MOSARSY-MVP

```
import React, { useState, useEffect } from "react";
import { db, auth, functions } from "./firebaseConfig";
import {
  collection,
  getDocs,
  doc,
  query,
  where,
  orderBy,
  updateDoc,
  addDoc,
  Timestamp,
  setDoc,
  getDoc
} from "firebase/firestore";
import {


function MemberDashboard() {
  const [user, setUser] = useState(null);
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [isRegistering, setIsRegistering] = useState(false);
  const [error, setError] = useState("");
  const [displayName, setDisplayName] = useState("");
  const [profileSaved, setProfileSaved] = useState(false);
  const [payoutMessages, setPayoutMessages] = useState([]);
  const [acknowledged, setAcknowledged] = useState([]);
  const [inviteEmail, setInviteEmail] = useState("");
  const [invited, setInvited] = useState("");
  const [managedRing, setManagedRing] = useState(null);
  const [members, setMembers] = useState([]);
  const [rings, setRings] = useState([]); // New: List of available rings
  const [newRingName, setNewRingName] = useState(""); // For creating new ring
  const [selectedRing, setSelectedRing] = useState("all");
  const [tab, setTab] = useState("dashboard"); // Tabs for navigation
  const [savingsData, setSavingsData] = useState({}); // Simulated savings/loan data

  useEffect(() => {
    const unsubscribe = onAuthStateChanged(auth, async (u) => {
```

```
    setUser(u);
    if (u) {
      const profileRef = doc(db, "users", u.uid);
      const profileSnap = await getDoc(profileRef);
      if (profileSnap.exists()) {
        const data = profileSnap.data();
        if (data.name) setDisplayName(data.name);
      }
      loadRings(); // Load available rings
      loadSavingsData(); // Simulate fintech calcs
    }
  });
  return () => unsubscribe();
}, []);

// Load available Rings of Trust
const loadRings = async () => {
  if (!user) return;
  const ringSnapshot = await getDocs(collection(db, "rings"));
  const userRings = [];
  ringSnapshot.forEach((docSnap) => {
    const data = docSnap.data();
    const isMember = data.memberHistory?.some((m) => m.email === user.email);
    if (isMember) {
      userRings.push({ id: docSnap.id, ...data });
      if (data.memberHistory.some((m) => m.email === user.email && m.role === "admin")) {
        setManagedRing({ id: docSnap.id, name: data.ringName });
        setMembers(data.memberHistory);
      }
    }
  });
  setRings(userRings);
};

// Simulate savings/loan data based on your 12-member example
const loadSavingsData = () => {
  // Placeholder calcs: Adapt from doc examples (e.g., $100/month, 12 members, 4.59% return)
  setSavingsData({
    investment: 1200, // $100 * 12 months
    loanReceived: 1100, // Position 1 example
    monthlyRepay: 116.92,
```

```
    profit: 55.01, // End-of-cycle profit
    rpf: 220 // 20% of loan
  });
};

const handleLogin = async (e) => {
  e.preventDefault();
  setError("");
  try {
    await signInWithEmailAndPassword(auth, email, password);
  } catch (err) {
    setError("Invalid login credentials");
  }
};

const handleRegister = async (e) => {
  e.preventDefault();
  setError("");
  try {
    const cred = await createUserWithEmailAndPassword(auth, email, password);
    await setDoc(doc(db, "users", cred.user.uid), {
      email: cred.user.email,
      createdAt: Timestamp.now()
    });
  } catch (err) {
    setError("Registration failed: " + err.message);
  }
};

const handleProfileSave = async () => {
  if (!user) return;
  await updateDoc(doc(db, "users", user.uid), {
    name: displayName,
    updatedAt: Timestamp.now()
  });
  setProfileSaved(true);
  setTimeout(() => setProfileSaved(false), 2000);
};

const handleInvite = async () => {
  if (!inviteEmail) return;
```

```
    try {
      const sendEmail = httpsCallable(functions, "sendEmailReminder"); // Assume Cloud Function
setup
      await sendEmail({
        email: inviteEmail,
        subject: "You're Invited to Join MÖSARSY",
        message: `You've been invited by ${user.email} to join MÖSARSY. Register at [your-app-url]`
      });
      setInvited("Invitation sent to " + inviteEmail);
      setInviteEmail("");
    } catch (err) {
      setError("Failed to send invite");
    }
  };

  // Create new Ring of Trust
  const createRing = async () => {
    if (!newRingName || !user) return;
    const newRingRef = await addDoc(collection(db, "rings"), {
      ringName: newRingName,
      memberHistory: [{ email: user.email, role: "admin" }],
      createdAt: Timestamp.now()
    });
    setNewRingName("");
    loadRings(); // Refresh
  };

  // Join an existing ring (simplified: assume public rings for MVP)
  const joinRing = async (ringId) => {
    if (!user) return;
    const ringRef = doc(db, "rings", ringId);
    const ringSnap = await getDoc(ringRef);
    if (ringSnap.exists()) {
      const data = ringSnap.data();
      if (data.memberHistory.length < 14) { // Max 14 members
        await updateDoc(ringRef, {
