Frontend Setup (React.js)

1.1. Install Dependencies

In your React project directory, install the necessary dependencies:

```
npm install agora-rtc-sdk-ng recordrtc downloadjs socket.io-client @material-ui/core
```

1.2. Configure Agora API

Create a setup file to configure the Agora API:

```
export const APP_ID = 'YOUR_AGORA_APP_ID'; // Replace with your Agora App ID
export const TOKEN = 'YOUR_AGORA_TOKEN'; // Replace with your Agora Token
```

1.3. Access Media Streams

```
import React, { useEffect, useRef, useState } from 'react';
import { createClient, createMicrophoneAndCameraTracks, ClientRole } from 'agora-rtc-sdk-ng';
import { APP_ID, TOKEN } from '../agoraConfig';

const VideoCall = ({ room, username, socket }) => {
  const [tracks, setTracks] = useState({ localAudioTrack: null, localVideoTrack: null });
  const [remoteUsers, setRemoteUsers] = useState([]);
  const client = useRef(null);
  const localVideoRef = useRef(null);
  const remoteVideoRefs = useRef({});

  useEffect(() => {
    const initAgora = async () => {
      client.current = createClient({ mode: 'rtc', codec: 'vp8' });

      client.current.on('user-published', async (user, mediaType) => {
        await client.current.subscribe(user, mediaType);
        if (mediaType === 'video') {
          remoteVideoRefs.current[user.uid] = user.videoTrack;
          setRemoteUsers([...remoteUsers, user]);
        }
      });
    client.current.on('user-unpublished', (user) => {
      const { [user.uid]: removedUser, ...rest } = remoteVideoRefs.current;
      setRemoteUsers(remoteUsers.filter(u => u.uid !== user.uid));
      remoteVideoRefs.current = rest;
    });

    await client.current.join(APP_ID, room, TOKEN, username);
    const { cameraTrack, microphoneTrack } = await createMicrophoneAndCameraTracks();
    setTracks({ localAudioTrack: microphoneTrack, localVideoTrack: cameraTrack });
    localVideoRef.current.srcObject = cameraTrack;
    await client.current.publish([cameraTrack, microphoneTrack]);
  };

  initAgora();

  return () => {
    tracks.localAudioTrack?.close();
    tracks.localVideoTrack?.close();
    client.current.leave();
  };
}, [room, username]);

return (
  <div>
    <video ref={localVideoRef} autoPlay playsInline style={{ width: '300px' }} />
    {remoteUsers.map(user => (
      <video
        key={user.uid}
        ref={(ref) => {
          if (ref) {
              user.videoTrack.play(ref);
            }
          }}
          autoPlay
          playsInline
          style={{ width: '300px' }}
        />
      ))}
    </div>
  );
};

export default VideoCall;
```

1.4. Add Meet Controls

```
import React from 'react';
import { Button } from '@material-ui/core';

const MeetControls = ({ tracks, setTracks, socket }) => {
  const toggleAudio = () => {
    const { localAudioTrack } = tracks;
    localAudioTrack.setEnabled(!localAudioTrack.enabled);
  };

  const toggleVideo = () => {
    const { localVideoTrack } = tracks;
    localVideoTrack.setEnabled(!localVideoTrack.enabled);
  };

  const leaveRoom = () => {
    socket.emit('leaveRoom');
    window.location.reload();
  };

  return (
    <div>
      <Button onClick={toggleAudio}>Toggle Audio</Button>
      <Button onClick={toggleVideo}>Toggle Video</Button>
      <Button onClick={leaveRoom}>Leave Room</Button>
    </div>
  );
};

export default MeetControls;
```

