```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=no">
    <title>FitTrack - Fitness Workout Tracker</title>
    <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }
        body {
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu,
sans-serif;
            background: #0f0f1a;
            min-height: 100vh;
            overflow-x: hidden;
        }
        #root {
            min-height: 100vh;
            display: flex;
            justify-content: center;
        }
        .app-container {
            width: 100%;
            max-width: 420px;
            min-height: 100vh;
            background: linear-gradient(180deg, #0f0f1a 0%, #1a1a2e 50%, #16213e 100%);
            position: relative;
            box-shadow: 0 0 60px rgba(0,0,0,0.5);
        }
        @media (max-width: 450px) {
            .app-container {
                max-width: 100%;
            }
        }
        .gradient-bg {
            min-height: 100vh;
```

```
    }
    .tab-bar {
        position: fixed;
        bottom: 0;
        left: 50%;
        transform: translateX(-50%);
        width: 100%;
        max-width: 420px;
        background: rgba(22, 33, 62, 0.95);
        backdrop-filter: blur(10px);
        border-top: 1px solid rgba(255,255,255,0.1);
        display: flex;
        justify-content: space-around;
        padding: 12px 0 30px;
        z-index: 100;
    }
    .tab-item {
        display: flex;
        flex-direction: column;
        align-items: center;
        cursor: pointer;
        padding: 8px 16px;
        border-radius: 16px;
        transition: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);
        opacity: 0.6;
    }
    .tab-item:hover {
        opacity: 1;
        background: rgba(255,107,107,0.1);
    }
    .tab-item.active {
        opacity: 1;
        background: linear-gradient(135deg, rgba(255,107,107,0.2) 0%, rgba(255,107,107,0.1)
100%);
    }
    .tab-icon {
        font-size: 22px;
        margin-bottom: 4px;
    }
    .tab-label {
        font-size: 11px;
        font-weight: 600;
        color: rgba(255,255,255,0.6);
    }
```

```
.tab-item.active .tab-label {
    color: #FF6B6B;
}
.content {
    padding: 20px;
    padding-bottom: 100px;
    min-height: 100vh;
}
.header {
    padding-top: 35px;
    margin-bottom: 25px;
}
.greeting {
    font-size: 34px;
    font-weight: 800;
    background: linear-gradient(135deg, #fff 0%, #FF6B6B 50%, #4ECDC4 100%);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
    background-clip: text;
}
.subtitle {
    font-size: 15px;
    color: rgba(255,255,255,0.5);
    margin-top: 4px;
}
.screen-title {
    font-size: 30px;
    font-weight: 800;
    color: #fff;
    margin-top: 35px;
    margin-bottom: 25px;
}
.section-title {
    fontSize: 18px;
    font-weight: 700;
    color: #fff;
    margin-bottom: 15px;
    margin-top: 20px;
}
.stats-grid {
    display: flex;
    flex-wrap: wrap;
    gap: 12px;
    margin-bottom: 25px;
```

```
}
.stat-card {
    flex: 1;
    min-width: calc(50% - 6px);
    background: rgba(22, 33, 62, 0.8);
    border-radius: 20px;
    padding: 20px 16px;
    text-align: center;
    border: 1px solid rgba(255,255,255,0.08);
    transition: all 0.3s ease;
}
.stat-card:hover {
    transform: translateY(-2px);
    border-color: rgba(255,107,107,0.3);
}
.stat-icon {
    font-size: 28px;
    margin-bottom: 8px;
}
.stat-value {
    font-size: 32px;
    font-weight: 800;
    background: linear-gradient(135deg, #FF6B6B 0%, #FF8E8E 100%);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
    background-clip: text;
}
.stat-label {
    font-size: 12px;
    color: rgba(255,255,255,0.5);
    margin-top: 4px;
    font-weight: 500;
}
.card {
    background: rgba(22, 33, 62, 0.8);
    border-radius: 20px;
    padding: 20px;
    margin-bottom: 15px;
    border: 1px solid rgba(255,255,255,0.08);
    transition: all 0.3s ease;
}
.card:hover {
    border-color: rgba(255,107,107,0.2);
}
```

```
.card-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 12px;
}
.card-title {
    font-size: 18px;
    font-weight: 700;
    color: #fff;
}
.card-badge {
    background: rgba(78, 205, 196, 0.2);
    color: #4ECDC4;
    padding: 6px 12px;
