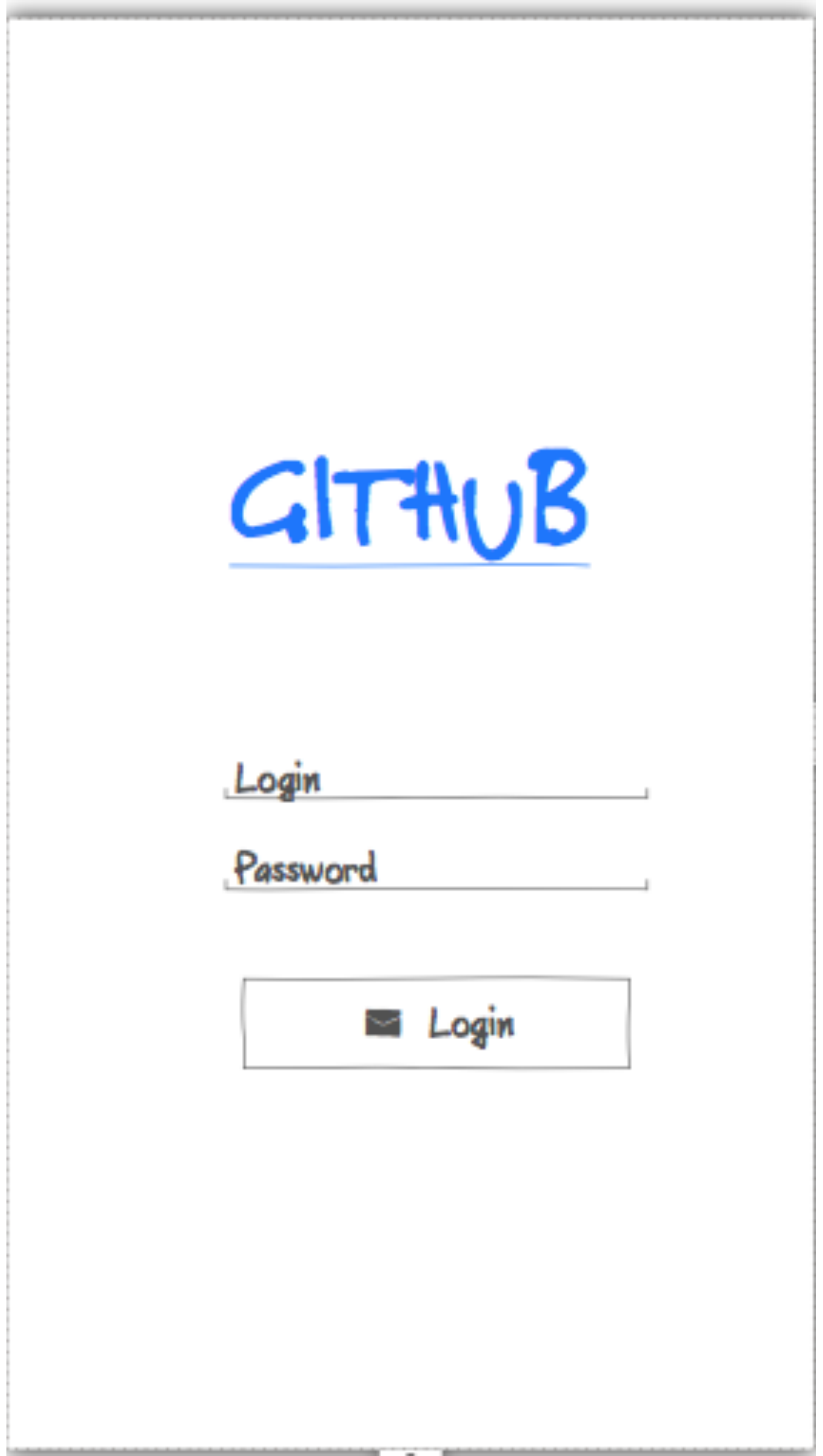
4 main components

1. Logo



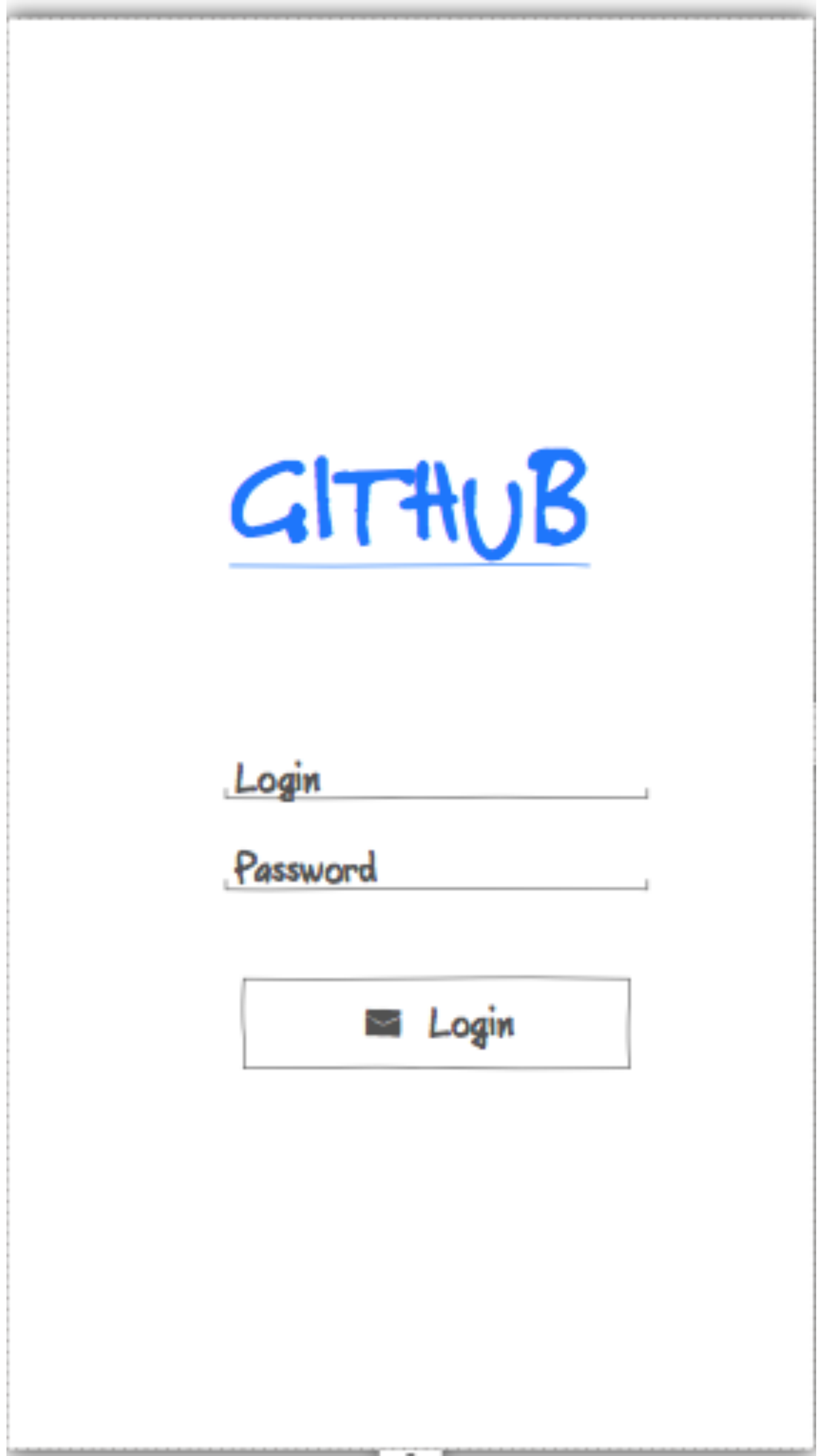
4 main components

1. Logo
2. Inputs



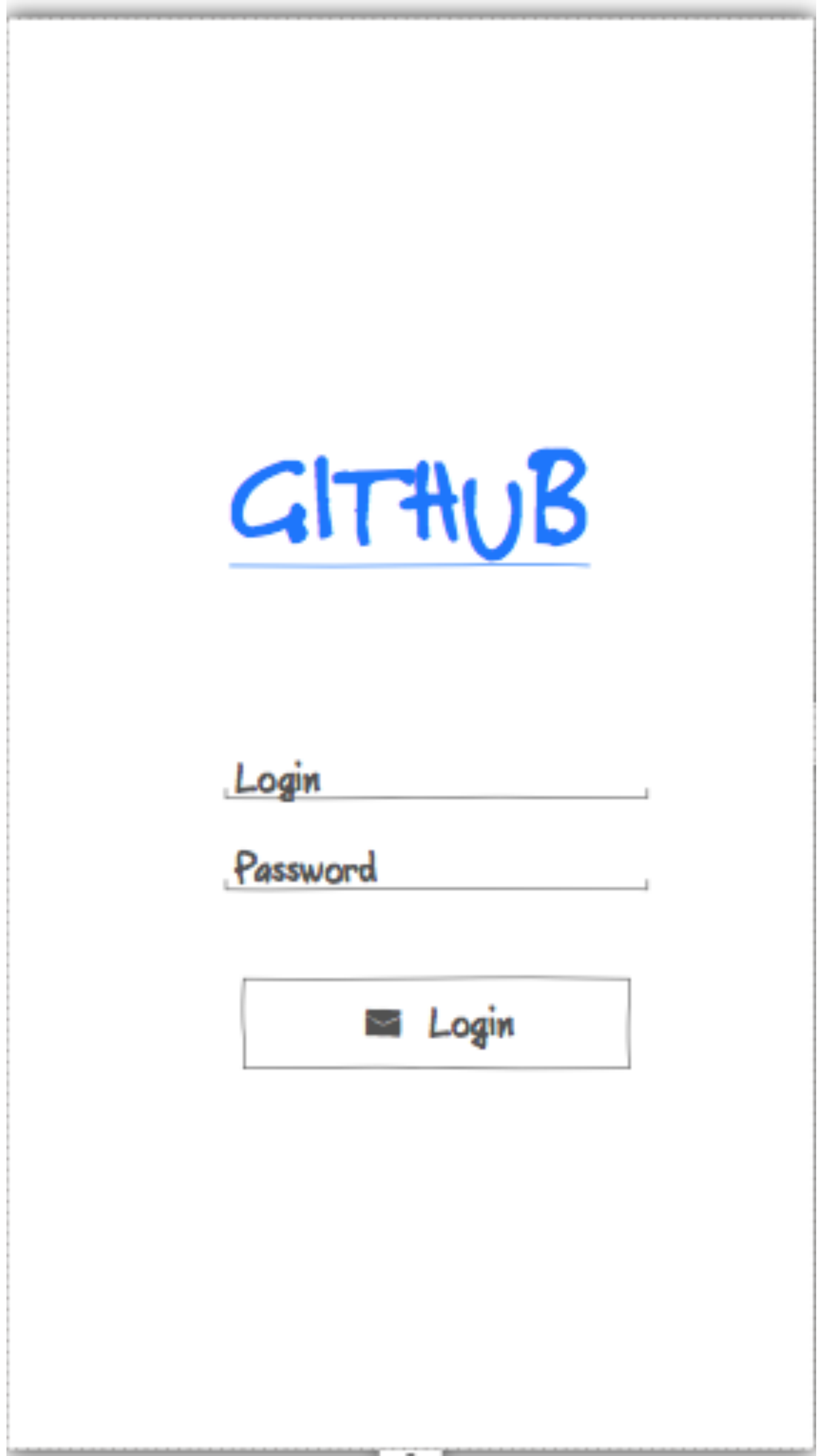
4 main components

1. Logo
2. Inputs
3. Submit button



4 main components

1. Logo
2. Inputs
3. Submit button
4. Optional message



Logo component

```
import React from 'react';
import {Image, View} from "react-native";

export default Logo = () => (
    <View style={{alignItems: 'center'}}>
        <Image
            source={require('./../../../GitHub-Logo.png')}
        />
    </View>
);
```

Logo component

```
export default Logo = () => (
    ...
);
```

Components can be

Components can be

1. Functional - no lifecycle, no state, only JSX

Components can be

1. Functional - no lifecycle, no state, only JSX
2. Class based - lifecycle, state, usage in redux, etc.

Logo component

```
export default Logo = () => (

    <View style={{alignItems: 'center'}}>
            <Image
                    source={require('./../../../GitHub-Logo.png')}
            />
    </View>
)
```

9:41
100 %
GitHub

Login inputs

Login inputs

1. Render login input

Login inputs

1. Render login input
2. Render password input(as hidden)

Login inputs

1. Render login input
2. Render password input(as hidden)
3. Pass somehow login and password to submit function

Login inputs

```
export default LoginInputs = ({ onChangeValue }) => (
    <View style={{ margin: 16 }}>
        <FormInput
            placeholder='login'
            onChangeText={(value) => {
                onChangeValue('login', value);
            }}
        />
        <FormInput
            secureTextEntry
            placeholder='password'
            onChangeText={(value) => {
                onChangeValue('password', value);
            }}
        />
    </View>
);
```

