```
import React, { useState, useMemo, useEffect } from 'react';
import {
  LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip,
ResponsiveContainer, AreaChart, Area, ComposedChart, Legend, BarChart,
Bar
} from 'recharts';
import {
  Activity, Calendar, ChevronRight, ClipboardList, PlusCircle, Save,
User, ArrowLeft, TrendingUp,
  AlertCircle, MapPin, Scale, Dumbbell, Smile, Frown, Meh, Target,
LogOut, Phone, ShieldCheck,
  Lock, Users, Camera, Image as ImageIcon, CheckCircle, Info, Filter,
ToggleLeft, ToggleRight, LayoutDashboard, HeartPulse
} from 'lucide-react';

// --- Constants ---
const PAIN_LOCATIONS = ['통증 없음', '목', '어깨', '허리', '골반', '무릎',
'발목', '손목', '기타'];

// FMS Tests with Bilateral flag & criteria
const FMS_TESTS = [
  {
    id: 'deepSquat',
    label: 'Deep Squat',
    shortLabel: 'DS',
    bilateral: false,
    desc: '전신 협응, 가동성, 안정성 평가',
    criteria: [
      '상체와 경골(정강이)이 평행하게 수직 유지',
      '대퇴골(허벅지)이 수평면보다 아래로 내려감',
      '무릎이 발 안쪽으로 모이지 않고 정렬 유지',
      '도웰(봉)이 발 위쪽에 위치하며 발뒤꿈치가 바닥에 고정'
    ]
  },
  {
    id: 'hurdleStep',
    label: 'Hurdle Step',
    shortLabel: 'HS',
    bilateral: true,
    desc: '보행 및 스텝 동작 안정성 평가',
    criteria: [
      '골반과 어깨의 수평 유지 (몸통 기울어짐 없음)',
      '발가락이 허들을 건드리지 않고 넘어가야 함',
```

```
      '지지하는 다리의 무릎이 흔들리지 않고 펴져 있음',
      '발목, 무릎, 고관절의 일직선 정렬'
    ]
  },
  {
    id: 'inlineLunge',
    label: 'Inline Lunge',
    shortLabel: 'IL',
    bilateral: true,
    desc: '좌우 불균형 및 안정성 평가',
    criteria: [
      '도웰이 머리, 등(흉추), 엉덩이(천골)에 모두 닿아 있음',
      '몸통이 흔들리지 않고 수직 유지',
      '앞발의 발뒤꿈치가 바닥에서 뜨지 않음',
      '뒷무릎이 앞발 뒤꿈치 뒤쪽 바닥에 닿음'
    ]
  },
  {
    id: 'shoulderMobility',
    label: 'Shoulder Mobility',
    shortLabel: 'SM',
    bilateral: true,
    desc: '어깨 관절 가동범위 평가',
    criteria: [
      '한 번의 부드러운 동작으로 주먹을 쥠',
      '등이 굽거나 목이 앞으로 빠지지 않음',
      '3점: 주먹 간격이 한 뼘(손길이) 이내',
      '2점: 주먹 간격이 한 뼘 반 이내'
    ]
  },
  {
    id: 'aslr',
    label: 'ASLR',
    shortLabel: 'ASLR',
    bilateral: true,
    desc: '하지 유연성 및 코어 조절 평가',
    criteria: [
      '들어올린 다리의 무릎이 펴져야 함',
      '반대쪽 다리가 바닥에서 뜨거나 무릎이 굽혀지지 않음',
      '3점: 복사뼈가 허벅지 중간 지점을 넘어감',
      '2점: 복사뼈가 무릎과 허벅지 중간 사이'
    ]
  },
```

```
  {
    id: 'trunkStability',
    label: 'Trunk Stability',
    shortLabel: 'TSPU',
    bilateral: false,
    desc: '척추 안정성 및 상체 근력 평가',
    criteria: [
      '몸통 전체가 일직선으로 동시에 올라옴',
      '허리가 아래로 처지거나 엉덩이가 먼저 들리지 않음',
      '남성: 엄지가 이마 라인 (3점) / 턱 라인 (2점)',
      '여성: 엄지가 턱 라인 (3점) / 쇄골 라인 (2점)'
    ]
  },
  {
    id: 'rotaryStability',
    label: 'Rotary Stability',
    shortLabel: 'RS',
    bilateral: true,
    desc: '회전 안정성 및 코어 협응력 평가',
    criteria: [
      '몸통이 바닥과 평행을 유지 (회전 없음)',
      '3점: 같은 쪽 팔과 다리가 만나고 펴짐',
      '2점: 대각선 팔과 다리가 만나고 펴짐',
      '팔꿈치와 무릎이 터치되어야 함'
    ]
  },
];

// --- Mock Data ---

const INITIAL_TEACHERS = [
  { id: 101, name: '김강사', phone: '1111' },
  { id: 102, name: '이재활', phone: '2222' },
];

const INITIAL_MEMBERS = [
  {
    id: 1,
    name: '홍길동',
    phone: '010-1234-5678',
    program: '허리 디스크 집중반',
    status: '회복중',
    assignedTeacherIds: [101],
```

```
    lastVisit: '2025-06-30',
    currentGoal: '통증 없이 골프 18홀 돌기',
    photos: {
      before: { front: null, back: null, left: null, right: null },
      after: []
    },
    history: [
      {
        date: '2025-06-01', location: '허리', pain: 8, bodyFat: 28.0,
condition: 2,
        bloodPressure: { systole: 135, diastole: 85 },
        fms: { deepSquat: 1, hurdleStep: { left: 1, right: 1 },
inlineLunge: { left: 1, right: 1 }, shoulderMobility: { left: 2, right:
1 }, aslr: { left: 1, right: 1 }, trunkStability: 1, rotaryStability: {
left: 1, right: 1 } },
        note: '첫 방문. 허리 숙일 때 통증 심함(VAS 8). FMS 전반적으로 1점대.'
      },
      {
        date: '2025-06-05', location: '허리', pain: 6, condition: 3,
        note: '호흡 패턴 교정 후 통증 약간 감소. 숙제(코어 호흡) 잘 해오심.'
      },
      {
        date: '2025-06-08', location: '허리', pain: 6,
        fms: { hurdleStep: { left: 1, right: 2 } }, // Only Hurdle Step
checked
        note: '보행 교정 수업. 오른쪽 허들 스텝 패턴 좋아짐.'
