**PRACTICAL: 11**

**AIM:**

**Build an exact similar clone of any social media web application like Instagram/LinkedIn/WhatsApp/Twitter etc. Your web application should contain following things:**

**1. Build a Login/Signup.**

**2. Build the sidebar.**

**3. Chatting can be done between 2 users.**

**4. Build the rows to display the recent chats.**

**5. Build the sidebar.**

**Finally, host your web application on Google Firebase and push the code to your GitHub Repository to enhance your GitHub Profile.**

**THEORY:**

React Components

Function Component: The app is built as a function component (App), which is a reusable, self-contained piece of UI in React.

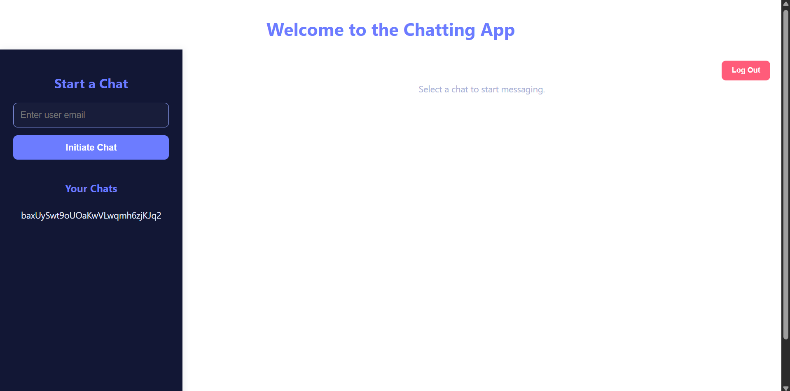
JSX (JavaScript XML)

JSX Syntax: The UI is described using JSX, which allows you to write HTML-like code within JavaScript. JSX makes it easy to visualize and structure the component’s output.

