MOBILE AND UBIQUITOUS COMPUTING

UNIVERSITY OF BIRMINGHAM

Smartfit(Smart Dumbell and Resistance Band)

Author: Yash Nawghare

ID: 2671357

Course Lead: Chris Baber

Instructor:

April 2024

Abstract:

Accurate and optimized workout recommendations are very important for improving physical health and to achieve the fitness goals. We can see the fitness industry is booming. Current fitness solutions like static weight dumbbells and resistance bands, do not provide feedback and do not adapt to the user's changing strength level and their efficiency. Here a very new idea of Internet of Things (IoT) concept is introduced to solve these limitations through smart fitness devices: adaptive dumbbells and a sensor-equipped resistance band.

The introduced Smart dumbbells and Resistance band are embedded with sensors that monitor user performance in real-time. The resistance band assess the force exerted and the dumbbells with sensors assess speed and the form of movements and gives feedback and gives recommendations in adjusting resistance based on the user's capability and failure levels. This way not only provides workouts are performed with better impact but also reduces the risk of injury due to improper weight usage or poor form.

This report explains a thorough technical concept of smart equipment for optimize user efforts while working out. The context widget and the mobile app improves the user experience making the workout enjoyable while getting a thorough analytics of metrics of workout. The widgets with the mobile app take account reactions and the interactions from system user and environment.

The data from the user and the Iot devices is used for the predicting future strength and configure workout and give advice for improving form while workout to reduce the risks of injury. This report also shows understanding how the users will react with the system. This aspect of reports takes consideration of different stakeholders and make sure that the user experience is enhanced.

Nowadays the necessity of security and privacy is on very high demand and is very important. Encrypting data, protecting the device from getting hacked is also considered very vital. At the same time mitigating the attack and incident response is also considered.

At the end the evaluation protocol is considered which not only provides a way to help the fitness system is effective, and user-friendly but also helps potential risks to be responded and treated with immediate strategies.

Application Domain and Problem:

In the domain of health and fitness technology, the use of smart equipment into home workouts has gaps in traditional fitness routines. Traditional home gym equipment does not interact and adapt, which is not good for meeting the user goal. This can lead to not much effective workouts and a very high risk of injuries due to bad technique. Traditional equipment do not offer feedback or adjustments based on user performance, which are important for user progress and effective workout routine

The smart fitness system with IoT-enabled dumbbells and resistance bands revolutionizes home workouts by simultaneously adjusting to user inputs and providing real-time feedback on performance. This technology helps exercises are performed with correct form, enhancing safety and efficiency. It seems far more better over current fitness apps and normal gym equipment that are behind in direct integration with workout devices and fail to improve user experience. Usually while using traditional dumbbells we are prone to use more or less weight which both are not good for the user

Stakeholders of this creative solution include people who work out at home seeking more engaging and personal workout routines, fitness trainers who require detailed performance data to modify coaching sessions, and healthcare providers who can use exercise data for better patient health. This approach not only helps user interaction and workout effectiveness but also promotes better performance and progress in fitness through a modified , data-driven experience. By considering these specific demands, the smart fitness system provides a much better alternative to traditional fitness tools , showing a very good progress in advancement in home and commercial gym equipment.

The use of such technology is very important in the personal fitness area, where the lack of personalized guidance can slow down progress in healing and increase injury risks. By using sensors, actuators, and smart feedback mechanisms, these IoT devices provide directly to the needs of the injured or disabled people. Also, they can be used for rehabilitation centers where personalized therapy are very essential for patient's recovery.

The concept introduced in this report has a upper hand over existing traditional fitness equipment by providing an intelligent, adaptive workout routine that will help user to progress and recover. Usually while using traditional dumbbells we tend to use more or less weight which both can lead to either slow the progress or lead to injury and often does not have the capability to track progress effectively .This IoT solution offers a seamless, data-driven exercise experience.

Context Widgets:

In SmartFIT Context Widgets play vital role .Following are the widgets:

Detailed Functionality of Individual Context Widgets:

1.User Engagement Widget:

- **Attributes:** This widget uses user-related data like fitness goals, preferences, progress.
- **Callback:** callback function of this widget is triggered when the user interacts with the app, updates their preferences, or completes a workout session.

1. **Purpose:** The purpose is to customize the exercise routine for each user wrt on their goals and preferences. It adjusts the difficulty level of each session, suggests suitable workouts aligned with the user's progress, and provides modified recommendations.

2. **Environmental Awareness Widget:**
   - **Attributes:** widget continuously monitors environmental conditions such as temperature, humidity, and noise levels using built-in sensors.
   - **Callback:** function is triggered when significant changes in environmental conditions are detected.
   - **Purpose:** Dynamically adjusts the workout routine to optimize user comfort and workout efficiency. For e.g. it may suggest easier workouts on hot days or increase rest time between sets.