      },
      {
        date: '2025-06-12', location: '허리', pain: 5, bodyFat: 27.2,
condition: 4,
        grip: { left: 28, right: 30 }, singleLeg: { left: 15, right: 20
},
        bloodPressure: { systole: 128, diastole: 82 },
        note: '중간 점검. 체지방 소폭 감소. 한발 서기 왼쪽이 불안정함.'
      },
      {
        date: '2025-06-15', location: '허리', pain: 7, condition: 2, //
Pain flared up
        privateNote: '주말에 무리해서 스크린 골프 침... 다시 통증 호소. 강도
조절 필요.',
        note: '주말 무리한 활동으로 통증 재발. 오늘은 스트레칭 위주 진행.'
      },
      {
```

```
        date: '2025-06-20', location: '허리', pain: 4, condition: 4,
        fms: { deepSquat: 2 }, // Deep Squat improved
        note: '통증 다시 잡힘. 딥 스쿼트 자세 2점으로 향상됨(발뒤꿈치 안 뜸).'
      },
      {
        date: '2025-06-25', location: '통증 없음', pain: 2, condition:
5,
        fms: { inlineLunge: { left: 2, right: 2 }, shoulderMobility: {
left: 2, right: 2 } },
        note: '일상 생활 통증 거의 없음. 런지 자세 안정적.'
      },
      {
        date: '2025-06-30', location: '통증 없음', pain: 1, bodyFat:
26.0, condition: 5,
        bloodPressure: { systole: 120, diastole: 80 },
        fms: { deepSquat: 2, hurdleStep: { left: 2, right: 2 },
inlineLunge: { left: 2, right: 2 }, shoulderMobility: { left: 3, right:
2 }, aslr: { left: 2, right: 2 }, trunkStability: 2, rotaryStability: {
left: 2, right: 2 } },
        note: '월말 결산. FMS 평균 2점대로 회복. 다음달부터 근력 강화 시작.'
      },
    ]
  },
  {
    id: 2,
    name: '이필라',
    phone: '010-9876-5432',
    program: '신규 상담',
    status: '신규',
    assignedTeacherIds: [102],
    lastVisit: '2025-06-28',
    currentGoal: '상담 예정',
    photos: { before: { front: null, back: null, left: null, right:
null }, after: [] },
    history: []
  }
];

// --- Components ---

const Card = ({ children, className = "" }) => (
  <div className={`bg-white rounded-xl shadow-sm border
border-slate-100 ${className}`}>
```

```
    {children}
  </div>
);

const Button = ({ children, onClick, variant = 'primary', className =
"", disabled = false, size = 'md' }) => {
  const baseStyle = "rounded-lg font-medium transition-all duration-200
flex items-center justify-center gap-2 disabled:opacity-50
disabled:cursor-not-allowed";
  const sizeStyles = {
    sm: "px-2 py-1 text-xs",
    md: "px-4 py-3 text-sm",
    lg: "px-6 py-4 text-lg"
  }
  const variants = {
    primary: "bg-teal-600 text-white hover:bg-teal-700 shadow-md
shadow-teal-100",
    secondary: "bg-white text-slate-600 border border-slate-200
hover:bg-slate-50",
    ghost: "text-slate-500 hover:bg-slate-50",
    danger: "bg-red-50 text-red-600 hover:bg-red-100",
    outline: "bg-white text-teal-600 border border-teal-200
hover:bg-teal-50",
    selected: "bg-teal-600 text-white border border-teal-600
shadow-inner"
  };
  return (
    <button onClick={onClick} disabled={disabled}
className={`${baseStyle} ${sizeStyles[size]} ${variants[variant]}
${className}`}>
      {children}
    </button>
  );
};

const Badge = ({ status }) => {
  const styles = {
    '회복중': 'bg-green-100 text-green-700',
    '관리필요': 'bg-red-100 text-red-700',
    '안정': 'bg-blue-100 text-blue-700',
    '신규': 'bg-purple-100 text-purple-700'
  };
  return (
```

```
    <span className={`text-xs px-2 py-1 rounded-full font-bold
${styles[status] || 'bg-gray-100 text-gray-600'}`}>
      {status}
    </span>
  );
};

const ConditionIcon = ({ level }) => {
  if (level >= 4) return <Smile className="text-green-500" size={20}
/>;
  if (level === 3) return <Meh className="text-yellow-500" size={20}
/>;
  return <Frown className="text-red-500" size={20} />;
};

// --- Main App Component ---

export default function WellusRecoveryApp() {
  const [user, setUser] = useState(null);
  const [view, setView] = useState('login');
  const [selectedMemberId, setSelectedMemberId] = useState(null);

  // LocalStorage Init
  const [members, setMembers] = useState(() => {
    try {
      const savedMembers = localStorage.getItem('wellus_members');
      if (savedMembers) {
        const parsed = JSON.parse(savedMembers);
        return parsed.length > 0 ? parsed : INITIAL_MEMBERS;
      }
      return INITIAL_MEMBERS;
    } catch (e) {
      return INITIAL_MEMBERS;
    }
  });

  const [teachers, setTeachers] = useState(INITIAL_TEACHERS);

  useEffect(() => {
    localStorage.setItem('wellus_members', JSON.stringify(members));
  }, [members]);

  // Login Form
```

```
  const [loginForm, setLoginForm] = useState({ name: '', phone: '',
step: 'input' });

  // Record Form
  const [formSections, setFormSections] = useState({
    bodyFat: true, pain: true, physical: false, fms: false, memo: true
  });
  const [formData, setFormData] = useState({
    pain: 5, location: '허리', condition: 3, bodyFat: '', goal: '',
    grip: { left: '', right: '' }, singleLeg: { left: '', right: '' },
    bloodPressure: { systole: '', diastole: '' }, // Add blood pressure
state
    // Initialize bilateral tests as objects
    fms: {
        deepSquat: 0,
        hurdleStep: { left: 0, right: 0 },
        inlineLunge: { left: 0, right: 0 },
        shoulderMobility: { left: 0, right: 0 },
        aslr: { left: 0, right: 0 },
        trunkStability: 0,
        rotaryStability: { left: 0, right: 0 }
    },
    roms: { neck: '', shoulder: '', thoracic: '', lumbar: '', hip: '',
ankle: '' },
    note: '', privateNote: '',
  });

  const selectedMember = useMemo(() =>
    members.find(m => m.id === selectedMemberId),
  [members, selectedMemberId]);

  // --- Helpers ---
  const canSeePrivateNote = user?.role === 'admin' || user?.role ===
'teacher';
  const canEditRecord = user?.role === 'admin' || user?.role ===
'teacher';

  // --- Handlers ---
  const handleLogin = () => {
    const name = loginForm.name.trim();
    const phone = loginForm.phone.trim();
    if (!name || !phone) return alert('이름과 전화번호를 모두
입력해주세요.');
```

```
    if (name === '원장님' && phone === '0000') {
      setUser({ id: 9999, name: '원장님', role: 'admin' });
      setView('list');
      return;
    }
    const teacher = teachers.find(t => t.name === name && t.phone ===
phone);
    if (teacher) {
        setUser({ ...teacher, role: 'teacher' });
        setView('list');
        return;
    }
    let member = members.find(m => m.name === name && m.phone ===
phone);
    if (!member) {
      if(window.confirm(`${name}님, 등록된 정보가 없습니다.\n새로운 회원으로
가입하시겠습니까?`)) {
        const newMember = {
          id: Date.now(), name, phone, program: '신규 상담 대기', status:
'신규',
          assignedTeacherIds: [], lastVisit: new
Date().toISOString().split('T')[0],
          currentGoal: '목표를 설정해주세요',