          memberHistory: [...data.memberHistory, { email: user.email, role: "member" }]
        });
        loadRings();
      } else {
        setError("Ring is full (max 14 members)");
```

```
    }
  }
};

// Fetch payout messages (from second snippet)
useEffect(() => {
  const fetchMessages = async () => {
    if (!user) return;
    const allMessages = [];
    const acknowledgements = [];
    for (const ring of rings) {
      const msgSnap = await getDocs(query(
        collection(db, `rings/${ring.id}/messages`),
        orderBy("timestamp", "desc")
      ));
      msgSnap.forEach((doc) => {
        const msg = doc.data();
        allMessages.push({ ...msg, ringId: ring.id, ringName: ring.ringName, docId: doc.id });
        if (msg.acknowledged?.includes(user.email)) {
          acknowledgements.push(doc.id);
        }
      });
    }
    setPayoutMessages(allMessages);
    setAcknowledged(acknowledgements);
  };
  if (rings.length > 0) fetchMessages();
}, [rings, user]);

const handleAcknowledge = async (ringId, docId) => {
  const ref = doc(db, `rings/${ringId}/messages/${docId}`);
  const messageData = payoutMessages.find((m) => m.docId === docId);
  const updatedAck = [...(messageData.acknowledged || []), user.email];
  await updateDoc(ref, { acknowledged: updatedAck });
  setAcknowledged((prev) => [...prev, docId]);

  // Check if all acknowledged (notify admin)
  const ring = rings.find((r) => r.id === ringId);
  const memberEmails = ring?.memberHistory.map((m) => m.email) || [];
  if (memberEmails.every((email) => updatedAck.includes(email))) {
    await addDoc(collection(db, `rings/${ringId}/messages`), {
```

```
      sender: "System",
      content: `✅ All members acknowledged payout on ${new Date().toLocaleDateString()}`,
      timestamp: Timestamp.now(),
      acknowledged: []
    });
  }
};

// Promote/Demote/Remove members (admin only)
const promoteMember = async (email) => {
  const updated = members.map((m) => (m.email === email ? { ...m, role: "admin" } : m));
  await updateDoc(doc(db, "rings", managedRing.id), { memberHistory: updated });
  setMembers(updated);
};

const demoteMember = async (email) => {
  const updated = members.map((m) => (m.email === email ? { ...m, role: "member" } : m));
  await updateDoc(doc(db, "rings", managedRing.id), { memberHistory: updated });
  setMembers(updated);
};

const removeMember = async (email) => {
  const updated = members.filter((m) => m.email !== email);
  await updateDoc(doc(db, "rings", managedRing.id), { memberHistory: updated });
  setMembers(updated);
};

if (!user) {
  return (
    <div className="p-8 max-w-sm mx-auto mt-20 border rounded shadow bg-white">
      <h2 className="text-xl font-bold mb-4">{isRegistering ? "Register" : "Login"} to
MÖSARSY</h2>
      <form onSubmit={isRegistering ? handleRegister : handleLogin}>
        <input
          type="email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          placeholder="Email"
          className="w-full p-2 border mb-2 rounded"
          required
        />
```

```
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            placeholder="Password"
            className="w-full p-2 border mb-4 rounded"
            required
          />
          <button type="submit" className="w-full bg-blue-600 text-white py-2 rounded">
            {isRegistering ? "Create Account" : "Log In"}
          </button>
          {error && <p className="text-red-500 mt-2 text-sm">{error}</p>}
        </form>
        <p className="mt-4 text-sm text-center">
          {isRegistering ? "Already have an account?" : "Need an account?"}{" "}
          <button onClick={() => setIsRegistering(!isRegistering)} className="text-blue-600
underline">
            {isRegistering ? "Log In" : "Register"}
          </button>
        </p>
      </div>
    );
  }

  return (
    <div className="p-8 max-w-4xl mx-auto bg-white rounded shadow">
      <div className="flex justify-between items-center mb-6">
        <div>
          <h2 className="text-2xl font-bold">Welcome, {displayName || user.email}</h2>
          <p className="text-sm text-gray-600">Your fintech partner in community savings!</p>
        </div>
        <button onClick={() => signOut(auth)} className="text-sm text-red-500 underline">
          Log out
        </button>
      </div>

      <div className="flex space-x-4 mb-6">
        <button onClick={() => setTab("dashboard")} className={`px-4 py-2 rounded ${tab ===
"dashboard" ? "bg-blue-600 text-white" : "bg-gray-200"}`}>Dashboard</button>
        <button onClick={() => setTab("rings")} className={`px-4 py-2 rounded ${tab ===
"rings" ? "bg-blue-600 text-white" : "bg-gray-200"}`}>Rings</button>
```

```
      <button onClick={() => setTab("payouts")} className={`px-4 py-2 rounded ${tab === "payouts" ? "bg-blue-600 text-white" : "bg-gray-200"}`}>Payouts</button>
      <button onClick={() => setTab("profile")} className={`px-4 py-2 rounded ${tab === "profile" ? "bg-blue-600 text-white" : "bg-gray-200"}`}>Profile</button>
    </div>

    {tab === "dashboard" && (