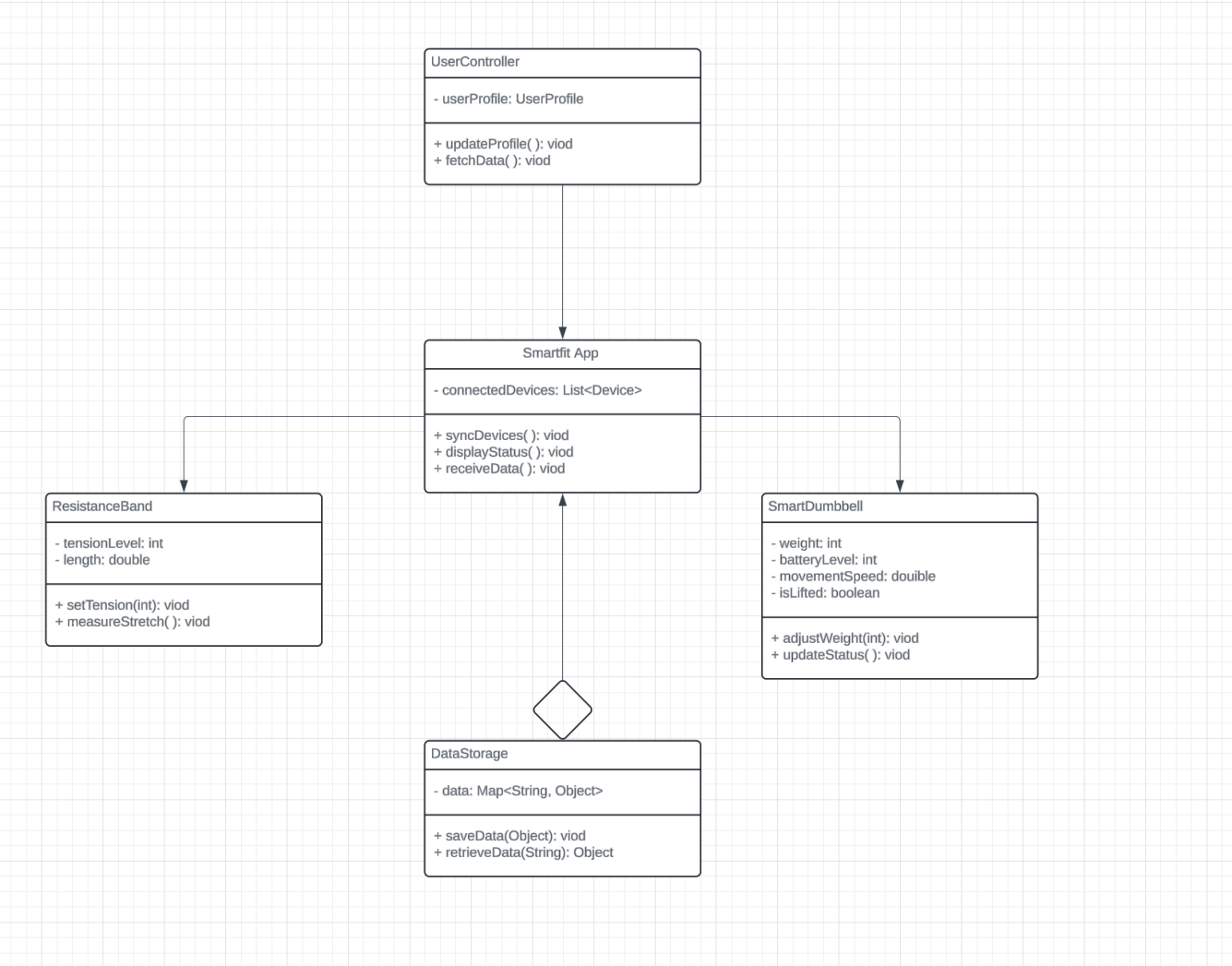
3. **Activity Analysis Widget:**
   - **Attributes:** Tracks the type, intensity, and quality of each workout session using motion sensors integrated into the fitness equipment.
   - **Callback:** function provides instant form and technique feedback to the user
   - **Purpose:** to improve workout performance and prevent injuries by providing real-time feedback on form and technique. analyzes user movements, identifies any deviations from proper form, and offers corrective guidance.
4. **Health Sync Widget:**
   - **Attributes:** retrieves vital signs such as blood pressure, heart rate, and sweating levels from wearable health monitors**.**
   - **Callback:** function adjusts the intensity of the workout in real-time based on the user's current health status.
   - **Purpose:** ensures user safety during workouts by optimizing the intensity level according to the user's health metrics,adjusts the workout parameters to prevent overexertion and minimize the risk of adverse health events.
5. **Connectivity and Data Management Widget:**
   - **Attributes:** manages data transfer between fitness equipment, user devices, and the cloud.
   - **Callback:** function handles data synchronization and facilitates remote access to user data.
   - **Purpose:** enables seamless integration between different platforms and devices, ensuring that user data is synchronized and accessible across all channels. It supports features such as remote progress tracking, virtual coaching sessions, and social sharing of achievements

Source Widget	Destination Widget	Type of Interaction	Description of Interaction
User Profile Widget	Activity Recognition Widget	Data Transfer	Sends updated user profile data to customize activity recognition and adaptation based on user's fitness goals and history.
Environmental Sensor Widget	Activity Recognition Widget	Environmental Data Input	Provides real-time environmental data (e.g., temperature, humidity) to adjust the exercise settings for safety and comfort.
Environmental Sensor Widget	Health Monitoring Widget	Environmental Data Input	Sends ambient condition data to help adjust health monitoring thresholds and warnings.
Activity Recognition Widget	User Profile Widget	Feedback Loop	Returns performance data to update the user profile with new insights on user's exercise habits and achievements.
Health Monitoring Widget	User Profile Widget	Health Data Feedback	Updates the user profile with real-time health metrics from wearables to tailor future workouts and health advice.
All Widgets	Connectivity Widget	Synchronization and Data Sharing	Ensures continuous synchronization of data across the system, updates settings, and facilitates cloud data management.

Interaction Among Context Widgets

- **User Engagement and Activity Analysis Widgets:** These widgets work together to modify workouts according to the user's performance and preferences. The Widget provides input on the user's performance, which the User Engagement Widget uses to make the personalized workout plans better.
- **Environmental Awareness and Health Sync Widgets:** These widgets work together to adjust the workout environment and intensity based on both external conditions and the user's physiological data. For eg, if the Environmental Awareness Widget detects a high temperature, the Health Sync Widget might adjust the intensity to prevent overexerting, based on current heart rate and other data**.**
- **Connectivity and Data Management Widget with All Widgets:** Acts as a hub, making sure all important data collected by other widgets is properly synchronized and used across devices. This widget supports all updates and enables seamless user experiences, from adjusting workouts to tracking long-term progress.