          photos: { before: { front: null, back: null, left: null,
right: null }, after: [] },
          history: []
        };
        setMembers(prev => [...prev, newMember]);
        member = newMember;
      } else { return; }
    }
    setUser({ ...member, role: 'member' });
    setSelectedMemberId(member.id);
    setView('detail');
  };

  const handlePhotoUpload = (e, type, position = null) => {
    const file = e.target.files[0];
    if (!file) return;
    const reader = new FileReader();
    reader.onloadend = () => {
        const imageUrl = reader.result;
```

```
        setMembers(prev => prev.map(m => {
            if (m.id === selectedMemberId) {
                const newPhotos = { ...m.photos };
                if (type === 'before') { newPhotos.before[position] =
imageUrl; }
                else { newPhotos.after.push({ date: new
Date().toISOString().split('T')[0], url: imageUrl }); }
                return { ...m, photos: newPhotos };
            } return m;
        }));
    };
    reader.readAsDataURL(file);
  };

  const handleTeacherAssignment = (memberId, teacherId) => {
    setMembers(prev => prev.map(m => {
        if (m.id === memberId) {
            const currentIds = m.assignedTeacherIds || [];
            const newIds = currentIds.includes(teacherId)
                ? currentIds.filter(id => id !== teacherId)
                : [...currentIds, teacherId];
            return { ...m, assignedTeacherIds: newIds };
        } return m;
    }));
  };

  const handleSaveRecord = () => {
    const newRecord = {
      date: new Date().toISOString().split('T')[0],
      location: formData.location,
      pain: formData.location === '통증 없음' ? 0 :
parseInt(formData.pain),
      condition: formData.condition,
      bodyFat: formData.bodyFat ? parseFloat(formData.bodyFat) : null,
      grip: formSections.physical ? { left:
parseFloat(formData.grip.left)||0, right:
parseFloat(formData.grip.right)||0 } : null,
      singleLeg: formSections.physical ? { left:
parseFloat(formData.singleLeg.left)||0, right:
parseFloat(formData.singleLeg.right)||0 } : null,
      bloodPressure: formSections.physical ? { systole:
parseFloat(formData.bloodPressure.systole)||0, diastole:
parseFloat(formData.bloodPressure.diastole)||0 } : null,
```

```
      fms: formSections.fms ? { ...formData.fms } : null,
      roms: { ...formData.roms },
      note: formData.note,
      privateNote: formData.privateNote
    };

    setMembers(prev => prev.map(m => {
      if (m.id === selectedMemberId) {
        return {
          ...m,
          currentGoal: formData.goal || m.currentGoal,
          history: [...m.history, newRecord]
        };
      } return m;
    }));

    setView('detail');
    // Reset form
    setFormData(prev => ({
      ...prev,
      pain: 5, note: '', privateNote: '', bodyFat: '', goal: '',
      bloodPressure: { systole: '', diastole: '' },
      fms: {
        deepSquat: 0,
        hurdleStep: { left: 0, right: 0 },
        inlineLunge: { left: 0, right: 0 },
        shoulderMobility: { left: 0, right: 0 },
        aslr: { left: 0, right: 0 },
        trunkStability: 0,
        rotaryStability: { left: 0, right: 0 }
      }
    }));
  };

  const handleFmsChange = (testId, val, side = null) => {
    setFormData(prev => {
        const currentVal = prev.fms[testId];
        let newVal;
        if (side) {
            newVal = { ...(typeof currentVal === 'object' ? currentVal
: {}), [side]: parseInt(val) };
        } else {
            newVal = parseInt(val);
```

```
        }
        return {
            ...prev,
            fms: { ...prev.fms, [testId]: newVal }
        }
    });
  };

  const toggleSection = (section) => {
      setFormSections(prev => ({...prev, [section]: !prev[section]}));
  };

  // --- Views ---

  const LoginView = () => (
    <div className="flex flex-col items-center justify-center
min-h-screen p-6 bg-teal-600 animate-fade-in">
      <div className="bg-white p-8 rounded-2xl shadow-2xl w-full
max-w-sm">
        <div className="text-center mb-8">
          <div className="w-16 h-16 bg-teal-100 text-teal-600
rounded-full flex items-center justify-center mx-auto mb-4">
            <Activity size={32} />
          </div>
          <h1 className="text-2xl font-extrabold
text-slate-800">Wellus</h1>
          <p className="text-sm text-slate-500">웰러스 리커버리 시스템</p>
        </div>
        <div className="space-y-4">
            <input type="text" placeholder="이름 입력"
value={loginForm.name} onChange={(e) => setLoginForm({...loginForm,
name: e.target.value})} className="w-full p-3 bg-slate-50 border
border-slate-200 rounded-lg outline-none"/>
            <input type="tel" placeholder="연락처 입력"
value={loginForm.phone} onChange={(e) => setLoginForm({...loginForm,
phone: e.target.value})} className="w-full p-3 bg-slate-50 border
border-slate-200 rounded-lg outline-none"/>
            <Button onClick={handleLogin} className="w-full py-3
text-lg mt-2">로그인 / 회원가입</Button>
            <div className="text-xs text-slate-400 mt-4 bg-slate-50 p-3
rounded"><p>• 원장님: 원장님 / 0000</p><p>• 강사: 김강사 / 1111</p><p>•
회원: 본인 이름/번호</p></div>
            <div className="text-center mt-6">
```

```
                <button
                    onClick={() => {
                        if(window.confirm('모든 데이터가 초기화되고 테스트용(홍길동) 데이터로 복구됩니다. 계속하시겠습니까?')) {
                            localStorage.removeItem('wellus_members');
                            setMembers(INITIAL_MEMBERS);
                            alert("데이터가 초기화되었습니다.");
                        }
                    }}
                    className="text-xs text-red-300 underline"
                >
                    데이터 초기화 (테스트용)
                </button>
            </div>
        </div>
      </div>
    </div>
  );

  const MemberListView = () => {
    const visibleMembers = members.filter(m => {
        if (user.role === 'admin') return true;
        if (user.role === 'teacher') return m.assignedTeacherIds?.includes(user.id);
        return false;
    });

    return (
        <div className="space-y-4 animate-fade-in">
        <div className="flex justify-between items-center mb-6">
            <div>
            <h2 className="text-xl font-bold text-slate-800">{user.role === 'admin' ? '전체 회원 관리' : '내 담당 회원'}</h2>
            </div>
            <Button variant="secondary" onClick={() => { setUser(null); setView('login'); }} className="!px-3"><LogOut size={18} /></Button>
        </div>
        <div className="grid gap-3">
