Summary of chat-react project:

1. on command line, navigate to desired directory for new app

2. invoke `create-app-react bloc-chat-react`

3. go to Firebase website, create new project and create new Realtime NoSQL Database

4. within Data section of the database, create a rooms key, with nested incremental numbers keys (1-3), and then nested key:value pairs corresponding to 3 rooms (`room1` , `room2` , and `room3`)

5. within the Rules section of the database, set the rules using an object that contains a single parent key `rules` . Nested within `rules` , place 2 key-value pairs using `.read` and `.write` as the keys, setting values for both keys to `true` . Each key-value pair is separated by a comma, NOT a semicolon

6. go back to command line and `cd` to to the project directory. Use `npm install -S firebase` to add firebase to the project

7. import firebase to `App.js` file with `import * as firebase from 'firebase'`

8. go back to Firebase website and register the bloc-chat-react

project with Firebase, and choose option to add Firebase to the web application. Paste the provided boilerplate code to `App.js`. The boilerplate code assigns to a variable `config` an object containing the API key and other identification codes Firebase creates for the application

9. Create a `components` folder within the src folder for the project Within `components`, create a new file `RoomList.js` which will contain a new React component called `RoomList`

10. Go back to `App.js`, and within the App component's `render()` function add `RoomList` to the `return` statement and pass firebase to the RoomList component as a prop:

```
class App extends Component {
render() {
return (
  <RoomList firebase={firebase}/>
);
}
}
```

11. import the `RoomList` component to `App.js`:

```
import RoomList from './components/RoomList';
```

12. go to RoomList.js and:

- import React:

```
import React, {Component} from 'react';
```

- create the basic RoomList class Component with the constructor() and render() methods required by React class components:

```
class RoomList extends Component {
    constructor(props){
        super(props);
    }
}

render(){
    return();
}
```

- export the RoomList component to App.js by adding an export statement to the bottom of RoomList.js:

```
export default RoomList;
```

13. The RoomList component will hold state for the list of rooms (currently `rooms1`, `rooms2`, `rooms3` as created on Firebase site), so create empty `rooms` array to define this state within the `constructor()` method of `RoomList` component:

```
this.state={rooms: []};
```

14. Create a firebase reference using the `.firebase.database().ref()` method within RoomList's `constructor()` method in order for the application to be able to access the rooms key in the firebase database. `App.js` passes firebase to `RoomList` as a prop. Assign the reference to `roomsRef`:

```
this.roomsRef = this.props.firebase.database().ref('rooms'
);
```

15. While `RoomList` is mounted, we want to access the Firebase database and update its information within the application. Use the `componentDidMount()` method to access the firebase reference `roomsRef`. Apply the `.on()` method and the `child-added` event type to access the returned snapshot object's `key` and `val` values and assign them to variables and update the `rooms` state for `RoomList`. When updating state, use the `.concat()` method rather than .push() so that we are not mutating the `rooms` array:

```
componentDidMount(){
this.roomsRef.on('child_added', snapshot => {
    const room = snapshot.val();
    room.key = snapshot.key;
    this.setState({rooms: this.state.rooms.concat(room) }
```

```
);
});
}
```

17.Add to the `render()` method for `RoomList` the JSX so that the current list of rooms in the database is displayed. Remember with JSX syntax the javascript code must be enclosed by curly braces:

```
render(){
    return(
      <nav className='chatRooms'>
        <ul>
          {
            this.state.rooms.map((room,index) =>
              <li key={index}>{room.name}</li>
            )
          }
        </ul>
      </nav>
    );
  }
```