    border-radius: 20px;
    font-size: 12px;
    font-weight: 600;
}
.card-date {
    font-size: 12px;
    color: rgba(255,255,255,0.4);
    margin-bottom: 15px;
}
.card-stats {
    display: flex;
    justify-content: space-around;
    padding-top: 15px;
    border-top: 1px solid rgba(255,255,255,0.08);
}
.mini-stat {
    text-align: center;
}
.mini-stat-value {
    font-size: 24px;
    font-weight: 800;
    background: linear-gradient(135deg, #FF6B6B 0%, #FF8E8E 100%);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
    background-clip: text;
}
.mini-stat-label {
    font-size: 11px;
    color: rgba(255,255,255,0.4);
```

```
    margin-top: 2px;
}
.exercise-list {
    margin-top: 12px;
}
.exercise-item {
    display: flex;
    justify-content: space-between;
    padding: 12px 0;
    border-bottom: 1px solid rgba(255,255,255,0.05);
}
.exercise-item:last-child {
    border-bottom: none;
}
.exercise-name {
    font-size: 14px;
    font-weight: 600;
    color: #fff;
}
.exercise-details {
    font-size: 13px;
    color: rgba(255,255,255,0.5);
}
.btn {
    width: 100%;
    padding: 16px;
    border-radius: 16px;
    border: none;
    cursor: pointer;
    font-size: 16px;
    font-weight: 700;
    transition: all 0.3s ease;
}
.btn-primary {
    background: linear-gradient(135deg, #FF6B6B 0%, #FF8E8E 100%);
    color: #fff;
}
.btn-primary:hover {
    transform: translateY(-2px);
    box-shadow: 0 10px 30px rgba(255,107,107,0.3);
}
.btn-primary:disabled {
    opacity: 0.5;
    cursor: not-allowed;
```

```
    transform: none;
    box-shadow: none;
}
.btn-secondary {
    background: rgba(45, 45, 68, 0.8);
    color: #fff;
    border: 1px solid rgba(255,255,255,0.1);
}
.btn-secondary:hover {
    background: rgba(45, 45, 68, 1);
}
.input-group {
    margin-bottom: 20px;
}
.input-label {
    font-size: 14px;
    font-weight: 600;
    color: rgba(255,255,255,0.9);
    margin-bottom: 10px;
    display: block;
}
.input {
    width: 100%;
    background: rgba(22, 33, 62, 0.8);
    border-radius: 14px;
    padding: 16px 18px;
    font-size: 16px;
    color: #fff;
    border: 1px solid rgba(255,255,255,0.1);
    outline: none;
    transition: all 0.3s ease;
}
.input:focus {
    border-color: #FF6B6B;
    box-shadow: 0 0 0 3px rgba(255,107,107,0.1);
}
.input::placeholder {
    color: rgba(255,255,255,0.3);
}
.quick-select {
    display: flex;
    flex-wrap: wrap;
    gap: 8px;
    margin-bottom: 15px;
```

```
        }
        .quick-btn {
            padding: 8px 14px;
            border-radius: 20px;
            font-size: 13px;
            cursor: pointer;
            transition: all 0.3s ease;
            border: 1px solid rgba(255,255,255,0.1);
            background: rgba(22, 33, 62, 0.8);
            color: rgba(255,255,255,0.7);
        }
        .quick-btn:hover {
            border-color: rgba(255,255,255,0.3);
        }
        .quick-btn.active {
            background: linear-gradient(135deg, #FF6B6B 0%, #FF8E8E 100%);
            border-color: transparent;
            color: #fff;
        }
        .row {
            display: flex;
            gap: 15px;
        }
        .col {
            flex: 1;
        }
        .progress-card {
            background: linear-gradient(135deg, rgba(22, 33, 62, 0.9) 0%, rgba(22, 33, 62, 0.6)
100%);
            border-radius: 24px;
            padding: 25px;
            margin-bottom: 20px;
            border: 1px solid rgba(255,255,255,0.08);
        }
        .progress-title {
            font-size: 18px;
            font-weight: 700;
            color: #fff;
            margin-bottom: 20px;
        }
        .chart-container {
            height: 200px;
            margin: 20px 0;
        }
```

```
.achievement-row {
    display: flex;
    justify-content: center;
    gap: 30px;
    flex-wrap: wrap;
}
.achievement {
    text-align: center;
}
.achievement-icon {
    font-size: 36px;
    margin-bottom: 8px;
    display: block;
    animation: bounce 2s infinite;
}
@keyframes bounce {
    0%, 100% { transform: translateY(0); }
    50% { transform: translateY(-5px); }
}
.achievement-label {
    font-size: 11px;
    color: rgba(255,255,255,0.5);
}
.goal-card {
    background: rgba(22, 33, 62, 0.8);