            {visibleMembers.map(member => (
            <Card key={member.id} className="hover:border-teal-300 transition-colors cursor-pointer group relative">
                <div className="p-4 flex items-center justify-between" onClick={() => { setSelectedMemberId(member.id); setView('detail'); }}>
```

```
                <div className="flex items-center gap-4">
                    <div className="w-12 h-12 bg-teal-50 rounded-full
flex items-center justify-center text-teal-600 font-bold
text-lg">{member.name[0]}</div>
                    <div>
                    <div className="flex items-center gap-2 mb-1"><h3
className="font-bold text-slate-800">{member.name}</h3><Badge
status={member.status} /></div>
                    <p className="text-xs text-slate-500 truncate
max-w-[150px]">{member.program}</p>
                    </div>
                </div>
                <ChevronRight className="text-slate-300
group-hover:text-teal-500" />
                </div>
                {user.role === 'admin' && (<button onClick={(e) => {
e.stopPropagation(); setSelectedMemberId(member.id);
setView('admin_assign'); }} className="absolute top-4 right-12 p-2
text-slate-400 hover:text-teal-600"><Users size={16} /></button>)}
            </Card>
            ))}
        </div>
        </div>
    );
  };

  const AdminAssignView = () => {
    if (!selectedMember) return null;
    return (
        <div className="space-y-6 animate-fade-in">
            <div className="flex items-center gap-2 mb-2"><Button
variant="ghost" onClick={() => setView('list')} className="!px-2
-ml-2"><ArrowLeft size={20} /></Button><h2 className="text-xl font-bold
text-slate-800">{selectedMember.name}님 담당 배정</h2></div>
            <Card className="p-4">
                <div className="space-y-2">{teachers.map(teacher => {
                        const isAssigned =
selectedMember.assignedTeacherIds?.includes(teacher.id);
                        return (<div key={teacher.id} onClick={() =>
handleTeacherAssignment(selectedMember.id, teacher.id)}
className={`flex items-center justify-between p-3 rounded-lg border
cursor-pointer ${isAssigned ? 'border-teal-500 bg-teal-50' :
'border-slate-200'}`}><span className={isAssigned ? 'text-teal-900
```

```
font-bold' : 'text-slate-600'}>{teacher.name}</span>{isAssigned ?
<CheckCircle className="text-teal-600" size={20}/> : <div
className="w-5 h-5 rounded-full border border-slate-300"/>}</div>)
                    })}
                </div>
            </Card>
        </div>
    )
  }

  const MemberDetailView = () => {
    if (!selectedMember) return null;
    const [activeTab, setActiveTab] = useState('chart');
    const [visibleKeys, setVisibleKeys] = useState({
        pain: true, bodyFat: true, gripL: false, gripR: false,
balanceL: false, balanceR: false
    });
    const [activeFmsTestId, setActiveFmsTestId] =
useState('deepSquat');

    const toggleChartKey = (key) => setVisibleKeys(prev => ({...prev,
[key]: !prev[key]}));

    const activeFmsTest = FMS_TESTS.find(t => t.id ===
activeFmsTestId);

    // Process Chart Data
    const chartData = useMemo(() => {
        // Sort history by date to ensure graph is chronological
        const sortedHistory = [...selectedMember.history].sort((a,b) =>
new Date(a.date) - new Date(b.date));

        return sortedHistory.map(h => ({
            date: h.date.substring(5), // Show MM-DD
            fullDate: h.date,
            bodyFat: h.bodyFat, // Recharts handles nulls by breaking
lines or connecting if connectNulls used
            pain: h.location !== '통증 없음' ? h.pain : 0,
            gripL: h.grip?.left, gripR: h.grip?.right,
            balanceL: h.singleLeg?.left, balanceR: h.singleLeg?.right,
            // Add active FMS score to main data for easy access
            fmsVal: h.fms ? (typeof h.fms[activeFmsTestId] === 'object'
? null : h.fms[activeFmsTestId]) : null,
```

```
            fmsLeft: h.fms && h.fms[activeFmsTestId] && typeof
h.fms[activeFmsTestId] === 'object' ? h.fms[activeFmsTestId].left :
null,
            fmsRight: h.fms && h.fms[activeFmsTestId] && typeof
h.fms[activeFmsTestId] === 'object' ? h.fms[activeFmsTestId].right :
null,
        }));
    }, [selectedMember, activeFmsTestId]);

    const latest = selectedMember.history[selectedMember.history.length
- 1] || {};

    return (
      <div className="space-y-6 animate-fade-in pb-24">
        <div className="flex items-center justify-between mb-2">
            <div className="flex items-center gap-2">
                {(user?.role === 'admin' || user?.role === 'teacher')
&& (<Button variant="ghost" onClick={() => setView('list')}
className="!px-2 -ml-2"><ArrowLeft size={20} /></Button>)}
                <div><h2 className="text-xl font-bold
text-slate-800">{selectedMember.name} 님의 기록</h2></div>
            </div>
            {user?.role === 'member' && (<button onClick={() => {
setUser(null); setView('login'); }} className="text-xs text-slate-400
underline">로그아웃</button>)}
        </div>

        <div className="flex bg-white p-1 rounded-xl shadow-sm border
border-slate-100">
            {['chart', 'photo', 'history'].map(tab => (
                <button key={tab} onClick={() => setActiveTab(tab)}
className={`flex-1 py-2 text-sm font-bold rounded-lg transition-colors
${activeTab === tab ? 'bg-teal-600 text-white' : 'text-slate-400'}`}>
                    {tab === 'chart' ? '통계' : tab === 'photo' ? '체형'
: '기록'}
                </button>
            ))}
        </div>

        {activeTab === 'chart' && (
             <div className="space-y-6">
                {/* Top Stats Grid - Restored */}
                <div className="grid grid-cols-2 gap-3">
```

```
                    <Card className="p-3 bg-white border-slate-100">
                        <p className="text-[10px] text-slate-400
font-bold mb-1 flex items-center gap-1"><Smile size={10}/> 최근
컨디션</p>
                        <div className="flex items-center gap-2">
                            <ConditionIcon level={latest.condition ||
3} />
                            <span className="text-sm font-bold
text-slate-700">
                                {latest.condition >= 4 ? '좋음' :
latest.condition === 3 ? '보통' : '나쁨'}
                            </span>
                        </div>
                    </Card>
                    <Card className="p-3 bg-green-50 border-green-100">
                        <p className="text-[10px] text-green-700
font-bold mb-1 flex items-center gap-1"><Target size={10}/> 현재 운동
목표</p>