**CODE:**

|  |
| --- |
| <!doctype html>  <html lang="en">  <head>    <meta charset="utf-8" />    <meta name="viewport" content="width=device-width, initial-scale=1" />    <title>Chatter — Minimal Social Chat</title>    <link rel="preconnect" href="https://fonts.googleapis.com">    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&display=swap" rel="stylesheet">    <style>      :root{        --bg:#0b1020; --panel:#121735; --muted:#93a5fd; --text:#e8eeff; --accent:#6c7cff; --accent-2:#16db65; --danger:#ff5c7a;      }      \*{box-sizing:border-box}      html,body{height:100%}      body{margin:0;background:linear-gradient(120deg,#0b1020,#141a3f);color:var(--text);font-family:Inter,system-ui,Segoe UI,Roboto,Helvetica,Arial,sans-serif}      .app{display:grid;grid-template-columns:280px 1fr;min-height:100vh}      .sidebar{background:rgba(18,23,53,.7);backdrop-filter: blur(8px);border-right:1px solid rgba(255,255,255,.08);padding:16px;display:flex;flex-direction:column;gap:16px}      .logo{display:flex;align-items:center;gap:10px;font-weight:700}      .logo .dot{width:10px;height:10px;border-radius:50%;background:var(--accent-2);box-shadow:0 0 18px var(--accent-2)}      .card{background:rgba(255,255,255,.04);border:1px solid rgba(255,255,255,.08);border-radius:16px}      .user-card{display:flex;align-items:center;gap:10px;padding:12px}      .user-card img{width:36px;height:36px;border-radius:50%}      .btn, button, input, textarea{font-family:inherit}      .btn{padding:10px 12px;border-radius:10px;border:1px solid rgba(255,255,255,.15);background:rgba(255,255,255,.05);color:var(--text);cursor:pointer}      .btn.primary{background:var(--accent);border-color:transparent}      .btn.ghost{background:transparent;border-color:rgba(255,255,255,.1)}      .btn.full{width:100%}      .muted{color:#a8b0d6;font-size:.9rem}      .search{display:flex;gap:8px;padding:10px}      .search input{flex:1;padding:10px;border-radius:10px;border:1px solid rgba(255,255,255,.12);background:rgba(0,0,0,.2);color:var(--text)}      .nav{display:flex;flex-direction:column;gap:8px;padding:10px}      .nav .item{display:flex;align-items:center;gap:10px;padding:10px;border-radius:10px;cursor:pointer}      .nav .item.active,.nav .item:hover{background:rgba(255,255,255,.08)}      .chats{display:flex;flex-direction:column;gap:6px;padding:8px;overflow:auto}      .chat-row{display:grid;grid-template-columns:46px 1fr auto;gap:10px;padding:10px;border-radius:12px;cursor:pointer}      .chat-row:hover{background:rgba(255,255,255,.06)}      .chat-row img{width:46px;height:46px;border-radius:50%}      .chat-row .name{font-weight:600}      .chat-row .last{color:#b7bde3;font-size:.92rem;white-space:nowrap;overflow:hidden;text-overflow:ellipsis}      .main{display:flex;flex-direction:column;height:100vh}      .topbar{display:flex;align-items:center;justify-content:space-between;padding:12px 16px;border-bottom:1px solid rgba(255,255,255,.08);background:rgba(18,23,53,.6);backdrop-filter: blur(8px)}      .topbar .title{display:flex;align-items:center;gap:10px}      .content{display:grid;grid-template-rows:1fr auto;gap:0;height:calc(100vh - 56px)}      .messages{padding:14px;overflow:auto;display:flex;flex-direction:column;gap:8px}      .bubble{max-width:70%;padding:10px 12px;border-radius:14px;line-height:1.35}      .me{align-self:flex-end;background:linear-gradient(135deg,var(--accent),#8e99ff)}      .you{align-self:flex-start;background:rgba(255,255,255,.08)}      .composer{display:grid;grid-template-columns:1fr auto;gap:10px;padding:12px;border-top:1px solid rgba(255,255,255,.08);background:rgba(0,0,0,.15)}      .composer textarea{resize:none;height:50px;padding:10px;border-radius:10px;border:1px solid rgba(255,255,255,.12);background:rgba(0,0,0,.2);color:var(--text)}      .auth{max-width:420px;margin:8vh auto;padding:20px;}      .auth .group{display:flex;flex-direction:column;gap:6px;margin:8px 0}      .auth input{padding:12px;border-radius:10px;border:1px solid rgba(255,255,255,.12);background:rgba(0,0,0,.2);color:var(--text)}      .switch{margin-top:6px;font-size:.95rem}      .empty{display:grid;place-items:center;height:100%;color:#aeb7e9}      .pill{padding:6px 10px;border-radius:999px;background:rgba(255,255,255,.08);border:1px dashed rgba(255,255,255,.18)}      @media(max-width:980px){.app{grid-template-columns:1fr}.sidebar{position:fixed;z-index:10;inset:0 30% 0 0;transform:translateX(-100%);transition:.25s}.sidebar.open{transform:none} .topbar .menu{display:inline-block} .topbar .menu button{font-size:20px} .topbar .title{gap:8px}}      @media(min-width:981px){.topbar .menu{display:none}}    </style>  </head>  <body>  <div id="authView" class="auth card" hidden>    <div class="logo" style="margin-bottom:12px"><span class="dot"></span> <span>Chatter</span></div>    <h2 id="authTitle">Create account</h2>    <div class="group"><label>Name</label><input id="nameInput" placeholder="Ada Lovelace"/></div>    <div class="group"><label>Email</label><input id="emailInput" type="email" placeholder="ada@chatter.app"/></div>    <div class="group"><label>Password</label><input id="passwordInput" type="password" placeholder="••••••••"/></div>    <div class="group"><button class="btn primary full" id="authSubmit">Sign up</button></div>    <div class="group"><button class="btn full" id="googleBtn">Continue with Google</button></div>    <div class="switch" id="authSwitch">Already have an account? <a href="#" id="toLogin">Sign in</a></div>    <p class="muted" style="margin-top:8px">Fill your Firebase config in the script to run locally.