UML Class Diagram:

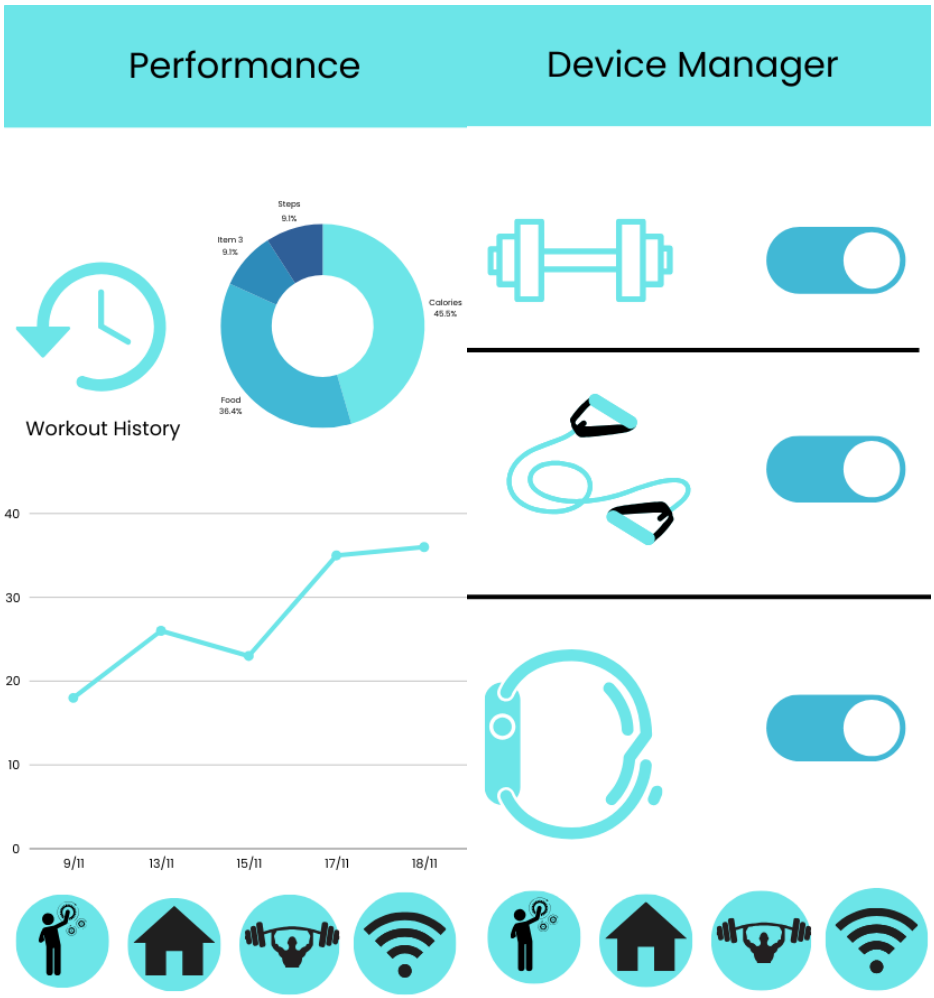


1. **Smartfit App**:Behaves like the central hub for the system. It manages a list of connected devices and includes functions to synchronize these devices (**syncDevices()**), display their status (**displayStatus()**), and receive data from them (**receiveData()**). This setup allows for centralized control and monitoring of all IoT devices within the app.
2. **UserController**: Administrates user-specific data and interactions. It contains methods to update user profiles (**updateProfile()**) and fetch data (**fetchData)**
3. **SmartDumbbell and ResistanceBand**: These classes represent the physical IoT devices. The SmartDumbbell includes attributes like weight, battery level, and a boolean to check if it is lifted, with methods to adjust its weight (**adjustWeight()**) and update its status (**updateStatus()**). The ResistanceBand tracks tension level and length, with capabilities to set the tension (**setTension()**) and measure stretch (**measureStretch()**), facilitating interactive and adaptive workout experiences.
4. **DataStorage**: This component handles data management within the system. It stores data in a key-value pair format and provides methods to save (**saveData()**) and retrieve (**retrieveData()**) data, ensuring that user and device data are effectively captured and available for processing and analytics.

Smartphone App:

The Smartfit app is made to improve fitness routines through its user-friendly interface, spanning several key screens each modify for specific functionality. The Dashboard or Home Screen greets users with personalized messages based on the time of day and provides quick access to necessary fitness metrics through a simple but informative layout. Users can navigate to the Workout Configuration Screen using the bottom navigation bar. This screen allows users to set up their workouts by adjusting parameters like weight and resistance and setting a timer for workout duration using dropdown menus and sliders. The Performance Analytics Screen gives insights through line charts and summary cards that track workout trends and statistics also a history list detailing past activity. The Device Management Screen helps user to connect, manage, and sync IoT devices, making sure integration and data consistency across their fitness equipment. The bottom navigation bar remains constant in all different screens, helping smooth switch and continuous access to all app functionalities.

UI Design:


Performance
Device Manager
Steps
9.1%
Item 3
9.1%
Calories
45.5%
Food
36.4%
Workout History
40
30
20
10
0
9/11
13/11
15/11
17/11
18/11

Overview of App Views

Smartfit is designed for unique features smart lifting:

1. **Dashboard or Home Screen**
2. **Workout Configuration Screen**
3. **Performance Analytics screen**

1. Dashboard or the Home screen

- It will have a greeting section (personalized based on the time of the day).
- Quick status which uses 'The row' and 'Stat Card' to display fitness metrics..
- It will have a Bottom navigation Bar which can be used for navigating in between different sections of the app.

```
class DashboardScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Dashboard')),
      body: SingleChildScrollView(
        child: Column(
          children: [
            Padding(
              padding: const EdgeInsets.all(16.0),
              child: Text('Good Morning, User!', style:
Theme.of(context).textTheme.headline4),
            ),
            QuickStats(),
            ElevatedButton(
              onPressed: () {
                // Navigate to the workout start screen
              },
              child: Text('Start Workout'),
            ),
            ElevatedButton(
              onPressed: () {
                // Navigate to the history screen
              },
              child: Text('View History'),
            ),
          ],
        ),
      ),
    );
  }
}
```

2. Workout Configuration Screen:

.