18. Add a form in the `RoomList` component for creating a new room. Form should include a text field and a submit button. On submit, a `createRoom` method is executed that pushes a new room to Firebase with the name given in the text field:

```
 import React, { Component } from 'react';

class RoomList extends Component{
//constructor method
 constructor(props){
   super(props);
   this.state={
     rooms:[],
     newRoomName: ''
   };
   this.roomsRef = this.props.firebase.database().ref('roo
ms');
 }
//end constructor method

 componentDidMount(){
   this.roomsRef.on('child_added', snapshot => {
     const room = snapshot.val();
     room.key = snapshot.key;
     this.setState({rooms: this.state.rooms.concat(room) }
);
   });
 }

 handleChange(e){
   this.setState({newRoomName: e.target.value});
 }
```

```
 createRoom(e){
   e.preventDefault();
   if(!this.state.newRoomName){return}
   const newRoom = {name: this.state.newRoomName};
   this.roomsRef.push(newRoom);
   this.setState({newRoomName: ''});
 }

//required render method within React component
 render(){
   return(
     <section>
     <h1>Bloc Chat Rooms</h1>
     <nav className='chatRooms'>
       <ul>
         {
           this.state.rooms.map((room,index) =>
             <li key={index}>{room.name}</li>
           )
         }
       </ul>
     </nav>
     <form onSubmit={(e)=> this.createRoom(e)}>
       <label for id='newRoomText'>Create a new room</labe
l>
       <input
       id ='newRoomText'
```

```
        type ='text'
        placeholder ='enter name here...'
        value = {this.state.newRoomName}
        onChange={(e)=> this.handleChange(e)}
        />
        <input
        type = 'submit'
        />
      </form>
      </section>
    );
 }
//end render method


}
//end RoomList component

export default RoomList;
```

19. Create a `MessageList` component that receives `firebase` as a prop and uses the `child_added` event handler function to add messages to the state:

```
import React, {Component} from 'react';

class MessageList extends Component {
  constructor(props){
```

```
    super(props);
    this.state = { messages: [] };
    this.messagesRef = this.props.firebase.database().ref(
'messages');
  }

  componentDidMount() {
        this.messagesRef.on('child_added', snapshot => {
            const message = snapshot.val();
            message.key = snapshot.key;
            this.setState({ messages: this.state.messages.
concat( message ) })
        });
    }
    ```

    20. Set an active room, which is stored as state in `A
pp.js` component. When user clicks name of room in the `Ro
omList` component, this becomes the active room:
    ```javascript
    import React, { Component } from 'react';
import './App.css';
import * as firebase from 'firebase';
import RoomList from './components/RoomList';
import MessageList from './components/MessageList';

  // Initialize Firebase
  var config = {
```

```
    apiKey: "AIzaSyCdZA6va-NxMgpnM41vrOnIRfzKV8epg4g",
    authDomain: "bloc-chat-react-a6744.firebaseapp.com",
    databaseURL: "https://bloc-chat-react-a6744.firebaseio
.com",
    projectId: "bloc-chat-react-a6744",
    storageBucket: "bloc-chat-react-a6744.appspot.com",
    messagingSenderId: "503080022258"
  };
  firebase.initializeApp(config);



class App extends Component {
  constructor(props){
    super(props);
    this.state={ activeRoom: []};
  }
//end of constructor method

  handleRoomClick(room){
      this.setState({activeRoom: room});
  }

  render() {
    return (
      <div className='App'>

      <section className='ChatRooms'>
```

```
          <RoomList
          firebase={firebase}
          currentRoom={this.state.activeRoom}
          handleRoomClick={(room)=> this.handleRoomClick(roo
m)}
          />
        </section>

        <section>
          <MessageList
          firebase={firebase}
          currentRoom={this.state.activeRoom}
          />
        </section>

        </div>
    );
  }
}

export default App;
```

```
import React, {Component} from 'react';

class MessageList extends Component {
  constructor(props){
    super(props);
```

```
    this.state = { messages: [] };
    this.messagesRef = this.props.firebase.database().ref(
'messages');
  }

  componentDidMount() {
        this.messagesRef.on('child_added', snapshot => {
            const message = snapshot.val();
            message.key = snapshot.key;
            this.setState({ messages: this.state.messages.
concat( message ) })
        });
    }

    currentMessages(){
      var messageCondition = this.state.messages.filter(me
ssage => message.roomId === this.props.currentRoom.key);
      console.log(messageCondition);
      return messageCondition;
    }

    render() {
      return (
        <div>
          <section className="messages">
            <h2>{(this.props.currentRoom.name)}</h2>
            <p>{console.log(this.props.currentRoom.key)}</
p>
```

```
            <p>{console.log(this.state.messages)}</p>
            <nav>
              {
                this.state.messages
                .filter(message => this.props.currentRoom.
key == '-'+message.roomId)
                .map((message,index) =>
                  <div
                  key={index}
                  >
                  <div>{message.content}</div>
                  <div>{message.username}</div>
                  <div>{message.sentAt}</div>
                  </div>
                )
              }
            </nav>
          </section>
        </div>
      );
    }
  }

export default MessageList;
```