    border-radius: 24px;
    padding: 25px;
    border: 1px solid rgba(255,255,255,0.08);
}
.goal-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}
.goal-title {
    font-size: 18px;
    font-weight: 700;
    color: #fff;
}
.goal-progress {
    display: flex;
    align-items: center;
    gap: 20px;
```

```
        }
        .goal-circle {
            width: 80px;
            height: 80px;
            border-radius: 50%;
            background: conic-gradient(#FF6B6B 0deg, #FF6B6B var(--progress),
rgba(255,255,255,0.1) var(--progress), rgba(255,255,255,0.1) 360deg);
            display: flex;
            align-items: center;
            justify-content: center;
        }
        .goal-circle-inner {
            width: 60px;
            height: 60px;
            border-radius: 50%;
            background: linear-gradient(135deg, #16213e 0%, #1a1a2e 100%);
            display: flex;
            align-items: center;
            justify-content: center;
        }
        .goal-percentage {
            font-size: 18px;
            font-weight: 800;
            color: #FF6B6B;
        }
        .goal-info {
            flex: 1;
        }
        .goal-text {
            font-size: 20px;
            font-weight: 700;
            color: #fff;
            margin-bottom: 5px;
        }
        .goal-subtext {
            font-size: 13px;
            color: rgba(255,255,255,0.5);
        }
        .empty-state {
            text-align: center;
            padding: 50px 20px;
        }
        .empty-icon {
            font-size: 60px;
```

```
    margin-bottom: 15px;
}
.empty-title {
    font-size: 20px;
    font-weight: 700;
    color: #fff;
    margin-bottom: 8px;
}
.empty-subtitle {
    font-size: 14px;
    color: rgba(255,255,255,0.5);
}
.modal-overlay {
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background: rgba(0,0,0,0.8);
    display: flex;
    justify-content: flex-end;
    z-index: 1000;
    animation: fadeIn 0.3s ease;
}
@keyframes fadeIn {
    from { opacity: 0; }
    to { opacity: 1; }
}
.modal-content {
    background: linear-gradient(180deg, #1a1a2e 0%, #16213e 100%);
    border-top-left-radius: 28px;
    border-top-right-radius: 28px;
    padding: 28px;
    padding-bottom: 40px;
    width: 100%;
    max-width: 420px;
    align-self: flex-end;
    animation: slideUp 0.3s ease;
}
@keyframes slideUp {
    from { transform: translateY(100%); }
    to { transform: translateY(0); }
}
.modal-header {
```

```
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 25px;
}
.modal-title {
    font-size: 24px;
    font-weight: 800;
    color: #fff;
}
.modal-close {
    background: rgba(255,255,255,0.1);
    border: none;
    color: rgba(255,255,255,0.6);
    font-size: 24px;
    cursor: pointer;
    width: 40px;
    height: 40px;
    border-radius: 12px;
    display: flex;
    align-items: center;
    justify-content: center;
}
.modal-close:hover {
    background: rgba(255,255,255,0.15);
    color: #fff;
}
.modal-actions {
    display: flex;
    gap: 12px;
    margin-top: 25px;
}
.modal-btn {
    flex: 1;
    padding: 16px;
    border-radius: 14px;
    border: none;
    cursor: pointer;
    font-size: 16px;
    font-weight: 700;
    transition: all 0.3s ease;
}
.modal-btn-cancel {
    background: rgba(45, 45, 68, 0.8);
```

```
    color: rgba(255,255,255,0.7);
}
.modal-btn-cancel:hover {
    background: rgba(45, 45, 68, 1);
}
.modal-btn-confirm {
    background: linear-gradient(135deg, #FF6B6B 0%, #FF8E8E 100%);
    color: #fff;
}
.modal-btn-confirm:hover {
    transform: translateY(-2px);
}
.exercise-added {
    display: flex;
    justify-content: space-between;
    align-items: center;
    background: rgba(22, 33, 62, 0.8);
    border-radius: 14px;
    padding: 16px;
    margin-bottom: 10px;
    border: 1px solid rgba(255,255,255,0.08);
}
.exercise-added-name {
    font-size: 16px;
    font-weight: 600;
    color: #fff;
}
.exercise-added-details {
    font-size: 13px;
    color: rgba(255,255,255,0.5);
    margin-top: 4px;
}
.exercise-added-remove {
    background: rgba(255,107,107,0.2);
    border: none;
    color: #FF6B6B;
    width: 30px;
    height: 30px;
    border-radius: 10px;