                        <div className="text-xs font-bold
text-slate-700 truncate">
                            {selectedMember.currentGoal || '목표 설정
전'}
                        </div>
                    </Card>
                </div>

                {/* Blood Pressure */}
                {latest.bloodPressure && (latest.bloodPressure.systole
|| latest.bloodPressure.diastole) && (
                    <Card className="p-3 bg-red-50 border-red-100">
                        <p className="text-[10px] text-red-700
font-bold mb-1 flex items-center gap-1"><HeartPulse size={10}/> 최근
혈압 (mmHg)</p>
                        <div className="flex items-baseline gap-1">
                            <span className="text-2xl font-bold
text-slate-800">{latest.bloodPressure.systole || '-'}</span>
                            <span className="text-sm
text-slate-500">/</span>
                            <span className="text-2xl font-bold
text-slate-800">{latest.bloodPressure.diastole || '-'}</span>
                        </div>
                    </Card>
                )}
```

```
                {/* 1. Main Stats Chart */}
                <div>
                    <h3 className="font-bold text-slate-700 mb-2 flex
items-center gap-2"><TrendingUp size={16} className="text-teal-600"/>
종합 변화 추이</h3>
                    <div className="flex flex-wrap gap-2 mb-2">
                        {[{key: 'bodyFat', label: '체지방'}, {key:
'pain', label: '통증'}, {key: 'gripL', label: '악력L'}, {key: 'gripR',
label: '악력R'}, {key: 'balanceL', label: '한발L'}, {key: 'balanceR',
label: '한발R'}].map(item => (
                            <button key={item.key} onClick={() =>
toggleChartKey(item.key)} className={`px-2 py-1 rounded text-[10px]
font-bold transition-all border ${visibleKeys[item.key] ? 'bg-teal-600
text-white border-teal-600' : 'bg-white text-slate-400
border-slate-200'}`}>{item.label}</button>
                        ))}
                    </div>
                    <Card className="p-4">
                        <div className="h-48 w-full">
                            <ResponsiveContainer width="100%"
height="100%">
                            <ComposedChart data={chartData}>
                                <CartesianGrid strokeDasharray="3 3"
vertical={false} stroke="#f1f5f9" />
                                <XAxis dataKey="date" tick={{fontSize:
10}} tickLine={false} axisLine={false} />
                                <YAxis yAxisId="left" domain={[0, 10]}
hide />
                                <YAxis yAxisId="right"
orientation="right" domain={['auto', 'auto']} tick={{fontSize: 10}}
tickLine={false} axisLine={false} />
                                <Tooltip contentStyle={{borderRadius:
'8px', border: 'none', boxShadow: '0 4px 6px -1px rgb(0 0 0 / 0.1)'}}/>
                                {/* Added connectNulls={true} to all
lines to handle sparse data */}
                                {/* Changed Body Fat color to Green
(#10b981) */}
                                {visibleKeys.bodyFat && <Area
connectNulls={true} yAxisId="right" type="monotone" dataKey="bodyFat"
name="체지방(%)" stroke="none" fill="#10b981" fillOpacity={0.2} />}
```

```
                                {visibleKeys.pain && <Line
connectNulls={true} yAxisId="left" type="monotone" dataKey="pain"
name="통증(VAS)" stroke="#ef4444" strokeWidth={2} dot={{r: 4}} />}
                                {visibleKeys.gripL && <Line
connectNulls={true} yAxisId="right" type="monotone" dataKey="gripL"
name="악력(좌)" stroke="#3b82f6" strokeDasharray="5 5" />}
                                {visibleKeys.gripR && <Line
connectNulls={true} yAxisId="right" type="monotone" dataKey="gripR"
name="악력(우)" stroke="#2563eb" />}
                                {visibleKeys.balanceL && <Line
connectNulls={true} yAxisId="right" type="monotone" dataKey="balanceL"
name="한발(좌)" stroke="#10b981" strokeDasharray="5 5" />}
                                {visibleKeys.balanceR && <Line
connectNulls={true} yAxisId="right" type="monotone" dataKey="balanceR"
name="한발(우)" stroke="#059669" />}
                            </ComposedChart>
                            </ResponsiveContainer>
                        </div>
                    </Card>
                </div>

                {/* 2. Separate FMS Analysis Section */}
                <div>
                    <h3 className="font-bold text-slate-700 mb-2 flex
items-center gap-2"><Activity size={16} className="text-indigo-600"/>
FMS 세부 분석</h3>
                    <div className="flex overflow-x-auto gap-2 pb-2
mb-2 no-scrollbar">
                        {FMS_TESTS.map(test => (
                            <button
                                key={test.id}
                                onClick={() =>
setActiveFmsTestId(test.id)}
                                className={`px-3 py-2 rounded-lg
text-xs font-bold whitespace-nowrap border transition-all ${
                                    activeFmsTestId === test.id
                                    ? 'bg-indigo-600 text-white
border-indigo-600 shadow-md'
                                    : 'bg-white text-slate-500
border-slate-200'
                                }`}
                            >
```

```
                                {test.shortLabel || test.label.split('
')[0]}
                            </button>
                        ))}
                    </div>
                    <Card className="p-4 bg-indigo-50
border-indigo-100">
                        <div className="flex justify-between
items-center mb-4">
                            <span className="text-sm font-bold
text-indigo-900">{activeFmsTest?.label} 변화</span>
                            <span className="text-[10px]
text-indigo-400 bg-white px-2 py-1 rounded-full border
border-indigo-100">0~3점 (높을수록 좋음)</span>
                        </div>
                        <div className="h-40 w-full">
                            <ResponsiveContainer width="100%"
height="100%">
                                <LineChart data={chartData}>
                                    <CartesianGrid strokeDasharray="3
3" vertical={false} stroke="#e0e7ff" />
                                    <XAxis dataKey="date"
tick={{fontSize: 10, fill: '#6366f1'}} tickLine={false}
axisLine={false} />
                                    <YAxis domain={[0, 3]}
tickCount={4} tick={{fontSize: 10, fill: '#6366f1'}} tickLine={false}
axisLine={false} />
                                    <Tooltip
contentStyle={{borderRadius: '8px', border: 'none'}} />