Login inputs

```
export default LoginInputs = ({ onChangeValue }) => (
    …
);
```

Login inputs

```
<View style={{ margin: 16 }}>
    <FormInput
        placeholder='login'
        onChangeText={(value) => {
            onChangeValue('login', value);
        }}
    />
    <FormInput
        secureTextEntry
        placeholder='password'
        onChangeText={(value) => {
            onChangeValue('password', value);
        }}
    />
</View>)
```

Login inputs

```
export default LoginInputs = ({ onChangeValue }) => (

    <View style={{ margin: 16 }}>
            { … }
    </View>

)
```

Login inputs

```
<FormInput
    placeholder='login'
    onChangeText={(value) => {
        onChangeValue('login', value);
    }}
/>
<FormInput
    secureTextEntry
    placeholder='password'
    onChangeText={(value) => {
        onChangeValue('password', value);
    }}
/>
```

Login inputs

```
<FormInput
     placeholder='login'
     onChangeText={(value) => {
         onChangeValue('login', value);
     }}
/>
<FormInput />
```

Login inputs

```
<FormInput />
<FormInput
    secureTextEntry
    placeholder='password'
    onChangeText={(value) => {
        onChangeValue('password', value);
    }}
/>
```

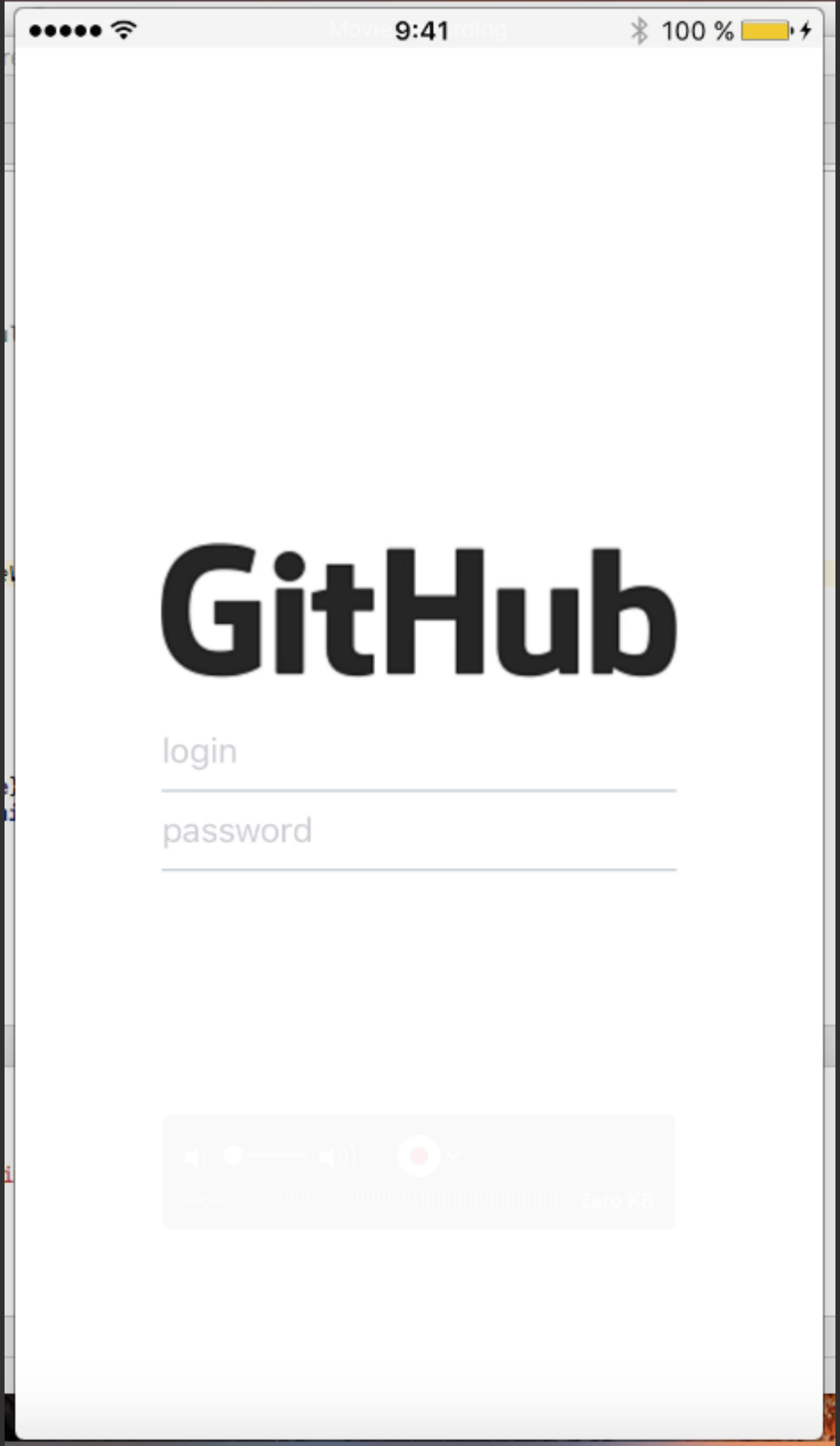
LoginScreen.js

```
render() {
        const {container} = styles;

        return (

                <View style={container}>
                   <Logo />
                   <LoginInputs …/>
                </View>


        )
    }
```