- Preference Filters: It will have a Dropdown button which allows list of workouts.
- Sliders having adjust parameters like weight and resistance dynamically.
- Timer: will have a timer setting so that the user can put a time for their workout duration. Start Users establish their criteria, and the system generates optimized routes, considering factors such as speed, convenience, and user choice.

```dart
class WorkoutConfigurationScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Workout Configuration')),
      body: SingleChildScrollView(
        child: Column(
          children: [
            DropdownButton<String>(
              items: <String>['Strength', 'Cardio',
'Flexibility'].map((String value) {
                return DropdownMenuItem<String>(
                  value: value,
                  child: Text(value),
                );
              }).toList(),
              onChanged: (_) {},
            ),
            Slider(
              min: 0,
              max: 100,
              divisions: 20,
              label: '50kg',
              onChanged: (double value) {},
              value: 50,
            ),
            TextField(
              decoration: InputDecoration(
                labelText: 'Duration (minutes)',
              ),
            ),
            ElevatedButton(
              onPressed: () {
                // Start the workout
              },
              child: Text('Start Workout'),
            ),
          ],
        ),
      ),
    );
  }
}
```

3. Performance Analytics screen:

- Line chart which displays the trend in a graph format.
- Summary Cards which provide quick view in the workout statistics in detail.
- History list shows the details about past workouts.

```
class PerformanceAnalyticsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Performance Analytics')),
      body: SingleChildScrollView(
        child: Column(
          children: [
            LineChart(), // Custom widget for displaying line chart
            Card(child: Text('Average Calories Burned: 500')),
            ListView(
              children: [ListTile(title: Text('Workout on Monday'))],
              shrinkWrap: true,
            ),
          ],
        ),
      ),
    );
  }
}
```

4. Device Managaement Screen:

- Connected Devices: List of all IOT devices and options to see details and disconnect.
- Sync Device Button: Syncs data in-between app and the Iot Device
- Add, Pairing new devices.

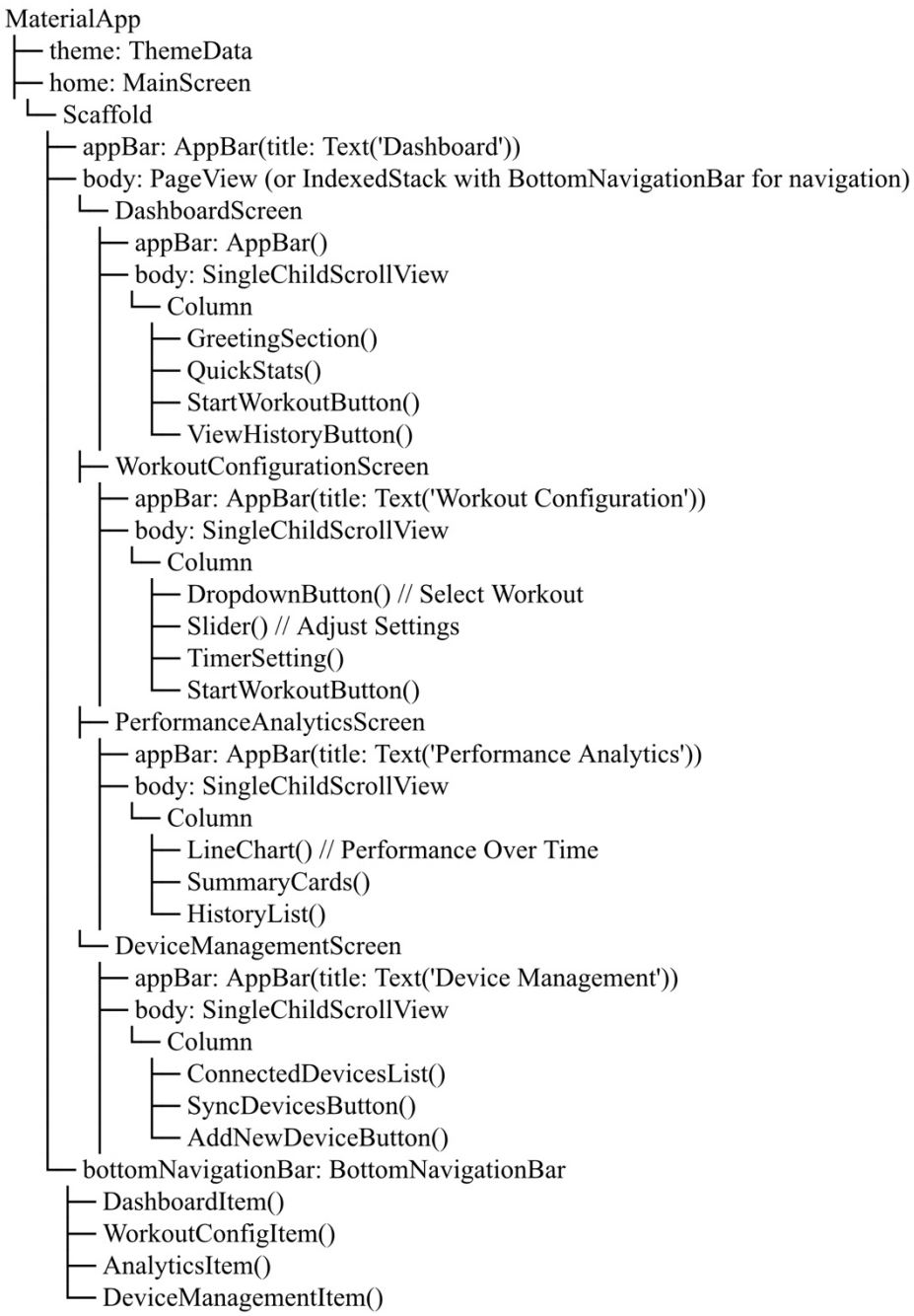
```
class DeviceManagementScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Device Management')),
      body: SingleChildScrollView(
        child: Column(
          children: [
            ListTile(title: Text('Smart Dumbbell - Connected')),
            ElevatedButton(
              onPressed: () {
                // Sync devices
              },
              child: Text('Sync Devices'),
            ),
            ElevatedButton(
              onPressed: () {
                // Add a new device
              },
              child: Text('Add New Device'),
            ),
          ],
        ),
      ),
    );
  }
}
```