</p>  </div>  <div id="appView" class="app" hidden>    <aside class="sidebar" id="sidebar">      <div class="logo"><span class="dot"></span><span>Chatter</span></div>      <div class="card user-card">        <img id="meAvatar" src="https://api.dicebear.com/9.x/thumbs/svg?seed=me" alt="me">        <div>          <div id="meName" style="font-weight:600">—</div>          <div id="meEmail" class="muted">—</div>        </div>      </div>      <div class="card search">        <input id="userSearch" placeholder="Search users by name/email"/>        <button class="btn" id="inviteBtn">+</button>      </div>      <div class="nav card">        <div class="item active" id="navChats">💬 Chats</div>        <div class="item" id="navPeople">👥 People</div>        <div class="item" id="navProfile">🙍 Profile</div>        <div class="item" id="navLogout">🚪 Logout</div>      </div>      <div class="card" style="padding:8px;">        <div class="muted" style="padding:8px 8px 0">Recent chats</div>        <div class="chats" id="recentList"></div>      </div>      <div class="muted" style="padding:8px;text-align:center">Built with Firebase</div>    </aside>    <main class="main">      <div class="topbar">        <div class="title">          <div class="menu"><button class="btn ghost" id="toggleSidebar">☰</button></div>          <div><strong id="chatTitle">Welcome</strong><div id="chatSub" class="muted" style="font-size:.9rem">Select a conversation or find people to start chatting.</div></div>        </div>      </div>      <div class="content">        <div class="messages" id="messages"></div>        <div class="composer" id="composer" hidden>          <textarea id="messageInput" placeholder="Type a message…"></textarea>          <button class="btn primary" id="sendBtn">Send</button>        </div>        <div class="empty" id="emptyState"><div class="pill">No chat selected</div></div>      </div>    </main>  </div>  <script type="module">    // === 1) Fill YOUR Firebase config here ===    // === 2) Firebase SDKs (v10 modular) ===    import { initializeApp } from 'https://www.gstatic.com/firebasejs/10.12.2/firebase-app.js';    import { getAuth, onAuthStateChanged, signInWithEmailAndPassword, createUserWithEmailAndPassword, signOut, updateProfile, GoogleAuthProvider, signInWithPopup } from 'https://www.gstatic.com/firebasejs/10.12.2/firebase-auth.js';    import { getFirestore, doc, setDoc, getDoc, addDoc, collection, query, where, getDocs, serverTimestamp, onSnapshot, orderBy, updateDoc } from 'https://www.gstatic.com/firebasejs/10.12.2/firebase-firestore.js';    const app = initializeApp(firebaseConfig);    const auth = getAuth(app);    const db = getFirestore(app);    // DOM refs    const authView = document.getElementById('authView');    const appView = document.getElementById('appView');    const authTitle = document.getElementById('authTitle');    const nameInput = document.getElementById('nameInput');    const emailInput = document.getElementById('emailInput');    const passwordInput = document.getElementById('passwordInput');    const authSubmit = document.getElementById('authSubmit');    const toLogin = document.getElementById('toLogin');    const googleBtn = document.getElementById('googleBtn');    const meName = document.getElementById('meName');    const meEmail = document.getElementById('meEmail');    const meAvatar = document.getElementById('meAvatar');    const recentList = document.getElementById('recentList');    const userSearch = document.getElementById('userSearch');    const inviteBtn = document.getElementById('inviteBtn');    const navLogout = document.getElementById('navLogout');    const chatTitle = document.getElementById('chatTitle');    const chatSub = document.getElementById('chatSub');    const messagesEl = document.getElementById('messages');    const composer = document.getElementById('composer');    const emptyState = document.getElementById('emptyState');    const messageInput = document.getElementById('messageInput');    const sendBtn = document.getElementById('sendBtn');    const toggleSidebar = document.getElementById('toggleSidebar');    const sidebar = document.getElementById('sidebar');    let isSignup = true;    let activeChatId = null;    let messagesUnsub = null; // real-time listener cleanup    const provider = new GoogleAuthProvider();    function setAuthMode(signup){      isSignup = signup;      authTitle.textContent = signup ? 'Create account' : 'Welcome back';      authSubmit.textContent = signup ? 