1.5. Add Screen Sharing

```
import React, { useState } from 'react';
import { Button } from '@material-ui/core';
import { createClient, createMicrophoneAndCameraTracks } from 'agora-rtc-sdk-ng';
import { APP_ID, TOKEN } from '../agoraConfig';

const ScreenShare = ({ client, room, username }) => {
  const [sharing, setSharing] = useState(false);

  const startScreenShare = async () => {
    const screenTracks = await AgoraRTC.createScreenTracks();
    await client.publish(screenTracks);
    setSharing(true);
  };

  const stopScreenShare = async () => {
    const tracks = await client.getLocalTracks();
    tracks.forEach(track => track.stop());
    const { cameraTrack, microphoneTrack } = await createMicrophoneAndCameraTracks();
    await client.publish([cameraTrack, microphoneTrack]);
    setSharing(false);
  };

  return (
    <div>
      {sharing ? (
        <Button onClick={stopScreenShare}>Stop Screen Share</Button>
      ) : (
        <Button onClick={startScreenShare}>Start Screen Share</Button>
      )}
    </div>
  );
};

export default ScreenShare;
```

1.6. Add Recording Functionality

```
import React, { useRef } from 'react';
import RecordRTC from 'recordrtc';
import { Button } from '@material-ui/core';
import { saveAs } from 'file-saver';

const Recording = ({ tracks }) => {
  const recorderRef = useRef(null);

  const startRecording = () => {
    const stream = new MediaStream([
      tracks.localAudioTrack.getMediaStreamTrack(),
      tracks.localVideoTrack.getMediaStreamTrack(),
    ]);
    recorderRef.current = new RecordRTC(stream, { type: 'video' });
    recorderRef.current.startRecording();
  };

  const stopRecording = () => {
    recorderRef.current.stopRecording(() => {
      const blob = recorderRef.current.getBlob();
      saveAs(blob, 'recording.mp4');
    });
  };

  return (
    <div>
      <Button onClick={startRecording}>Start Recording</Button>
      <Button onClick={stopRecording}>Stop Recording</Button>
    </div>
  );
};

export default Recording;
```

1.7. Add In-Meet Chat Facility

```
import React, { useState, useEffect } from 'react';
import { TextField, Button, List, ListItem } from '@material-ui/core';
import io from 'socket.io-client';

const Chat = ({ socket }) => {
  const [message, setMessage] = useState('');
  const [messages, setMessages] = useState([]);

  useEffect(() => {
    socket.on('chatMessage', (msg) => {
      setMessages((msgs) => [...msgs, msg]);
    });
  }, [socket]);

  const sendMessage = () => {
    if (message) {
      socket.emit('chatMessage', message);
      setMessage('');
    }
  };

  return (
    <div>
      <List>
        {messages.map((msg, index) => (
          <ListItem key={index}>{msg}</ListItem>
        ))}
      </List>
      <TextField
        value={message}
        onChange={(e) => setMessage(e.target.value)}
        variant="outlined"
        fullWidth
      />
      <Button onClick={sendMessage} variant="contained" color="primary">
        Send
      </Button>
    </div>
  );
};

export default Chat;
```

1.8. User Profile Section

```
import React, { useState } from 'react';
import { TextField, Button, Typography } from '@material-ui/core';
import axios from 'axios';

const Profile = () => {
  const [userData, setUserData] = useState({ username: '', email: '' });

  const fetchProfile = async () => {
    // Replace with your API call to fetch user data
    const result = await axios.get('/api/profile');
    setUserData(result.data);
  };

  const updateProfile = async () => {
    // Replace with your API call to update user data
    await axios.put('/api/profile', userData);
  };

  return (
    <div>
      <Typography variant="h4">Profile</Typography>
      <TextField
        label="Username"
        value={userData.username}
        onChange={(e) => setUserData({ ...userData, username: e.target.value })}
        variant="outlined"
        fullWidth
      />
      <TextField
        label="Email"
        value={userData.email}
        onChange={(e) => setUserData({ ...userData, email: e.target.value })}
        variant="outlined"
        fullWidth
      />
      <Button onClick={updateProfile} variant="contained" color="primary">
        Update Profile
      </Button>
    </div>
  );
};

export default Profile;
```