5. Bottom Navigator Bar:

- Switching between these main screens.
- This widget remains at the bottom of the Main Screen, allowing for consistent navigation

```
BottomNavigationBar(
items: const <BottomNavigationBarItem>[
BottomNavigationBarItem(icon: Icon(Icons.dashboard), label:
'Dashboard'),
BottomNavigationBarItem(icon: Icon(Icons.settings), label: 'Workout
Config'),
BottomNavigationBarItem(icon: Icon(Icons.analytics), label:
'Analytics'),
BottomNavigationBarItem(icon: Icon(Icons.devices), label: 'Devices'),
],
onTap: (index) {
// Update the PageView controller or IndexedStack index
},
)
```

The Widget Tree:



Explanation of the Widget Tree

1) **MaterialApp**: Root widget of application. It sets up the material design style of app and provides features like navigation.
   a) **theme**: It defines the overall theme data for the entire app.
   b) **home: DashboardScreen**: sets the initial route of the application to the DashboardScreen.
2) **DashboardScreen**: main screen of application.
3) **Scaffold**: Provides a basic layout structure for the screen.
   i) **appBar: AppBar()**: Displays an app bar at the top of the screen.
   ii) **body: SingleChildScrollView**: allows scrolling if the content doesn't fit on the screen.
      (1) **Column**: arranges its children vertically.
         (a) **GreetingSection()**: displays a greeting message.
         (b) **QuickStats()**: shows quick statistics.
         (c) **StartWorkoutButton()**: button to start a workout.
         (d) **ViewHistoryButton()**: button to view workout history.
   iii) **bottomNavigationBar:** displays a bottom navigation bar.
      (1) **DashboardItem()**: item for dashboard.
      (2) **WorkoutConfigItem()**: item for workout configuration.
      (3) **AnalyticsItem()**: item for analytics.
      (4) **DeviceManagementItem()**: Item for device management.
4) **WorkoutConfigurationScreen**: Screen for configuring workouts.
5) **Scaffold**: provides layout structure.
   i) **appBar: AppBar()**: displays an app bar.
   ii) **body: SingleChildScrollView**: allows scrolling.
      (1) **Column**: arranges its children vertically.
         (a) **DropdownButton()**: allows selecting a workout.
         (b) **Slider()**: adjusts workout settings.
         (c) **TimerSetting()**: sets workout timer.
         (d) **StartWorkoutButton()**: button to start the workout.
6) **PerformanceAnalyticsScreen**: screen for performance analytics.
7) **Scaffold**: provides layout structure.
   i) **appBar:**displays an app bar.
   ii) **body: SingleChildScrollView**: allows scrolling.
      (1) **Column**: arranges its children vertically.
         (a) **LineChart()**: displays performance over time.
         (b) **SummaryCards()**: shows summary of performance.
         (c) **HistoryList()**: displays workout history.
8) **DeviceManagementScreen**: screen for managing connected devices.
9) **Scaffold**: provides layout structure.
   i) **appBar:** displays an app bar.
   ii) **body: SingleChildScrollView**: allows scrolling.
      (1) **Column**: arranges its children vertically.
         (a) **ConnectedDevicesList()**: lists connected devices.
         (b) **SyncDevicesButton()**: button to sync devices.
         (c) **AddNewDeviceButton()**: button to add a new device.

W3C Thing Description for Smartfit

Following the W3C Thing Description, these 'things' have various features and interactions that enable the Smartfit to improve the workout routine.

Smart Dumbbell

1. **Properties:**
    - **Weight:** Users can both view and set the weight of the dumbbell, important for adaptive training**.**
    - **Battery Level:** Allows monitoring of the device's power status to make sure it is charged sufficiently for workouts.
    - **Motion:** Detailed motion data is captured including acceleration and gyroscope readings, vital for analysing the quality of each exercise**.**
    - **Lifting Speed:** Derived from motion data, providing insights into the user's workout intensity and form.
2. **Actions:**
    - **Adjust Weight:** This allows users or the system to set the weight within a specified range dynamically based on the workout plan or real-time performance feedback.
3. **Events:**
    - **Weight Changed:** Notifies the system or user whenever the weight setting is adjusted which is useful for logging changes and analysing workout trends.
    - **Motion Detected:** Important for real-time feedback on exercise form and for initiates alerts if incorrect motions are detected.


```
{
"id": "urn:dev:wot:smart-fitness:smart-dumbbell",
"title": "Smart Dumbbell",
"description": "An IoT-enabled dumbbell that allows weight adjustment, monitors usage, and tracks
motion and speed of lifting.",
"properties": {
"weight": {
"type": "integer",
"description": "Current weight of the dumbbell in kilograms",
"writable": true,
"observable": true
},
"batteryLevel": {
"type": "integer",
"description": "Battery level of the dumbbell",
"writable": false,
"observable": true
},
"motion": {
"type": "object",
"description": "Motion data including acceleration and orientation",
"properties": {
"acceleration": {
"type": "array",
"items": {
"type": "number"
},
"description": "Acceleration vector x, y, z in m/s²"
},
"gyroscope": {
"type": "array",
"items": {
"type": "number"
},
"description": "Angular velocity vector x, y, z in degrees/sec"
}
},
"writable": false,
"observable": true
},
"liftingSpeed": {
"type": "number",
"description": "Speed of lifting calculated from motion data in m/s",
"writable": false,
"observable": true
}
},
"actions": {
"adjustWeight": {
"description": "Adjusts the weight of the dumbbell",
"input": {
"type": "integer",
"minimum": 1,
"maximum": 50
}
}
},
"events": {
"weightChanged": {
"description": "Emitted when the weight of the dumbbell changes"
},
"motionDetected": {
"description": "Emitted when new motion data is recorded"
}
}
}
```