21. Create a few messages manually on the firebase site where each message has 4 properties: `username`, `content`, `sentAt`, and

`roomId`.

22. Enable authentication by clicking on “Authentication” tab on firebase site. Choose using Google as provider for sign-in.

23. Create `User` component that renders a sign-in button. On click, the button calls the firebase `signInWithPopup` method. Also add a sign-out button, that calls firebase’s `signOut` method. In order to respond to these methods, add a `componentDidMount` method that registers an `onAuthStateChanged` event handler. Finally, render `this.props.user.displayName` in this component:

```
import React, { Component } from 'react';

class User extends Component {
  constructor(props){
    super(props);
    this.state={user: ''};
  }

  componentDidMount(){
    this.props.firebase.auth().onAuthStateChanged(user =>
{
      this.props.setUser(user);
    });
  }
```

```
  handleSignIn(e){
    const provider = new this.props.firebase.auth.GoogleAu
thProvider();
    this.props.firebase.auth().signInWithPopup( provider )
;
  }

  handleSignOut(e){
    this.props.firebase.auth().signOut();
  }

  render(){
    const currentUser = this.props.user === null ?
    "Guest" :
    this.props.user.displayName

    return(
      <div>
      <span>Logged in as: {currentUser}</span>
      <button onClick={(e)=> this.handleSignIn(e)}>
        Sign-in
      </button>
      <button onClick={(e)=> this.handleSignOut(e)}>
        Sign-out
      </button>

      </div>
```

```
    );
  }
}

export default User;
```

24. Create a `setUser` method in `App` component, passing the method to `User` component as a prop:

```
import React, { Component } from 'react';
import './App.css';
import * as firebase from 'firebase';
import RoomList from './components/RoomList';
import MessageList from './components/MessageList';
import User from './components/User';

  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyCdZA6va-NxMgpnM41vrOnIRfzKV8epg4g",
    authDomain: "bloc-chat-react-a6744.firebaseapp.com",
    databaseURL: "https://bloc-chat-react-a6744.firebaseio
.com",
    projectId: "bloc-chat-react-a6744",
    storageBucket: "bloc-chat-react-a6744.appspot.com",
    messagingSenderId: "503080022258"
  };
  firebase.initializeApp(config);
```

```
class App extends Component {
  constructor(props){
    super(props);
    this.state={
      activeRoom: [],
      activeUser: null
    };
  }
//end of constructor method

  handleRoomClick(room){
      this.setState({activeRoom: room});
  }

  setUser(user){
    this.setState({activeUser: user});
  }

  render() {
    return (
      <div className='App'>

      <section className='authorization'>
      <User
      firebase={firebase}
      setUser={(user)=> this.setUser(user)}
```

```
        user={this.state.activeUser}
        />
        </section>

        <section className='chatrooms'>
          <RoomList
          firebase={firebase}
          currentRoom={this.state.activeRoom}
          handleRoomClick={(room)=> this.handleRoomClick(roo
m)}
          />
        </section>

        <section className='messages'>
          <MessageList
          firebase={firebase}
          currentRoom={this.state.activeRoom}
          />
        </section>

        </div>
    );
  }
}

export default App;
```