    cursor: pointer;
    font-size: 18px;
}
.exercise-added-remove:hover {
    background: rgba(255,107,107,0.3);
```

```
      }
    </style>
</head>
<body>
    <div id="root"></div>
    <script type="text/babel">
        const { useState, useEffect, useRef } = React;

        const commonExercises = [
            'Push-ups', 'Pull-ups', 'Squats', 'Deadlifts', 'Bench Press',
            'Lunges', 'Plank', 'Burpees', 'Dumbbell Rows', 'Shoulder Press',
            'Bicep Curls', 'Tricep Dips', 'Leg Press', 'Calf Raises', 'Crunches'
        ];

        function App() {
            const [currentScreen, setCurrentScreen] = useState('home');
            const [workouts, setWorkouts] = useState(() => {
                const saved = localStorage.getItem('fittrack_workouts');
                return saved ? JSON.parse(saved) : [];
            });
            const [showExerciseModal, setShowExerciseModal] = useState(false);
            const [currentExercise, setCurrentExercise] = useState({ name: '', sets: 3, reps: 12,
weight: 0 });
            const [workoutName, setWorkoutName] = useState('');
            const [workoutExercises, setWorkoutExercises] = useState([]);

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

            const saveWorkout = () => {
                if (workoutName && workoutExercises.length > 0) {
                    const workout = {
                        id: Date.now(),
                        name: workoutName,
                        date: new Date().toLocaleDateString(),
                        time: new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' }),
                        duration: Math.floor(Math.random() * 60) + 15,
                        exercises: workoutExercises
                    };
                    setWorkouts(prev => [...prev, workout]);
                    setWorkoutName('');
                    setWorkoutExercises([]);
                    setCurrentScreen('history');
```

```
    }
  };

  const addExercise = () => {
    if (currentExercise.name) {
      setWorkoutExercises(prev => [...prev, { ...currentExercise, id: Date.now() }]);
      setCurrentExercise({ name: '', sets: 3, reps: 12, weight: 0 });
      setShowExerciseModal(false);
    }
  };

  const removeExercise = (id) => {
    setWorkoutExercises(prev => prev.filter(e => e.id !== id));
  };

  const calculateStats = () => {
    const today = new Date().toDateString();
    const todayWorkouts = workouts.filter(w => new Date(w.date).toDateString() ===
today);
    const totalSets = todayWorkouts.reduce((acc, w) => acc + w.exercises.reduce((a, e)
=> a + e.sets, 0), 0);
    const totalSetsAll = workouts.reduce((acc, w) => acc + w.exercises.reduce((a, e) => a
+ e.sets, 0), 0);
    const totalMinutes = workouts.reduce((acc, w) => acc + w.duration, 0);
    const totalVolume = workouts.reduce((acc, w) => acc + w.exercises.reduce((a, e) => a
+ (e.sets * e.reps * e.weight), 0), 0);
    const exercises = [...new Set(workouts.flatMap(w => w.exercises.map(e =>
e.name)))];
    const weekWorkouts = workouts.filter(w => {
      const workoutDate = new Date(w.date);
      const today = new Date();
      const weekAgo = new Date(today.getTime() - 7 * 24 * 60 * 60 * 1000);
      return workoutDate >= weekAgo;
    }).length;

    return { todayWorkouts, totalSets, totalSetsAll, totalMinutes, totalVolume, exercises,
weekWorkouts };
  };

  const stats = calculateStats();
  const progressPercent = Math.min(Math.round((stats.weekWorkouts / 5) * 100), 100);

  return (
    <div className="app-container">
```

```
            <div className="gradient-bg">
              {currentScreen === 'home' && <HomeScreen stats={stats} workouts={workouts}
/>}
              {currentScreen === 'workout' && (
                <AddWorkoutScreen
                  workoutName={workoutName}
                  setWorkoutName={setWorkoutName}
                  workoutExercises={workoutExercises}
                  saveWorkout={saveWorkout}
                  showExerciseModal={showExerciseModal}
                  setShowExerciseModal={setShowExerciseModal}
                  currentExercise={currentExercise}
                  setCurrentExercise={setCurrentExercise}