Arduino for Smart Dumbell

1. **Initialization:**
   - Initializing the I2C communication via Wire.begin() and setting up serial communication for debugging and monitoring outputs.
   - The sensor initialization and connection test provide a good feedback mechanism to troubleshoot or verify sensor functionality during setup.
2. **Data Collection:**
   - function getMotion6() is used to gather both accelerometer and gyroscope data. This is important for tracking the movement and placement of the dumbbell during exercises.
3. **Lifting Speed Calculation:**
   - The lifting speed uses the magnitude of the acceleration vector, which helps understanding the movement intensity

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 sensor;

void setup() {
Wire.begin();
Serial.begin(9600);
sensor.initialize();
if (!sensor.testConnection()) {
Serial.println("MPU6050 connection failed");
} else {
Serial.println("MPU6050 connection successful");
}
}

void loop() {
int16_t ax, ay, az, gx, gy, gz;
sensor.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

// Calculate lifting speed - this is a simplified version for
demonstration.
// Real calculation would need integration over time and consideration of
lift dynamics.
double liftingSpeed = sqrt(ax * ax + ay * ay + az * az);

Serial.print("Acceleration (x, y, z): ");
Serial.print(ax); Serial.print(" "); Serial.print(ay); Serial.print(" ");
Serial.println(az);
Serial.print("Gyroscope (x, y, z): ");
Serial.print(gx); Serial.print(" "); Serial.print(gy); Serial.print(" ");
Serial.println(gz);
Serial.print("Estimated Lifting Speed: ");
Serial.println(liftingSpeed);

delay(100);
}
```

Smart Resistance Bands:

1. **Properties:**
    - **Tension:** Allows both monitoring and setting the tension level of the resistance band,
2. **Actions:**
    - **Set Tension:** facilitates the adjustment of the band's tension, which is essential for dynamically changing workout intensities based on the user's performance and preferences.
3. **Events:**
    - **Tension Changed:** Notifies other components of the system like the app when the tension has been modified, which can be used for logging changes and triggering other actions or informing the user.

```
{
"id": "urn:dev:wot:smart-fitness:resistance-band",
"title": "Smart Resistance Band",
"description": "A resistance band that adjusts tension based on user
settings.",
"properties": {
"tension": {
"type": "integer",
"description": "Current tension level of the band",
"writable": true,
"observable": true
}
},
"actions": {
"setTension": {
"description": "Sets the tension level of the resistance band",
"input": {
"type": "integer",
"minimum": 10,
"maximum": 100
}
}
},
"events": {
"tensionChanged": {
"description": "Emitted when the tension level changes"
}
}
}
```

Arduino for Smart Resistance Bands

1. **Servo Library:** Utilizes the Servo library to control a servo motor
2. **Servo Motor Setup:** The servo motor is attached to pin 9, establishing its connection to the microcontroller.
3. **Tension Control:**
    - Initial Setting: Sets an initial current Tension value to start with a known state.
    - Tension Adjustment Function: Includes a function set Tension to adjust the band's tension by mapping the desired tension level to a servo angle. It also checks if the wanted tension is in the valid range before applying it.
    - Serial Feedback: Outputs the new tension to the serial monitor for debugging and monitoring, which is essential for real-time feedback during development.
4. **Loop Function:**
    - The loop demonstrates a static example of setting the tension to 50, with a 10-second interval between adjustments.

```
#include <Servo.h>

Servo tensionAdjuster;
int currentTension = 10; // Example starting tension

void setup() {
tensionAdjuster.attach(9); // Attach servo motor to pin 9
Serial.begin(9600);
}

void setTension(int tension) {
if (tension > 100 || tension < 10) return; // Tension bounds check
int angle = map(tension, 10, 100, 0, 180); // Map tension to servo angle
tensionAdjuster.write(angle);
currentTension = tension;
Serial.println("Tension adjusted to: " + String(currentTension));
}

void loop() {
// Set tension example (normally, this would be triggered by an action from
the app)
setTension(50); // Set tension to 50
delay(10000); // Adjust tension every 10 seconds for demonstration
}
```

Health Monitoring Wristband

- **Interaction Affordances:**
  - **Properties:** heartRate, oxygenLevel, providing real-time health data**.**
  - **Actions:** None in this case, as the wristband is mostly for monitoring**.**
  - **Events:** health Alert, triggered if any readings go beyond normal thresholds.
- **Schemas for Data Exchange**: how health data is structured and communicated.
- **Security Definitions:** ensures that health data is transmitted securely.
- **Web Links:** links to health tips or user support.