                                    <Legend iconSize={8}
wrapperStyle={{fontSize: '10px'}}/>
                                    {/* Added connectNulls={true} to
FMS charts as well */}
                                    {activeFmsTest?.bilateral ? (
                                        <>
                                            <Line connectNulls={true}
type="step" dataKey="fmsLeft" name="Left (좌)" stroke="#4f46e5"
strokeWidth={2} dot={{r:3, fill:'#4f46e5'}} />
                                            <Line connectNulls={true}
type="step" dataKey="fmsRight" name="Right (우)" stroke="#a5b4fc"
strokeWidth={2} dot={{r:3, fill:'#a5b4fc'}} />
                                        </>
                                    ) : (
```

```
                                        <Line connectNulls={true}
type="step" dataKey="fmsVal" name="Score" stroke="#4f46e5"
strokeWidth={2} dot={{r:4, fill:'#4f46e5'}} />
                                    )}
                                </LineChart>
                            </ResponsiveContainer>
                        </div>
                    </Card>
                </div>
            </div>
        )}

        {activeTab === 'photo' && (
            <div className="space-y-6">
                <Card className="p-4">
                    <h3 className="font-bold text-slate-700 mb-3 flex
items-center gap-2"><Camera size={18} className="text-teal-500"/>
Before</h3>
                    <div className="grid grid-cols-2 gap-3">
                        {['front', 'back', 'left', 'right'].map(pos =>
(
                            <div key={pos} className="relative
aspect-[3/4] bg-slate-100 rounded-lg overflow-hidden border
border-slate-200 group">
                                {selectedMember.photos.before[pos] ?
(<img src={selectedMember.photos.before[pos]} className="w-full h-full
object-cover" />) : (<div className="absolute inset-0 flex flex-col
items-center justify-center text-slate-400"><ImageIcon size={24}/><span
className="text-xs mt-1 uppercase">{pos}</span></div>)}
                                {canEditRecord && (<label
className="absolute inset-0 bg-black/50 flex items-center
justify-center opacity-0 group-hover:opacity-100 cursor-pointer
text-white font-bold text-xs"><input type="file" className="hidden"
accept="image/*" onChange={(e) => handlePhotoUpload(e, 'before', pos)}
/>변경</label>)}
                            </div>
                        ))}
                    </div>
                </Card>
                <Card className="p-4">
                    <div className="flex justify-between mb-3"><h3
className="font-bold text-slate-700 flex items-center
gap-2"><TrendingUp size={18} className="text-blue-500"/>
```

```
After</h3>{canEditRecord && (<label className="bg-blue-50 text-blue-600
px-3 py-1 rounded-lg text-xs font-bold cursor-pointer
hover:bg-blue-100"><input type="file" className="hidden"
accept="image/*" onChange={(e) => handlePhotoUpload(e, 'after')} />+
추가</label>)}</div>
                    <div className="grid grid-cols-3
gap-2">{selectedMember.photos.after.map((photo, idx) => (<div key={idx}
className="relative aspect-square bg-slate-100 rounded-lg
overflow-hidden"><img src={photo.url} className="w-full h-full
object-cover" /><span className="absolute bottom-0 left-0 right-0
bg-black/60 text-white text-[10px] text-center
py-1">{photo.date}</span></div>))}</div>
                </Card>
            </div>
        )}

        {activeTab === 'history' && (
             <div className="space-y-3">
                 {selectedMember.history.slice().reverse().map((record,
idx) => (
                     <Card key={idx} className="p-4 border-l-4
border-l-teal-500">
                         <div className="flex justify-between
items-center mb-2">
                             <span className="text-sm font-bold
text-slate-800">{record.date}</span>
                             <span className={`text-xs px-2 py-0.5
rounded font-medium ${record.location === '통증 없음' ? 'bg-green-100
text-green-700' : 'bg-slate-100
text-slate-600'}`}>{record.location}</span>
                         </div>
                         {record.note && <div className="bg-slate-50 p-2
rounded mb-2 text-sm text-slate-700">{record.note}</div>}
                         {canSeePrivateNote && record.privateNote &&
<div className="bg-amber-50 p-2 rounded border border-amber-100 text-sm
text-slate-700"><Lock size={10} className="inline mr-1
text-amber-600"/>{record.privateNote}</div>}

                         <div className="grid grid-cols-2 gap-2 mt-2">
                             {record.bodyFat && <div className="text-xs
bg-purple-50 p-1 rounded text-purple-700">체지방:
{record.bodyFat}%</div>}
```

```
                            {record.grip && <div className="text-xs
bg-blue-50 p-1 rounded text-blue-700">악력:
L{record.grip.left}/R{record.grip.right}</div>}
                            {record.bloodPressure &&
(record.bloodPressure.systole || record.bloodPressure.diastole) && <div
className="text-xs bg-red-50 p-1 rounded text-red-700">혈압:
{record.bloodPressure.systole}/{record.bloodPressure.diastole}</div>}
                        </div>

                        {record.fms && (
                            <div className="mt-2 pt-2 border-t
border-slate-100">
                                <p className="text-xs font-bold
text-indigo-600 mb-1">FMS 평가</p>
                                <div className="flex flex-wrap gap-1">

{Object.entries(record.fms).map(([key, val]) => {
                                        const shortName =
key.substring(0,2).toUpperCase();
                                        const displayVal = typeof val
=== 'object' ? `L${val.left}/R${val.right}` : val;
                                        return (
                                            <span key={key}
className="text-[10px] bg-indigo-50 px-1.5 py-0.5 rounded
text-indigo-800 border border-indigo-100">
                                                {shortName}:
{displayVal}
                                            </span>
                                        )
                                    })}
                                </div>
                            </div>
                        )}
                    </Card>
                ))}
            </div>
        )}

        {canEditRecord && (<div className="fixed bottom-6 right-6
lg:absolute lg:bottom-6 lg:right-6"><button onClick={() =>
setView('form')} className="w-14 h-14 bg-teal-600 rounded-full