                  addExercise={addExercise}
                  removeExercise={removeExercise}
                />
              )}
              {currentScreen === 'history' && <HistoryScreen workouts={workouts} />}
              {currentScreen === 'progress' && <ProgressScreen stats={stats}
progressPercent={progressPercent} workouts={workouts} />}
            </div>

            <div className="tab-bar">
              <div className={`tab-item ${currentScreen === 'home' ? 'active' : ''}`}
onClick={() => setCurrentScreen('home')}>
                <span className="tab-icon">ðŸ  </span>
                <span className="tab-label">Home</span>
              </div>
              <div className={`tab-item ${currentScreen === 'workout' ? 'active' : ''}`}
onClick={() => setCurrentScreen('workout')}>
                <span className="tab-icon">âž•</span>
                <span className="tab-label">Workout</span>
              </div>
              <div className={`tab-item ${currentScreen === 'history' ? 'active' : ''}`}
onClick={() => setCurrentScreen('history')}>
                <span className="tab-icon">ðŸ“‹</span>
                <span className="tab-label">History</span>
              </div>
              <div className={`tab-item ${currentScreen === 'progress' ? 'active' : ''}`}
onClick={() => setCurrentScreen('progress')}>
                <span className="tab-icon">ðŸ“Š</span>
                <span className="tab-label">Progress</span>
              </div>
            </div>
```

```
        </div>
      );
    }

    function HomeScreen({ stats, workouts }) {
      const today = new Date().toDateString();
      const todayWorkouts = workouts.filter(w => new Date(w.date).toDateString() === today);

      return (
        <div className="content">
          <div className="header">
            <h1 className="greeting">FitTrack</h1>
            <p className="subtitle">Your Personal Fitness Journey</p>
          </div>

          <div className="stats-grid">
            <div className="stat-card">
              <div className="stat-icon">ðŸ’ª</div>
              <div className="stat-value">{todayWorkouts.length}</div>
              <div className="stat-label">Today's Workouts</div>
            </div>
            <div className="stat-card">
              <div className="stat-icon">ðŸ“Š</div>
              <div className="stat-value">{stats.totalSets}</div>
              <div className="stat-label">Sets Today</div>
            </div>
            <div className="stat-card">
              <div className="stat-icon">ðŸ“…</div>
              <div className="stat-value">{stats.weekWorkouts}</div>
              <div className="stat-label">This Week</div>
            </div>
            <div className="stat-card">
              <div className="stat-icon">ðŸŽ¯</div>
              <div className="stat-value">{stats.exercises.length}</div>
              <div className="stat-label">Exercises</div>
            </div>
          </div>

          <h2 style={{ fontSize: '18px', fontWeight: '700', color: '#fff', marginBottom: '15px',
marginTop: '5px' }}>Recent Workouts</h2>
          {todayWorkouts.length === 0 ? (
            <div className="card" style={{ textAlign: 'center', padding: '40px' }}>
              <div style={{ fontSize: '50px', marginBottom: '15px' }}>ðŸ‹ï¸ </div>
```

```
                    <p style={{ fontSize: '18px', fontWeight: '600', color: '#fff', marginBottom: '8px'
}}>No workouts today yet!</p>
                    <p style={{ fontSize: '14px', color: 'rgba(255,255,255,0.5)' }}>Tap the + button
to start</p>
                </div>
            ) : (
                todayWorkouts.slice(0, 3).map((workout) => (
                    <WorkoutCard key={workout.id} workout={workout} />
                ))
            )}
        </div>
    );
}

function WorkoutCard({ workout }) {
    return (
        <div className="card">
            <div className="card-header">
                <span className="card-title">{workout.name}</span>
                <span className="card-badge">{workout.duration} min</span>
            </div>
            <p className="card-date">{workout.date} â€¢ {workout.time}</p>
            <div className="exercise-list">