'Sign up' : 'Sign in';      nameInput.parentElement.style.display = signup ? 'flex' : 'none';      document.getElementById('authSwitch').innerHTML = signup ? 'Already have an account? <a href="#" id="toLogin">Sign in</a>' : "Don't have an account? <a href='#' id='toSignup'>Create one'</a>";      document.getElementById('toLogin')?.addEventListener('click', e=>{e.preventDefault(); setAuthMode(false)});      document.getElementById('toSignup')?.addEventListener('click', e=>{e.preventDefault(); setAuthMode(true)});    }    setAuthMode(true);    googleBtn.addEventListener('click', async ()=>{      try{        const cred = await signInWithPopup(auth, provider);        await ensureUserDoc(cred.user);      }catch(e){ alert(e.message) }    });    authSubmit.addEventListener('click', async ()=>{      const email = emailInput.value.trim();      const pass = passwordInput.value.trim();      try{        if(isSignup){          const name = nameInput.value.trim() || email.split('@')[0];          const cred = await createUserWithEmailAndPassword(auth, email, pass);          await updateProfile(cred.user, { displayName: name, photoURL: `https://api.dicebear.com/9.x/thumbs/svg?seed=${encodeURIComponent(name)}` });          await ensureUserDoc(cred.user);        }else{          const cred = await signInWithEmailAndPassword(auth, email, pass);          await ensureUserDoc(cred.user);        }      }catch(e){ alert(e.message) }    });    navLogout.addEventListener('click', ()=> signOut(auth));    toggleSidebar.addEventListener('click', ()=> sidebar.classList.toggle('open'));    onAuthStateChanged(auth, async (user)=>{      if(user){        authView.hidden = true; appView.hidden = false;        meName.textContent = user.displayName || 'Anonymous';        meEmail.textContent = user.email || '';        meAvatar.src = user.photoURL || `https://api.dicebear.com/9.x/thumbs/svg?seed=${encodeURIComponent(user.uid)}`;        loadRecentChats(user.uid);      }else{        appView.hidden = true; authView.hidden = false; setAuthMode(false);      }    });    async function ensureUserDoc(user){      const ref = doc(db, 'users', user.uid);      const snap = await getDoc(ref);      if(!snap.exists()){        await setDoc(ref, { uid:user.uid, name:user.displayName||'Anonymous', email:user.email, photoURL:user.photoURL||'', createdAt: serverTimestamp() });      }    }    // Search & people listing    userSearch.addEventListener('input', debounce(async (e)=>{      const term = e.target.value.trim().toLowerCase();      const me = auth.currentUser;      if(!me) return;      const usersRef = collection(db, 'users');      let q;      if(term){        // Simple client filter because Firestore doesn't do contains for strings without indexes; keep demo-friendly        const qs = await getDocs(usersRef);        const items = qs.docs.map(d=>d.data()).filter(u=>u.uid!==me.uid && ((u.name||'').toLowerCase().includes(term) || (u.email||'').toLowerCase().includes(term)));        renderPeople(items);      }else{        const qs = await getDocs(usersRef);        const items = qs.docs.map(d=>d.data()).filter(u=>u.uid!==me.uid).slice(0,50);        renderPeople(items);      }    }, 300));    function renderPeople(users){      const list = document.createElement('div');      list.style.padding = '8px';      list.style.display = 'flex';      list.style.flexDirection = 'column';      list.style.gap = '6px';      users.forEach(u=>{        const row = document.createElement('div');        row.className = 'chat-row';        row.innerHTML = `<img src="${u.photoURL || `https://api.dicebear.com/9.x/thumbs/svg?seed=${encodeURIComponent(u.name||u.uid)}`}"><div><div class='name'>${escapeHtml(u.name||'User')}</div><div class='last'>${escapeHtml(u.email||'')}</div></div><div><button class='btn'>Chat</button></div>`;        row.querySelector('button').addEventListener('click', ()=> startChatWith(u.uid));        recentList.prepend(row);      })    }    // Recent chats (userChats subcollection per user for quick listing)    function loadRecentChats(uid){      const ucRef = collection(db, 'userChats', uid, 'items');      const qy = query(ucRef, orderBy('updatedAt','desc'));      onSnapshot(qy, (qs)=>{        recentList.innerHTML = '';        qs.forEach(docu=>{          const c = docu.data();          const row = document.createElement('div');          row.className = 'chat-row';          row.dataset.chatId = c.chatId;          row.innerHTML = `            <img src="${c.otherPhoto || `https://api.dicebear.com/9.x/thumbs/svg?seed=${encodeURIComponent(c.otherUid)}`}" alt="pfp">            <div><div class="name">${escapeHtml(c.otherName||'User')}</div><div class="last">${escapeHtml(c.lastMessage||'Say hi 👋')}</div></div>            <div class="muted">${timeAgo(c.updatedAt?.toDate?.