```
{
"id": "healthWristband1",
"name": "Health Monitoring Wristband",
"properties": {
"heartRate": {
"type": "integer",
"unit": "beats per minute",
"observable": true
},
"oxygenLevel": {
"type": "integer",
"unit": "percentage",
"observable": true
}
},
"events": {
"health Alert": {
"output": {
"type": "string",
"description": "Alert on abnormal health reading"
}
}
}
}
```

Learning and Intelligence

Data Collection

Public Dataset vs. Custom Data Collection : It's not possible to find a public record that directly matches how smart dumbbells and resistance bands are used. So, my plan is to collect custom data by having people use the gadgets themselves. Sensors built into the dumbbells and tension bands will collect data that includes:

- **Weight Adjustments:** frequency and range of weight changes.
- **Motion and Speed:** movements during exercise sessions**.**
- **User Feedback:** post-workout feedback regarding comfort and effectiveness.
- **Exercise Outcomes:** repetitions, set completion, and consistency of exercise form.

Data Collection Mechanism: Data will be sent to the main server via Wi-Fi for each time that the equipment is used. Users will interact with a mobile app that also give the feedback and information about their workout preferences, which is very helpful for making personalised modifications.

Challenges in Training the Model

1. **Quality and Diversity of Data:**
    - **Challenge:** making high-quality, variety data is important for a strong model. Poor sensor accuracy, incorrect user input, and same user statistical data can harm model performance**.**
    - **Approach:** Implementing strict validation checks on incoming data for irregularities and ensure variety by encouraging a wide variety of users through targeted marketing and community engagement.
2. **Real-Time Data Processing Needs:**
    - **Challenge:** Real-time feedback is important for adjusting workouts dynamically. Processing large streams of data in real-time presents significant computational challenges.
    - **Approach**: edge computing to handle real-time data analysis locally on the device, reduce the need for continuous data transmission and speed up response times.

Training and Inference Pipeline

Pipeline Overview:

- **Data Preprocessing:** Device data will be cleaned up and normalized so that it is the same from session to session.
- **Model Training**: train machine learning models using past aggregated data in the cloud, where computational resources are scalable**.**
- **Model Deployment**: Trained models are deployed back on the devices or kept in the cloud depends on inference.
- **Inference**: For real-time adjustments during workouts, inference happens locally on the devices using light models

Execution Environment:

- **Local (Edge):** Local inference on the devices for fast feedback using simple predictive models.
- **Cloud:** Complex computations and historical data analyses are run in the cloud to use more powerful computational resources and storage capabilities**.**

Performance Metrics

1. **Accuracy of Weight and Tension Settings:**
    - **Importance**: makes the device accurately adjusts to the user's selected settings
    - **Measurement**: Percentage of times the device achieves the preferred settings with a less error margin.
2. **User Retention Rate:**
    - **Importance:** A high retention rate indicates user satisfaction and the system's efficiency in meeting fitness goals.
    - **Measurement:** The percentage of users who continue using the app after the initial month.
3. **Response Time for Adjustments:**
    - **Importance:** Important for real-time apps, where delays can disrupt workout flow and affect user experience.
    - **Measurement:** Time taken from the moment an adjustment is triggered to when it is executed.
4. **Predictive Accuracy:**
    - **Importance:** Measures how well the system predicts the appropriate weights/tensions based on past user performance and preferences.
    - Measurement: Comparison of predicted settings vs actual effective settings that led to successful workouts.

Managing Human-Computer Interaction:

An in-depth knowledge of Human-Computer Interaction (HCI) must be incorporated into the creation of an IoT fitness system in order to make it usable and increase user engagement. This system comes with smart dumbbells and resistance bands that can both be controlled by a mobile app. The app also tracks performance and gives feedback. Let's take a look at how people will interact with this system. We'll focus on the steps involved and think about how they will affect users (fitness fans), personal trainers, and equipment managers.

Storyboard for User Interaction with the IoT Fitness System

Frame 1: Login and Initial Setup

- **User Interaction:** Users download the mobile app and create an account. During the setup, they input basic fitness goals and preferences.
- **System Response:** app provides a modified dashboard based on the user's fitness goals.
- **Stakeholder Impact:** Personal trainers can review user goals to customize workout plans.

Frame 2: Connecting Devices

- **User Interaction:** Users power on their smart dumbbells and resistance bands. They use the app to find and connect devices via Bluetooth.
- **System Response:** app confirms each device's connection and battery status.
- **Stakeholder Impact:** Equipment managers monitor the status and maintenance needs of devices through backend .

Frame 3: Configuring a Workout

- **User Interaction:** Users select a workout type on the app, which suggests weights and resistance settings based on their fitness level and past progress.
- **System Response:** dumbbells and bands automatically adjust to the recommended settings.
- **Stakeholder Impact:** Personal trainers use data collected during sessions to refine and update training .
-

Frame 4: During the Workout

- **User Interaction:** users perform exercises, the devices track movement accuracy and effort.
- **System Response:** app provides real-time audio and visual feedback on form, speed, and tension adjustments.
- **Stakeholder Impact:** Users receive immediate corrections, enhancing training effectiveness and reducing injury risks.

Frame 5: Post-Workout Review

- **User Interaction:** After completing the workout, users rate the session's difficulty and how they feel.
- **System Response:** app updates the user's progress dashboard and suggests recovery tips.
- **Stakeholder Impact:** Personal trainers review session feedback to further personalize the user's fitness program.

Frame 6: Performance Analytics and Sharing

- **User Interaction:** Users access detailed analytics on their workouts over time and can share achievements with friends or social media.
- **System Response:** app provides graphs of progress.
- **Stakeholder Impact:** Equipment managers and trainers use all the data to improve system recommendations .

Explanation of the HCI Considerations
For a fun and useful exercise experience, the storyboard shows how IoT technology can be easily combined with easy-to-use interfaces. Each interaction point is made to be easy to use, giving us data and changes right when we need them to improve our workout while causing the least amount of trouble. This method not only helps the person doing the workout by making it more fun and effective, but it also helps personal trainers and equipment managers do their jobs better by giving them detailed information to support them.

Personal trainer benefit from having access to detailed workout data, which allows them to offer a more personalized and modified coaching experience. Equipment managers, on the other hand, can ensure equipment life and efficient performance through continuous monitoring and maintenance updates.

By carefully managing these human-computer interactions, the design concept assure to enhance the fitness routines of users, support the professional roles of trainers.

Managing Security and Privacy in Smartfit

The smart dumbbells and resistance bands connected to a mobile app, several security and privacy risks come up. These devices collect, transmit, and store private user data, which could be unprotected to various security threats.

Security and Privacy Risks

1. **Data Interception:** There is a chance that someone could steal the data that IoT devices like smart dumbbells and resistance bands send over the network to a mobile app or cloud server. Depending on what is sent, cybercriminals could steal this information to learn about a user's workouts or even personal information.