                {workout.exercises.slice(0, 4).map((ex, i) => (
                    <div key={i} className="exercise-item">
                        <span className="exercise-name">{ex.name}</span>
                        <span className="exercise-details">{ex.sets}Ã—{ex.reps} @
{ex.weight}lbs</span>
                    </div>
                ))}
                {workout.exercises.length > 4 && (
                    <p style={{ fontSize: '12px', color: 'rgba(255,255,255,0.4)', marginTop: '8px' }}>
                        +{workout.exercises.length - 4} more exercises
                    </p>
                )}
            </div>
        </div>
    );
}

function AddWorkoutScreen({ workoutName, setWorkoutName, workoutExercises,
saveWorkout, showExerciseModal, setShowExerciseModal, currentExercise,
setCurrentExercise, addExercise, removeExercise }) {
    return (
```

```
        <div className="content">
          <h1 className="screen-title">New Workout</h1>

          <div className="input-group">
            <label className="input-label">Workout Name</label>
            <input
              type="text"
              className="input"
              placeholder="e.g., Morning Chest Day"
              value={workoutName}
              onChange={(e) => setWorkoutName(e.target.value)}
            />
          </div>

          <h2 style={{ fontSize: '16px', fontWeight: '700', color: '#fff', marginBottom: '15px' }}>
            Exercises ({workoutExercises.length})
          </h2>

          {workoutExercises.map((exercise) => (
            <div key={exercise.id} className="exercise-added">
              <div>
                <p className="exercise-added-name">{exercise.name}</p>
                <p className="exercise-added-details">{exercise.sets} sets Ã—
{exercise.reps} reps @ {exercise.weight}lbs</p>
              </div>
              <button className="exercise-added-remove" onClick={() =>
removeExercise(exercise.id)}>Ã—</button>
            </div>
          ))}

          <button className="btn btn-secondary" style={{ marginTop: '10px' }} onClick={()
=> setShowExerciseModal(true)}>
            + Add Exercise
          </button>

          <button
            className="btn btn-primary"
            style={{ marginTop: '25px' }}
            onClick={saveWorkout}
            disabled={!workoutName || workoutExercises.length === 0}
          >
            Save Workout
          </button>
```

```
            {showExerciseModal && (
              <div className="modal-overlay" onClick={() => setShowExerciseModal(false)}>
                <div className="modal-content" onClick={e => e.stopPropagation()}>
                  <div className="modal-header">
                    <h2 className="modal-title">Add Exercise</h2>
                    <button className="modal-close" onClick={() =>
setShowExerciseModal(false)}>Ã—</button>
                  </div>

                  <div className="input-group">
                    <label className="input-label">Exercise Name</label>
                    <input
                      type="text"
                      className="input"
                      placeholder="Select or type exercise"
                      value={currentExercise.name}
                      onChange={(e) => setCurrentExercise({ ...currentExercise, name:
e.target.value })}
                    />
                  </div>

                  <div style={{ marginBottom: '15px' }}>
                    <span style={{ fontSize: '12px', color: 'rgba(255,255,255,0.5)', display:
'block', marginBottom: '10px' }}>Quick Select:</span>
                    <div className="quick-select">
                      {commonExercises.map((ex) => (
                        <button
                          key={ex}
                          className={`quick-btn ${currentExercise.name === ex ? 'active' :
''}`}
                          onClick={() => setCurrentExercise({ ...currentExercise, name: ex
})}
                        >
                          {ex}
                        </button>
                      ))}
                    </div>
                  </div>

                  <div className="row">