9:41
100 %
GitHub
login
password

Login inputs

Value propagation

Component state

Component state

1. Each class based component has state

Component state

1. Each class based component has state
2. State is no more than a javascript object

Component state

1. Each class based component has state
2. State is no more than a javascript object
3. Updating state is async, but this is not really important now

Component state

1. Each class based component has state
2. State is no more than a javascript object
3. Updating state is async, but this is not really important now
4. Updating state happens with this.setState() function

Saving login and password to screen state

```
…
class LoginScreen extends Component {
    state = { error: null };
…
}
```

Saving login and password to screen state

```
…

class LoginScreen extends Component {

    state = { error: null };

    onChangeValue = (prop, value) => {
        this.setState({ [prop]: value });
    };

…
}
```

Saving login and password to screen state

```
{ login: ‘v’ }
{ login: ‘vl’ }
{ login: ‘vli’ }

…

{ login: ‘vlivanov’, password: ’1’ }
{ login: ‘vlivanov’, password: ’12’ }
…
{ login: ‘vlivanov’, password: ’123abc123’ }
```

Submit button

```
<Button
    raised
    title='LOGIN'
    backgroundColor="#00F"
    icon={{name: 'cached'}}
    onPress={this.doLogin}
/>
```

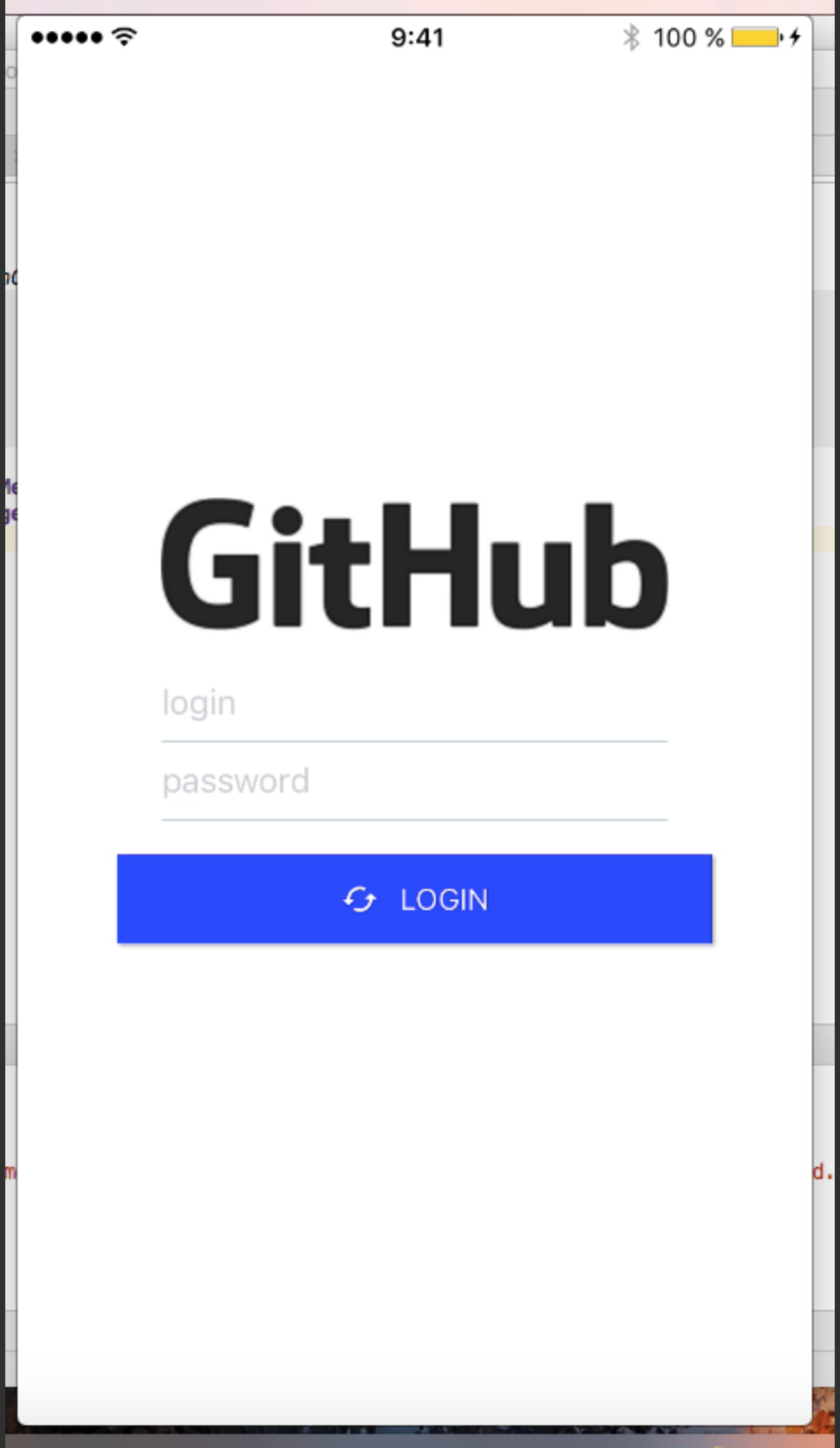
LoginScreen.js

```
render() {

    return (

        <View style={container}>
            <Logo />
            <LoginInputs onChangeValue={…}/>
            <Button
                …
            />
       </View>


    )
}
```


9:41
100 %