2. **Unauthorized Access:** If security measures are weak, an attacker could gain access to the IoT device itself allowing them to manipulate device functionality. For instance, changing the resistance or weight settings could lead to personal injury.

3. **Data Privacy:** The app collects detailed information about a user's health metrics, workout preferences, and potentially even location data. This sensitive information could be misused if accessed by unauthorized parties or mishandled by the service providers.

4. **Device Firmware Security:** IoT devices operate on firmware that could be exploited if not updated or securely managed. An attacker could inject malware into the firmware, affecting the device's operation and compromising user data.

Solutions to Mitigate Risks

1. **Secure Data Transmission:** Implement end-to-end encryption for all data transmitted between the IoT devices, mobile app, and servers. Using protocols like TLS (Transport Layer Security)

2. **Robust Authentication Mechanisms:** Utilize multi-factor authentication (MFA) for device and app access, ensuring that only authorized users can control the devices and access data.

3. **Regular Updates and Patch Management:** Regularly update the firmware and software of both the IoT devices and the mobile app

4. **Data Minimization and Anonymization:** Limit the collection of personal data to what is absolutely necessary for the functionality of the app. Prevent data it from being traced back to an individual, when used for analytics or performance improvements.

5. **Privacy by Design:** Include privacy considerations into the development process of the IoT devices and app. This includes features like user-controlled privacy settings that let users decide what data they are comfortable sharing.

6. **Network Security:** Implement network security measures such as firewalls.

Other Forms of Risk

- **Physical Security of Devices:** make sure the devices are tamper-resistant and have secure storage for cryptographic keys used in encryption.

- **Compliance with Regulations:** Stick to relevant legal frameworks such as GDPR for handling personal data, which includes ensuring transparency in data collection and providing users with control over their data.

Taking these security and privacy risks into account ahead of time can make the smart fitness system rollout safer and more reliable, protecting both user trust and the stability of the system. These steps not only keep you safe from cyber threats, but they also help make the exercise centre a more private place.

Approaches to Evaluation:

Component-level Testing

Test Type	Test Name	Introduction	Acceptance Criteria	Test Execution	Strategy
Usability	User Experience Test	Evaluate user satisfaction and usability of the system.	High user satisfaction ratings and positive feedback.	In-person testing	Structured user sessions with observation and feedback collection.
Usability	Accessibility Test	Ensure the system is usable by people with a range of abilities.	Compliance with ADA standards and positive user feedback.	Manual testing	Sessions with users having different accessibility needs.

Component-Level Testing

- **Objective:** Make sure that each part (sensors, actuators, communication units, etc.) works as it should on its own.
- **Tests:**
  - **Sensor Accuracy Test:** Sensors should be able to correctly measure motion, weight, and tension. Criteria for acceptance: There must be a 2% error margin between measurements and their true numbers.
  - **Battery Performance Test:** Check how long the battery lasts when it's being used normally. Criteria for acceptance: devices must work steadily for at least two hours, which is the length of a normal workout session.

System-Level Testing

Test Type	Test Name	Introduction	Acceptance Criteria	Test Execution	Strategy
Integration	End-to-End System Test	Test all system components working together from sensors to user interface.	System performs seamlessly from input to output.	Automated testing	Simulate user interactions and system responses.
Performance	Load Testing	Assess system performance under peak loads.	System handles maximum expected users without lag.	Automated stress tests	Use load testing software to simulate multiple users.

- **Objective:** Check the operation of the integrated system to make sure that all of its parts work together well.
- **Tests:**
  - **Integration Test:** End-to-end tests make sure that all of the system's parts work together properly and that no data is lost or errors happen. Criteria for acceptance: Ten test scenarios of smooth running with no integration errors.
  - **Load Test:** Finding out if the system can handle the most people at the same time is important. Criteria for acceptance: The system must be able to handle up to 100 user sessions at the same time without slowing down.
  - ○

User Trials:

Test Type	Test Name	Introduction	Acceptance Criteria	Test Execution	Strategy
Usability	User Experience Test	Evaluate user satisfaction and usability of the system.	High user satisfaction ratings and positive feedback.	In-person testing	Structured user sessions with observation and feedback collection.
Usability	Accessibility Test	Ensure the system is usable by people with a range of abilities.	Compliance with ADA standards and positive user feedback.	Manual testing	Sessions with users having different accessibility needs.

- **Objective:** Validate usability, accessibility, and overall user satisfaction.
- **Tests:**
    - **Usability Testing:** Test with different groups of users to see how simple and easy it is to use. Acceptance criteria: 85% of users must say they are happy with how easy it is to use the system.

Accessibility Evaluation: Ensure the system is accessible to users with varying abilities. Acceptance criteria: Compliance with ADA (Americans with Disabilities Act) standards

Non-Functional Requirements:

	Test Name	Introduction	Acceptance Criteria	Test Execution	Strategy
Performance	Bandwidth Usage Test	Verify the system operates efficiently under various network conditions.	Maintains functionality at different bandwidth levels.	Automated tests	Simulate different network speeds and conditions.
Reliability	Reliability Test	Test system reliability over extended periods.	System uptime of 99% over six months.	Automated monitoring	Continuous operation monitoring with automated reporting.
Security	Security Penetration Test	Identify vulnerabilities in the system that could be exploited.	No critical vulnerabilities found.	External security audits	Engage with security professionals to conduct tests.
Scalability	Scalability Test	Determine the system's capacity to handle growth in users and data volume.	System performance remains stable as scale increases.	Load testing	Gradually increase load while monitoring performance metrics.

- **Objective:** Ensure the system meets operational benchmarks beyond basic functionality**.**
- **Tests:**
    - **Communication Bandwidth Requirement:** Check how much data is sent during normal use to make sure it's within the speed limits of most home networks. Acceptance criteria: The system can still do everything it needs to do at bandwidths as low as 256 kbps.
    - **Energy Requirement:** Check that energy use is in line with goals for energy economy by keeping an eye on it. Acceptance criteria: Devices must not use more than 10% more energy than was planned when they were designed.
    - **Reliability Test:** Check the system's dependability over long periods of time. Criteria for acceptance: A 99.5% uptime rate for the system over a 6-month testing period.
    - **Security Penetration Testing:** Get rid of any possible security holes found. Criteria for acceptance: There should be no major vulnerabilities and all issues of medium to low severity must be fixed.

Risks and Mitigation Strategies

Risk 1: Device Malfunction in Real-World Settings

- **Mitigation:** A lot of tests in different environments with changing temperature, humidity, and mechanical stress to make it feel like it would be used in the real world. Real users are used in beta testing during controlled release phases to get feedback and operational data for making changes over time.

Risk 2: Data Privacy Breaches

- **Mitigation:** Use cutting edge encryption for both data that is at rest and data that is being sent. Update security measures often and do regular security audits. Teach people how to use technology safely.