     <div>
      <h3 className="text-lg font-semibold mb-2">Your Savings Overview</h3>
      <div className="grid grid-cols-2 gap-4">
       <div className="p-4 border rounded">
        <p>Investment: ${savingsData.investment}</p>
        <p>Loan Received: ${savingsData.loanReceived}</p>
       </div>
       <div className="p-4 border rounded">
        <p>Monthly Repay: ${savingsData.monthlyRepay.toFixed(2)}</p>
        <p>Profit Earned: ${savingsData.profit.toFixed(2)} (4.59% return)</p>
       </div>
       <div className="p-4 border rounded col-span-2">
        <p>Risk Provision Fund (RPF): ${savingsData.rpf} (20% protected)</p>
       </div>
      </div>
     </div>
    )}

    {tab === "rings" && (
     <div>
      <h3 className="text-lg font-semibold mb-2">Your Rings of Trust</h3>
      <ul className="list-disc ml-6 mb-4">
       {rings.map((ring) => (
        <li key={ring.id}>
         {ring.ringName} (Members: {ring.memberHistory.length}/14)
         {!ring.memberHistory.some((m) => m.email === user.email) && (
          <button onClick={() => joinRing(ring.id)} className="ml-2 text-green-600 underline">Join</button>
         )}
        </li>
       ))}
      </ul>
      {managedRing && (
       <div className="mb-4">
```

```
          <h4 className="font-semibold">Manage {managedRing.name}</h4>
          <ul className="list-disc ml-6">
            {members.map((m, idx) => (
              <li key={idx}>
                {m.email} — {m.role}
                {m.email !== user.email && (
                  <>
                    {m.role === "member" ? (
                      <button onClick={() => promoteMember(m.email)} className="ml-2 text-
green-600 underline">Promote</button>
                    ) : (
                      <button onClick={() => demoteMember(m.email)} className="ml-2 text-yellow-
600 underline">Demote</button>
                    )}
                    <button onClick={() => removeMember(m.email)} className="ml-2 text-red-600
underline">Remove</button>
                  </>
                )}
              </li>
            ))}
          </ul>
        </div>
      )}
      <div>
        <input
          type="text"
          value={newRingName}
          onChange={(e) => setNewRingName(e.target.value)}
          placeholder="New Ring Name"
          className="p-2 border rounded mr-2"
        />
        <button onClick={createRing} className="bg-blue-600 text-white px-4 py-2 rounded">
          Create Ring
        </button>
      </div>
    </div>
  )}

  {tab === "payouts" && (
    <div>
      <h3 className="text-lg font-semibold mb-2">Payout History</h3>
```

```
        <select value={selectedRing} onChange={(e) => setSelectedRing(e.target.value)}
classNameName="mb-4 p-2 border rounded block">
          <option value="all">All Rings</option>
          {rings.map((ring) => <option key={ring.id}
value={ring.id}>{ring.ringName}</option>)}
        </select>
        <ul classNameName="list-disc ml-6">
          {payoutMessages
            .filter((msg) => selectedRing === "all" || msg.ringId === selectedRing)
            .map((msg, i) => (
              <li key={i} classNameName="mb-2">
                <strong>{msg.ringName}</strong>: {msg.content} —
{msg.timestamp?.toDate().toLocaleString()}
                {!acknowledged.includes(msg.docId) && (
                  <button onClick={() => handleAcknowledge(msg.ringId, msg.docId)}
classNameName="ml-4 text-blue-600 underline text-sm">
                    Acknowledge
                  </button>
                )}
                {acknowledged.includes(msg.docId) && <span classNameName="ml-2 text-green-600 text-
sm">✓ Acknowledged</span>}
              </li>
            ))}
          {payoutMessages.length === 0 && <li>No payouts yet</li>}
        </ul>
      </div>
    )}

    {tab === "profile" && (
      <div>
        <h3 classNameName="text-lg font-semibold mb-2">Edit Profile</h3>
        <input
          type="text"
          value={displayName}
          onChange={(e) => setDisplayName(e.target.value)}
          placeholder="Full Name"
          classNameName="p-2 border rounded w-full mb-2"
        />
        <button onClick={handleProfileSave} classNameName="bg-blue-600 text-white px-4 py-2
rounded">
          Save
```

```
        </button>
        {profileSaved && <p className="text-green-600 mt-2">✅ Updated!</p>}
        <div className="mt-4">
          <input
            type="email"
            value={inviteEmail}
            onChange={(e) => setInviteEmail(e.target.value)}
            placeholder="Invite by email"
            className="p-2 border rounded mr-2"
          />
          <button onClick={handleInvite} className="bg-green-600 text-white px-4 py-2
rounded">
            Invite
          </button>
          {invited && <p className="mt-2 text-green-600">{invited}</p>}
        </div>
      </div>
    )}
    {error && <p className="text-red-500 mt-4">{error}</p>}
  </div>
 );
}

export default MemberDashboard;
```