shadow-lg shadow-teal-200 flex items-center justify-center text-white
```

```
hover:scale-105 transition-transform"><PlusCircle size={28}
/></button></div>)}
      </div>
    );
  };

  const RecordFormView = () => (
    <div className="space-y-6 animate-fade-in pb-10">
      <div className="flex items-center gap-2 mb-2"><Button
variant="ghost" onClick={() => setView('detail')} className="!px-2
-ml-2"><ArrowLeft size={20} /></Button><h2 className="text-xl font-bold
text-slate-800">기록 추가</h2></div>

      {/* 1. Selective Input Toggles */}
      <Card className="p-4">
        <h3 className="font-bold text-slate-700 text-sm mb-3">오늘 기록할
항목을 선택하세요</h3>
        <div className="flex flex-wrap gap-2">
            {[
                {id: 'pain', label: '통증/부위', icon: <MapPin
size={14}/>},
                {id: 'bodyFat', label: '체지방/컨디션', icon: <Scale
size={14}/>},
                {id: 'physical', label: '신체기능', icon: <Dumbbell
size={14}/>},
                {id: 'fms', label: 'FMS 평가', icon: <Activity
size={14}/>},
                {id: 'memo', label: '메모', icon: <ClipboardList
size={14}/>},
            ].map(item => (
                <button
                    key={item.id}
                    onClick={() => toggleSection(item.id)}
                    className={`flex items-center gap-1.5 px-3 py-2
rounded-lg text-xs font-bold border transition-all ${
                        formSections[item.id]
                        ? 'bg-teal-600 text-white border-teal-600
shadow-md transform scale-105'
                        : 'bg-white text-slate-500 border-slate-200
hover:bg-slate-50'
                    }`}
                >
                    {item.icon} {item.label}
```

```
                </button>
            ))}
        </div>
      </Card>

      {/* 2. Body Fat & Condition */}
      {formSections.bodyFat && (
          <Card className="p-5 space-y-4 animate-fade-in">
             <div className="flex justify-between items-center"><label
className="font-bold text-slate-700 text-sm flex gap-2"><Scale
size={16} className="text-purple-600"/>체지방률 & 컨디션</label></div>
             <div className="flex gap-4">
                <div className="w-1/2">
                    <label className="text-xs text-slate-500 mb-1
block">체지방률(%)</label>
                    <input type="number" placeholder="25.0"
value={formData.bodyFat} onChange={(e)=>setFormData({...formData,
bodyFat: e.target.value})} className="w-full p-2 border rounded
font-bold text-purple-700"/>
                </div>
                <div className="w-1/2">
                    <label className="text-xs text-slate-500 mb-1
block">컨디션</label>
                    <div className="flex justify-between bg-slate-50
p-1.5 rounded">{[1,3,5].map(lvl => (<button key={lvl}
onClick={()=>setFormData({...formData, condition: lvl})}
className={`p-1 rounded-full ${formData.condition===lvl?'bg-white
shadow':''}`}><ConditionIcon level={lvl}/></button>))}</div>
                </div>
             </div>
          </Card>
      )}

      {/* 3. Pain Section */}
      {formSections.pain && (
          <Card className="p-5 space-y-4 animate-fade-in">
            <label className="font-bold text-slate-700 text-sm flex
gap-2"><AlertCircle size={16} className="text-red-500"/>통증 부위 및
점수</label>
            <div className="grid grid-cols-4 gap-2">
                {PAIN_LOCATIONS.map(loc => (
                    <button key={loc} onClick={() =>
setFormData({...formData, location: loc, pain: loc === '통증 없음' ? 0 :
```

```
formData.pain})} className={`py-2 text-xs rounded-lg border
${formData.location === loc ? 'bg-red-500 text-white border-red-500
font-bold' : 'bg-white text-slate-500'}`}>{loc}</button>
                ))}
            </div>
            {formData.location !== '통증 없음' && (
                <div className="bg-red-50 p-3 rounded-lg border
border-red-100">
                    <div className="flex justify-between text-red-600
font-bold text-xs mb-2"><span>통증 강도 (VAS)</span><span
className="text-lg">{formData.pain}</span></div>
                    <input type="range" min="0" max="10"
value={formData.pain} onChange={(e)=>setFormData({...formData, pain:
e.target.value})} className="w-full accent-red-500"/>
                </div>
            )}
        </Card>
    )}

    {/* 4. Physical (Grip/Balance/BloodPressure) */}
    {formSections.physical && (
        <Card className="p-5 space-y-4 animate-fade-in">
            <label className="font-bold text-slate-700 text-sm flex
gap-2"><Dumbbell size={16} className="text-blue-500"/>신체 기능
측정</label>
            <div className="grid grid-cols-2 gap-4">
              <div>
                  <label className="text-xs text-slate-500 block
mb-1">악력 (L/R kg)</label>
                  <div className="flex gap-1"><input type="number"
placeholder="L" className="w-full p-2 bg-slate-50 rounded text-center
text-sm" value={formData.grip.left}
onChange={(e)=>setFormData({...formData, grip: {...formData.grip, left:
e.target.value}})}/><input type="number" placeholder="R"
className="w-full p-2 bg-slate-50 rounded text-center text-sm"
value={formData.grip.right} onChange={(e)=>setFormData({...formData,
grip: {...formData.grip, right: e.target.value}})}/></div>
                </div>
                <div>
                  <label className="text-xs text-slate-500 block
mb-1">한발서기 (L/R sec)</label>
                  <div className="flex gap-1"><input type="number"
placeholder="L" className="w-full p-2 bg-slate-50 rounded text-center
```

```
text-sm" value={formData.singleLeg.left}
onChange={(e)=>setFormData({...formData, singleLeg:
{...formData.singleLeg, left: e.target.value}})}/><input type="number"
placeholder="R" className="w-full p-2 bg-slate-50 rounded text-center
text-sm" value={formData.singleLeg.right}
onChange={(e)=>setFormData({...formData, singleLeg:
{...formData.singleLeg, right: e.target.value}})}/></div>
                </div>
              </div>
              <div className="pt-2 border-t border-slate-100">
                  <label className="text-xs text-slate-500 block
mb-1">혈압 (수축기/이완기 mmHg)</label>
                  <div className="flex gap-2 items-center">
                      <input type="number" placeholder="120"
className="w-20 p-2 bg-slate-50 rounded text-center text-sm"
value={formData.bloodPressure.systole}
onChange={(e)=>setFormData({...formData, bloodPressure:
{...formData.bloodPressure, systole: e.target.value}})}/>
                      <span className="text-slate-400">/</span>
                      <input type="number" placeholder="80"