()||new Date())}</div>`;          row.addEventListener('click', ()=> openChat(c.chatId, {uid:c.otherUid, name:c.otherName, photo:c.otherPhoto}));          recentList.appendChild(row);        });      });    }    function composeChatId(a,b){ return a < b ? `${a}\_${b}` : `${b}\_${a}` }    async function startChatWith(otherUid){      const me = auth.currentUser;      if(!me || me.uid===otherUid) return;      const chatId = composeChatId(me.uid, otherUid);      const chatRef = doc(db, 'chats', chatId);      const chatSnap = await getDoc(chatRef);      if(!chatSnap.exists()){        // Create chat doc        await setDoc(chatRef, { chatId, participants:[me.uid, otherUid], createdAt: serverTimestamp(), updatedAt: serverTimestamp(), lastMessage:'' });      }      // Mirror in each user's userChats for fast recent list      const other = await getDoc(doc(db,'users',otherUid));      const meUser = await getDoc(doc(db,'users',me.uid));      await setDoc(doc(db,'userChats',me.uid,'items',chatId), {        chatId, otherUid, otherName: other.data()?.name||'User', otherPhoto: other.data()?.photoURL||'', lastMessage:'', updatedAt: serverTimestamp()      }, { merge:true });      await setDoc(doc(db,'userChats',otherUid,'items',chatId), {        chatId, otherUid: me.uid, otherName: meUser.data()?.name||'User', otherPhoto: meUser.data()?.photoURL||'', lastMessage:'', updatedAt: serverTimestamp()      }, { merge:true });      openChat(chatId, {uid: otherUid, name: other.data()?.name, photo: other.data()?.photoURL});    }    function openChat(chatId, other){      activeChatId = chatId;      chatTitle.textContent = other?.name || 'Chat';      chatSub.textContent = other?.email || '';      emptyState.hidden = true; composer.hidden = false; messagesEl.innerHTML = '';      sidebar.classList.remove('open');      if(messagesUnsub) messagesUnsub();      const msgsRef = collection(db,'chats',chatId,'messages');      const qy = query(msgsRef, orderBy('createdAt','asc'));      messagesUnsub = onSnapshot(qy, (qs)=>{        messagesEl.innerHTML = '';        qs.forEach(d=>{          const m = d.data();          const isMe = m.senderId === auth.currentUser?.uid;          const div = document.createElement('div');          div.className = `bubble ${isMe? 'me':'you'}`;          div.textContent = m.text;          messagesEl.appendChild(div);        });        messagesEl.scrollTop = messagesEl.scrollHeight;      });    }    sendBtn.addEventListener('click', sendMessage);    messageInput.addEventListener('keydown', (e)=>{      if(e.key==='Enter' && !e.shiftKey){ e.preventDefault(); sendMessage(); }    });    async function sendMessage(){      const text = messageInput.value.trim();      if(!text || !activeChatId) return;      const me = auth.currentUser; if(!me) return;      messageInput.value = '';      const msgsRef = collection(db,'chats',activeChatId,'messages');      await addDoc(msgsRef, { text, senderId: me.uid, createdAt: serverTimestamp() });      // update last message mirrors      const chatRef = doc(db,'chats',activeChatId);      await updateDoc(chatRef, { lastMessage:text, updatedAt: serverTimestamp() });      // Determine other userId from chatId      const [a,b] = activeChatId.split('\_');      const otherUid = me.uid===a? b:a;      await updateDoc(doc(db,'userChats',me.uid,'items',activeChatId), { lastMessage:text, updatedAt: serverTimestamp() });      await updateDoc(doc(db,'userChats',otherUid,'items',activeChatId), { lastMessage:text, updatedAt: serverTimestamp() });    }    // Helpers    function debounce(fn, ms){ let t; return (...args)=>{ clearTimeout(t); t=setTimeout(()=>fn(...args),ms) } }    function escapeHtml(s){ return s?.replace(/[&<>"']/g, c=>({"&":"&amp;","<":"&lt;",">":"&gt;","\"":"&quot;","'":"&#39;"}[c])) }    function timeAgo(date){ const d = (typeof date==='number')? new Date(date):date; const diff = Math.floor((Date.now()-d.getTime())/1000); if(diff<60) return diff+"s"; if(diff<3600) return Math.floor(diff/60)+"m"; if(diff<86400) return Math.floor(diff/3600)+"h"; return Math.floor(diff/86400)+"d" }    // Invite button simply opens mailto with this page URL (when hosted)    inviteBtn.addEventListener('click', ()=>{      const url = location.href;      location.href = `mailto:?subject=Join me on Chatter&body=Let’s chat here: ${url}`;    });  </script>  </body>  </html> |

**OUTPUT:**



**LATEST APPLICATIONS:**

In today’s time, HTML, CSS, and JavaScript serve as great stepping stones for anyone starting their journey in web development. Many simple and static websites still rely on this core tech stack to reduce complexity and lower hosting costs. However, modern development has seen the rise of frameworks like React, which allows developers to build dynamic, scalable, and interactive user interfaces. React is widely used in real-world applications, from social media platforms to e-commerce websites, making it a valuable skill for any aspiring web developer.

**LEARNING OUTCOME:**

By performing this practical XI I got a thorough understanding of basics and firebase. And how the frontend and backend connect in a real website.

**REFERENCES:**

1. React: react.dev
2. Html: <https://www.freecodecamp.org/learn/2022/responsive-web-design/learn-html-by-building-a-cat-photo-app/step-66>
3. Css: <https://www.w3schools.com/css/>