GitHub
login
password
LOGIN

Submit button

```
<Button
    …
    onPress={this.doLogin}
/>
```

Submit button

```
doLogin = async () => {
    const { login, password } = this.state;
    let result = await loginAsync(login, password);
    this.setState({
      loggedIn: result.error === undefined,
      error:  result.error
    });
};
```

Submit button

```
doLogin = async () => {
    const { login, password } = this.state;
        …
};
```

Submit button

```
doLogin = async () => {
    const { login, password } = this.state;
    let result = await loginAsync(login, password);
      …
};
```

Submit button

```
doLogin = async () => {
    const { login, password } = this.state;
    let result = await loginAsync(login, password);
    this.setState({
      loggedIn: result.error === undefined,
      error:  result.error
    });
};
```

loginAsync

```
export const loginAsync = async (login, password) => {
    let base64 = encode(`${login}:${password}`);
    try {
        let result = await fetch('https://api.github.com/user', {
            method: 'GET',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json',
                'Authorization': 'Basic ' + base64
            },
        });
        if (result.status === 200) {
            return {
                user: JSON.parse(result._bodyInit),
                auth: base64
            };
        } else {
            return { error: `Failed to login with ${result.status}` };
        }
    } catch (error) {
        console.log("[LoginActions] error = " + JSON.stringify(error));
        return { error: `Failed to login with ${result.error}` };
    }
};
```