                    <div className="col">
                      <label className="input-label">Sets</label>
                      <input
                        type="number"
```

```
                                    className="input"
                                    placeholder="3"
                                    value={currentExercise.sets}
                                    onChange={(e) => setCurrentExercise({ ...currentExercise, sets:
parseInt(e.target.value) || 0 })}
                                />
                            </div>
                            <div className="col">
                                <label className="input-label">Reps</label>
                                <input
                                    type="number"
                                    className="input"
                                    placeholder="12"
                                    value={currentExercise.reps}
                                    onChange={(e) => setCurrentExercise({ ...currentExercise, reps:
parseInt(e.target.value) || 0 })}
                                />
                            </div>
                        </div>

                        <div className="input-group" style={{ marginTop: '15px' }}>
                            <label className="input-label">Weight (lbs)</label>
                            <input
                                type="number"
                                className="input"
                                placeholder="0"
                                value={currentExercise.weight}
                                onChange={(e) => setCurrentExercise({ ...currentExercise, weight:
parseInt(e.target.value) || 0 })}
                            />
                        </div>

                        <div className="modal-actions">
                            <button className="modal-btn modal-btn-cancel" onClick={() =>
setShowExerciseModal(false)}>Cancel</button>
                            <button className="modal-btn modal-btn-confirm"
onClick={addExercise}>Add Exercise</button>
                        </div>
                    </div>
                </div>
            )}
        </div>
    );
}
```

```
function HistoryScreen({ workouts }) {
  return (
    <div className="content">
      <h1 className="screen-title">Workout History</h1>

      {workouts.length === 0 ? (
        <div className="empty-state">
          <div className="empty-icon">ðŸ“‹</div>
          <p className="empty-title">No workouts yet!</p>
          <p className="empty-subtitle">Start your fitness journey today</p>
        </div>
      ) : (
        [...workouts].reverse().map((workout) => (
          <div key={workout.id} className="card">
            <div className="card-header">
              <span className="card-title">{workout.name}</span>
              <span style={{ background: 'rgba(255,255,255,0.1)', padding: '6px 12px',
borderRadius: '20px', fontSize: '12px', color: 'rgba(255,255,255,0.6)' }}>
                {workout.date}
              </span>
            </div>
            <div className="card-stats">
              <div className="mini-stat">
                <div className="mini-stat-value">{workout.exercises.length}</div>
                <div className="mini-stat-label">Exercises</div>
              </div>
              <div className="mini-stat">
                <div className="mini-stat-value">{workout.exercises.reduce((acc, e)
=> acc + e.sets, 0)}</div>
                <div className="mini-stat-label">Sets</div>
              </div>
              <div className="mini-stat">
                <div className="mini-stat-value">{workout.duration}</div>
                <div className="mini-stat-label">Minutes</div>
              </div>
            </div>
            <div className="exercise-list" style={{ marginTop: '15px', paddingTop:
'15px', borderTop: '1px solid rgba(255,255,255,0.08)' }}>
              {workout.exercises.map((ex, i) => (
                <div key={i} className="exercise-item">
                  <span className="exercise-name">{ex.name}</span>
                  <span className="exercise-details">{ex.sets}Ã—{ex.reps} @
{ex.weight}lbs</span>
```