25. Add a form for creating a new message to the `MessageList` component. Use the `.push()` method. Also, pass the active user from `App` to `MessageList` as a prop so it can be used to populate the `username` property of the message being constructed:

```
import React, {Component} from 'react';

class MessageList extends Component {
  constructor(props){
    super(props);
    this.state = { messages: [], newMessage:'' };
    this.messagesRef = this.props.firebase.database().ref(
'messages');
  }

  componentDidMount() {
        this.messagesRef.on('child_added', snapshot => {
            const message = snapshot.val();
            message.key = snapshot.key;
            this.setState({ messages: this.state.messages.
concat( message ) })
        });
    }

    handleNewMessage(e){
      this.setState({newMessage: e.target.value});
    }
```

```
    createMessage(e){
      e.preventDefault();
      if(!this.state.newMessage){return}
      const currentUser = this.props.user === null ? "Gues
t" : this.props.user.displayName;
      const newestMessage = {
        content: this.state.newMessage,
        roomId: this.props.currentRoom.key,
        sentAt: this.props.firebase.database.ServerValue.T
IMESTAMP,
        username: currentUser
      };
      this.messagesRef.push(newestMessage);
      this.setState({newMessage: ''});
    }

    render() {
      return (
        <div>
          <section className="messages">
            <h2>{(this.props.currentRoom.name)}</h2>
            <h2>{(this.props.currentRoom.key)}</h2>
            <nav>
              {
                this.state.messages
                .filter(message => this.props.currentRoom.
key === message.roomId)
```

```
                .map((message,index) =>
                  <div
                  key={index}
                  >
                  <div>{message.content}</div>
                  <div>{message.username}</div>
                  <div>{message.sentAt}</div>
                  </div>
                )
              }
            </nav>
          </section>
          <section>
          <form onSubmit={(e)=> this.createMessage(e)}>
            <label htmlFor='newMessageText'>Create a new m
essage</label>
            <textarea
            id ='newMessageText'
            type ='text'
            placeholder ='enter new message here...'
            value = {this.state.newMessage}
            onChange={(e)=> this.handleNewMessage(e)}
            rows='5'
            cols='50'
            >
            </textarea>
            <input
            type = 'submit'
```

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

export default MessageList;
```

```
import React, { Component } from 'react';
import './App.css';
import * as firebase from 'firebase';
import RoomList from './components/RoomList';
import MessageList from './components/MessageList';
import User from './components/User';

  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyCdZA6va-NxMgpnM41vrOnIRfzKV8epg4g",
    authDomain: "bloc-chat-react-a6744.firebaseapp.com",
    databaseURL: "https://bloc-chat-react-a6744.firebaseio
.com",
    projectId: "bloc-chat-react-a6744",
    storageBucket: "bloc-chat-react-a6744.appspot.com",
    messagingSenderId: "503080022258"
  };
```

```
  firebase.initializeApp(config);



class App extends Component {
  constructor(props){
    super(props);
    this.state={
      activeRoom: [],
      activeUser: null
    };
  }
//end of constructor method

  handleRoomClick(room){
      this.setState({activeRoom: room});
  }

  setUser(user){
    this.setState({activeUser: user});
  }

  render() {
    return (
      <div className='App'>

      <section className='authorization'>
      <User
```

```
        firebase={firebase}
        setUser={(user)=> this.setUser(user)}
        user={this.state.activeUser}
        />
        </section>

        <section className='chatrooms'>
          <RoomList
          firebase={firebase}
          currentRoom={this.state.activeRoom}
          handleRoomClick={(room)=> this.handleRoomClick(roo
m)}
          />
        </section>

        <section className='messages'>
          <MessageList
          firebase={firebase}
          currentRoom={this.state.activeRoom}
          user={this.state.activeUser}
          />
        </section>

        </div>
    );
  }
}
```

```
export default App;
```