LoginScreen.js

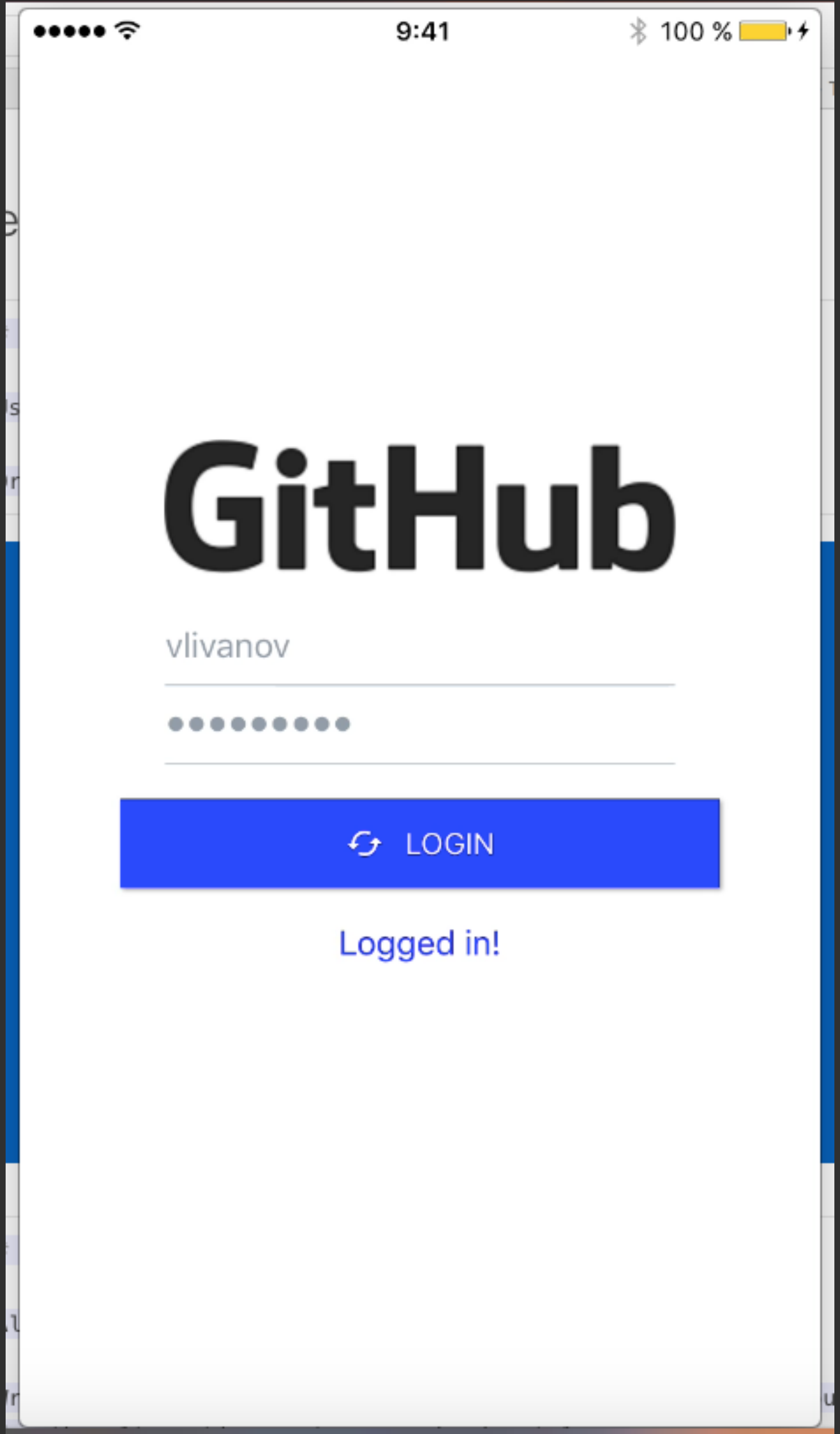
```
render() {
    const {container, successMessage, errorMessage} = styles;

    return (
        <View style={container}>
            <Logo />
            <LoginInputs …/>
            <Button
                …
            />
            {this.state.loggedIn
                && <Text style={successMessage}>Logged in!</Text>
            }
            {this.state.error
                && <Text style={errorMessage}>{this.state.error}</Text>
            }
        </View>
    )
}
```

doLogin

```
doLogin = async () => {
    const { login, password } = this.state
    let result = await loginAsync(login, password)

    this.setState({
        loggedIn: result.error === undefined,
        error:  result.error
    });
}
```


9:41
100 %
GitHub
vlivanov
••••••••••
LOGIN
Logged in!

Stats

Stats

— 153 lines

Stats

- 153 lines
- 1 hour of development time

Useful links

Useful links

— https://facebook.github.io/react-native/docs/getting-started.html

Useful links

- https://facebook.github.io/react-native/docs/getting-started.html
- https://www.udemy.com/the-complete-react-native-and-redux-course/

Useful links

- https://facebook.github.io/react-native/docs/getting-started.html
- https://www.udemy.com/the-complete-react-native-and-redux-course/
- https://expo.io

Useful links

- https://facebook.github.io/react-native/docs/getting-started.html
- https://www.udemy.com/the-complete-react-native-and-redux-course/
- https://expo.io
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Questions?