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

function ProgressScreen({ stats, progressPercent, workouts }) {
    const chartRef = useRef(null);
    const chartInstance = useRef(null);

    useEffect(() => {
        if (chartRef.current && workouts.length > 0) {
            const ctx = chartRef.current.getContext('2d');

            if (chartInstance.current) {
                chartInstance.current.destroy();
            }

            const last7Days = [];
            const counts = [];
            for (let i = 6; i >= 0; i--) {
                const date = new Date();
                date.setDate(date.getDate() - i);
                last7Days.push(date.toLocaleDateString('en-US', { weekday: 'short' }));
                const dayWorkouts = workouts.filter(w => {
                    const workoutDate = new Date(w.date);
                    return workoutDate.toDateString() === date.toDateString();
                }).length;
                counts.push(dayWorkouts);
            }

            chartInstance.current = new Chart(ctx, {
                type: 'bar',
                data: {
                    labels: last7Days,
                    datasets: [{
                        label: 'Workouts',
                        data: counts,
                        backgroundColor: 'rgba(255, 107, 107, 0.8)',
                        borderRadius: 8,
```

```
                borderSkipped: false,
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            plugins: {
                legend: { display: false }
            },
            scales: {
                x: {
                    grid: { display: false },
                    ticks: { color: 'rgba(255,255,255,0.5)', font: { size: 11 } }
                },
                y: {
                    grid: { color: 'rgba(255,255,255,0.05)' },
                    ticks: { color: 'rgba(255,255,255,0.5)', font: { size: 11 } },
                    beginAtZero: true
                }
            }
        }
    });
  }

  return () => {
    if (chartInstance.current) {
      chartInstance.current.destroy();
    }
  };
}, [workouts]);

return (
  <div className="content">
    <h1 className="screen-title">Progress</h1>

    <div className="progress-card">
      <h3 className="progress-title">Your Achievements</h3>
      <div className="achievement-row">
        {stats.weekWorkouts >= 1 && (
          <div className="achievement">
            <span className="achievement-icon">ðŸŽ¯</span>
            <span className="achievement-label">First Workout</span>
          </div>
        )}
```

```
                {stats.weekWorkouts >= 5 && (
                  <div className="achievement">
                    <span className="achievement-icon">ðŸ”¥</span>
                    <span className="achievement-label">5 Workouts</span>
                  </div>
                )}
                {stats.weekWorkouts >= 10 && (
                  <div className="achievement">
                    <span className="achievement-icon">ðŸ’Ž</span>
                    <span className="achievement-label">10 Workouts</span>
                  </div>
                )}
                {stats.weekWorkouts >= 20 && (
                  <div className="achievement">
                    <span className="achievement-icon">ðŸ†</span>
                    <span className="achievement-label">20 Workouts</span>
                  </div>
                )}
              </div>
            </div>

            <h2 style={{ fontSize: '18px', fontWeight: '700', color: '#fff', marginBottom: '15px'
}}>Weekly Activity</h2>
            <div className="card" style={{ padding: '20px' }}>
              <div className="chart-container">
                <canvas ref={chartRef}></canvas>
              </div>
            </div>

            <div className="goal-card">
              <div className="goal-header">
                <h3 className="goal-title">Weekly Goal</h3>
              </div>
              <div className="goal-progress">
                <div className="goal-circle" style={{ '--progress': `${progressPercent *
3.6}deg` }}>
                  <div className="goal-circle-inner">
                    <span className="goal-percentage">{progressPercent}%</span>
                  </div>
                </div>
                <div className="goal-info">
                  <p className="goal-text">{stats.weekWorkouts} / 5 workouts</p>
                  <p className="goal-subtext">
```

```
                        {stats.weekWorkouts >= 5 ? 'ðŸŽ‰ Goal reached!' : `${5 -
stats.weekWorkouts} more to go`}
                    </p>
                  </div>
                </div>

                <div style={{ marginTop: '20px' }}>
                  <h2 style={{ fontSize: '18px', fontWeight: '700', color: '#fff', marginBottom: '15px'
}}>All Stats</h2>
                  <div className="stats-grid" style={{ gap: '10px' }}>
                    <div className="stat-card" style={{ minWidth: 'calc(50% - 5px)' }}>
                      <div className="stat-icon">ðŸ'ª</div>
                      <div className="stat-value">{stats.weekWorkouts}</div>
                      <div className="stat-label">Total Workouts</div>
                    </div>
                    <div className="stat-card" style={{ minWidth: 'calc(50% - 5px)' }}>
                      <div className="stat-icon">ðŸ“Š</div>
                      <div className="stat-value">{stats.totalSetsAll}</div>
                      <div className="stat-label">Total Sets</div>
                    </div>
                    <div className="stat-card" style={{ minWidth: 'calc(50% - 5px)' }}>
                      <div className="stat-icon">â ±ï¸ </div>
                      <div className="stat-value">{stats.totalMinutes}</div>
                      <div className="stat-label">Minutes</div>
                    </div>
                    <div className="stat-card" style={{ minWidth: 'calc(50% - 5px)' }}>
                      <div className="stat-icon">ðŸ ‹ï¸ </div>
                      <div className="stat-value">{(stats.totalVolume / 1000).toFixed(1)}k</div>
                      <div className="stat-label">Volume (lbs)</div>
                    </div>
                  </div>
                </div>
              </div>
            );
        }

        const root = ReactDOM.createRoot(document.getElementById('root'));
        root.render(<App />);
    </script>
</body>
</html>
```