className="w-20 p-2 bg-slate-50 rounded text-center text-sm"
value={formData.bloodPressure.diastole}
onChange={(e)=>setFormData({...formData, bloodPressure:
{...formData.bloodPressure, diastole: e.target.value}})}/>
                  </div>
              </div>
          </Card>
      )}

      {/* 5. FMS Section - Updated for L/R */}
      {formSections.fms && (
        <Card className="p-5 space-y-4 animate-fade-in">
            <h3 className="font-bold text-slate-700 text-sm flex
items-center gap-2"><Activity size={16} className="text-indigo-500"/>
FMS 평가 (0~3점)</h3>
            <div className="space-y-6">
                {FMS_TESTS.map(test => (
                    <div key={test.id} className="border-b
border-slate-100 pb-4 last:border-0 last:pb-0">
                        <div className="flex justify-between
items-start mb-2">
```

```
                            <div><span className="text-sm font-bold
text-slate-700 block">{test.label}</span><span className="text-[10px]
text-slate-400">{test.desc}</span></div>
                        </div>

                        {/* Scoring Area */}
                        {test.bilateral ? (
                            <div className="grid grid-cols-2 gap-3
mb-2">
                                <div className="space-y-1">
                                    <span className="text-[10px]
text-slate-500 font-bold block text-center">Left (좌)</span>
                                    <div className="flex justify-center
gap-1 bg-slate-100 p-1 rounded-lg">
                                        {[0,1,2,3].map(score => (
                                            <button key={score}
onClick={() => handleFmsChange(test.id, score, 'left')}
                                                className={`w-6 h-6
rounded text-[10px] font-bold transition-colors
${formData.fms[test.id]?.left === score ? 'bg-indigo-500 text-white' :
'text-slate-400 hover:bg-slate-200'}`}>
                                                {score}
                                            </button>
                                        ))}
                                    </div>
                                </div>
                                <div className="space-y-1">
                                    <span className="text-[10px]
text-slate-500 font-bold block text-center">Right (우)</span>
                                    <div className="flex justify-center
gap-1 bg-slate-100 p-1 rounded-lg">
                                        {[0,1,2,3].map(score => (
                                            <button key={score}
onClick={() => handleFmsChange(test.id, score, 'right')}
                                                className={`w-6 h-6
rounded text-[10px] font-bold transition-colors
${formData.fms[test.id]?.right === score ? 'bg-indigo-500 text-white' :
'text-slate-400 hover:bg-slate-200'}`}>
                                                {score}
                                            </button>
                                        ))}
                                    </div>
                                </div>
```

```
                            </div>
                        ) : (
                            <div className="flex gap-1 bg-slate-100 p-1
rounded-lg w-fit mb-2">
                                {[0,1,2,3].map(score => (
                                    <button key={score} onClick={() =>
handleFmsChange(test.id, score)}
                                        className={`w-8 h-8 rounded
text-xs font-bold transition-colors ${formData.fms[test.id] === score ?
'bg-indigo-500 text-white' : 'text-slate-400 hover:bg-slate-200'}`}>
                                        {score}
                                    </button>
                                ))}
                            </div>
                        )}

                        <div className="bg-indigo-50 p-2.5 rounded-lg
text-xs text-indigo-800 space-y-1">
                            <p className="font-bold flex items-center
gap-1 mb-1 text-[10px] text-indigo-500"><Info size={10}/> 평가 기준</p>
                            <ul className="list-disc list-inside
space-y-0.5 opacity-80 text-[10px]">
                                {test.criteria.map((crit, idx) => (<li
key={idx}>{crit}</li>))}
                            </ul>
                        </div>
                    </div>
                ))}
            </div>
        </Card>
      )}

      {/* 6. Memo Section */}
      {formSections.memo && (
          <Card className="p-5 space-y-4 animate-fade-in">
            <label className="font-bold text-slate-700 text-sm flex
gap-2"><ClipboardList size={16} className="text-teal-600"/>메모</label>
            <textarea rows={2} placeholder="공유 메모"
value={formData.note} onChange={(e) => setFormData({...formData, note:
e.target.value})} className="w-full p-2 border rounded text-sm"/>
            <textarea rows={2} placeholder="비공개 메모(강사용)"
value={formData.privateNote} onChange={(e) => setFormData({...formData,
```

```
privateNote: e.target.value})} className="w-full p-2 border
border-amber-200 bg-amber-50 rounded text-sm"/>
          </Card>
      )}

      <Button onClick={handleSaveRecord} className="w-full py-4 text-lg
mt-4 shadow-xl"><Save size={20} /> 선택한 항목 저장하기</Button>
    </div>
  );

  if (!user && view === 'login') return <div className="min-h-screen
bg-slate-50 flex justify-center"><div className="w-full max-w-md
bg-white shadow-xl min-h-screen"><LoginView /></div></div>;

  return (
    <div className="min-h-screen bg-slate-50 flex justify-center">
      <div className="w-full max-w-md bg-white shadow-xl min-h-screen
relative flex flex-col">
        <div className="bg-teal-600 p-4 pt-12 pb-6 text-white shadow-md
z-10 sticky top-0">
          <div className="flex items-center justify-between">
            <div><h1 className="text-2xl font-extrabold">Wellus</h1><p
className="text-teal-100 text-xs">{user?.role === 'admin' ? 'Master
Admin' : '웰러스 리커버리 시스템'}</p></div>
            <div className="w-8 h-8 bg-white/20 rounded-full flex
items-center justify-center backdrop-blur-sm"><Calendar size={16}
/></div>
          </div>
        </div>
        <div className="flex-1 overflow-y-auto p-4 bg-slate-50 pb-24">
          {view === 'list' && <MemberListView />}
          {view === 'admin_assign' && <AdminAssignView />}
          {view === 'detail' && <MemberDetailView />}
          {view === 'form' && <RecordFormView />}
        </div>
      </div>
    </div>
  );
}
```